Parameterized Approximation Algorithms for Bidirected Steiner Network Problems

Rajesh Chitnis*[1], Andreas Emil Feldmann†[2], and Pasin Manurangsi‡[3]

[1]University of Birmingham, UK. rajeshchitnis@gmail.com
[2]Charles University, Czechia. feldmann.a.e@gmail.com
[3]University of California, Berkeley, USA. pasin@berkeley.edu.

Abstract

The Directed Steiner Network (DSN) problem takes as input a directed graph $G = (V, E)$ with non-negative edge-weights and a set $\mathcal{D} \subseteq V \times V$ of k demand pairs. The aim is to compute the cheapest network $N \subseteq G$ for which there is an $s \to t$ path for each $(s, t) \in \mathcal{D}$. It is known that this problem is notoriously hard as there is no $k^{1/4-o(1)}$-approximation algorithm under Gap-ETH, even when parametrizing the runtime by k [Dinur & Manurangsi, ITCS 2018]. In light of this, we systematically study several special cases of DSN and determine their parameterized approximability for the parameter k.

For the bi-DSN$_{\text{Planar}}$ problem, the aim is to compute a planar optimum solution $N \subseteq G$ in a bidirected graph G, i.e. for every edge uv of G the reverse edge vu exists and has the same weight. This problem is a generalization of several well-studied special cases. Our main result is that this problem admits a parameterized approximation scheme (PAS) for k. We also prove that our result is tight in the sense that (a) the runtime of our PAS cannot be significantly improved, and (b) no PAS exists for any generalization of bi-DSN$_{\text{Planar}}$, under standard complexity assumptions. The techniques we use also imply a polynomial-sized approximate kernelization scheme (PSAKS). Additionally, we study several generalizations of bi-DSN$_{\text{Planar}}$ and obtain upper and lower bounds on obtainable runtimes parameterized by k.

One important special case of DSN is the Strongly Connected Steiner Subgraph (SCSS) problem, for which the solution network $N \subseteq G$ needs to strongly connect a given set of k terminals. It has been observed before that for SCSS a parameterized 2-approximation exists for parameter k [Chitnis et al., IPEC 2013]. We give a tight inapproximability result by showing that for k no parameterized $(2 - \varepsilon)$-approximation algorithm exists under Gap-ETH. Additionally, we show that when restricting the input of SCSS to bidirected graphs, the problem remains NP-hard but becomes FPT for k.

*Supported by ERC grant 2014-CoG 647557. Part of this work was done while at Weizmann Institute of Science, Israel (and supported by Israel Science Foundation grant #897/13) and visiting Charles University in Prague, Czechia

†Supported by the Czech Science Foundation GAČR (grant #19-27871X), and by the Center for Foundations of Modern Computer Science (Charles Univ. project UNCE/SCI/004).

‡This work was done while the author was visiting Weizmann Institute of Science. Currently at Google Research.

1 Introduction

In this work we study the Directed Steiner Network (DSN) problem,[1] in which a directed graph $G = (V, E)$ with non-negative edge weights is given together with a set of k *demands* $\mathcal{D} = \{(s_i, t_i)\}_{i=1}^{k} \subseteq V \times V$. The aim is to compute a minimum cost (in terms of edge weights) network $N \subseteq G$ containing a directed $s_i \to t_i$ path for each $i \in \{1, \ldots, k\}$. This problem has applications in network design [49], and for instance models the setting where nodes in a radio or ad-hoc wireless network connect to each other unidirectionally [13, 73].

The DSN problem is notoriously hard. First of all, it is NP-hard, and one popular way to handle NP-hard problems is to efficiently compute an *α-approximation*, i.e., a solution that is guaranteed to be at most a factor α worse than the optimum. For this paradigm we typically demand that the algorithm computing such a solution runs in polynomial time in the input size $n = |V|$. However for DSN it is known that even computing an $O(2^{\log^{1-\varepsilon} n})$-approximation is not possible [24] in polynomial time, unless NP $\subseteq$ DTIME($n^{\text{polylog}(n)}$). It is possible to obtain approximation factors $O(n^{2/3+\varepsilon})$ and $O(k^{1/2+\varepsilon})$ though [5, 11, 33]. For settings where the number k of demands is fairly small, one may aim for algorithms that only have a mild exponential runtime blow-up in k, i.e., a runtime of the form $f(k) \cdot n^{O(1)}$, where $f(k)$ is some function independent of n. If an algorithm computing the optimum solution with such a runtime exists for a computable function $f(k)$, then the problem is called *fixed-parameter tractable (FPT)* for parameter k. However it is unlikely that DSN is FPT for this well-studied parameter, as it is known to be W[1]-hard [42] when parameterized by k. In fact one can show [17] that under the *Exponential Time Hypothesis (ETH)* there is no algorithm computing the optimum in time $f(k) \cdot n^{o(k)}$ for any function $f(k)$ independent of n. ETH assumes that there is no $2^{o(n)}$ time algorithm to solve 3SAT [44, 45]. The best we can hope for is therefore a so-called *XP-algorithm* computing the optimum in time $n^{O(k)}$, and this was also shown to exist by Feldman and Ruhl [32].

None of the above algorithms for DSN seem satisfying though, either due to slow runtimes or large approximation factors, and this is hardly surprising given the problem's complexity, as mentioned above. To circumvent the hardness of the problem, one may aim for *parameterized approximations*, which have recently received increased attention for various problems. In this paradigm an α-approximation is computed in time $f(k) \cdot n^{O(1)}$ for parameter k, where $f(k)$ again is a computable function independent of n. Unfortunately, a recent result by Dinur and Manurangsi [23] excludes significant improvements over the known polynomial time approximation algorithms [5, 11, 33], even if allowing a runtime parameterized in k. More specifically, no $k^{1/4-o(1)}$-approximation is possible[2] in time $f(k) \cdot n^{O(1)}$ for any function $f(k)$ under the *Gap Exponential Time Hypothesis (Gap-ETH)*, which postulates that there exists a constant $\varepsilon > 0$ such that no (possibly randomized) algorithm running in $2^{o(n)}$ time can distinguish whether all or at most a $(1-\varepsilon)$-fraction of clauses of any given 3SAT formula can be satisfied[3] [22, 60].

Given these hardness results, the main question we explore is: what approximation factors and runtimes are possible for special cases of DSN when parameterizing by k? There are two types of standard special cases that are considered in the literature:

- Restricting the input graph G to some special graph class. A typical assumption for instance is that G is planar (where a directed graph is planar if the underlying undirected graph is).

[1] Also sometimes called Directed Steiner Forest. Note however that in contrast to the undirected Steiner Forest problem, an optimum solution to DSN is not necessarily a forest.

[2] In a previous version of this work, we showed that no $k^{o(1)}$-approximation is possible for DSN in time $f(k) \cdot n^{O(1)}$. This result in now subsumed by [23]; see Section 8 for more details.

[3] Gap-ETH follows from ETH given other standard conjectures, such as the existence of linear sized PCPs or exponentially-hard locally-computable one-way functions. See [3, 9] for more details.

- Restricting the pattern of the demands in $\mathcal{D}$. For example, one standard restriction is to have a set $R \subseteq V$ of *terminals*, a fixed *root* $r \in R$, and demand set $\mathcal{D} = \{(r,t) \mid t \in R\}$, which is the well-known Directed Steiner Tree (DST) problem.

In fact, an optimum solution to the DST problem is an arborescence (hence the name), i.e., it is planar. Thus if an algorithm is able to compute a solution that costs at most as much as the cheapest planar DSN solution in an otherwise unrestricted graph, it can be used for both the above types of restrictions: it can of course be used if the input graph is planar as well, and it can also be used if the demand pattern implies that the optimum must be planar. Taking the structure of the optimum solution into account has been a fruitful approach leading to several results on related problems, both for approximation and fixed-parameter tractability, from which we also draw some of the inspiration for our results (cf. Section 1.2). A main focus of our work is to systematically explore the influence of the structure of solutions on the complexity of the DSN problem. Formally, fixing a class $\mathcal{K}$ of graphs, we define the DSN$_{\mathcal{K}}$ problem, which asks for a solution network $N \subseteq G$ for k given demands such that the cost of N is at most that of any feasible solution in G belonging to the class $\mathcal{K}$. Note that the solution N does not have to belong to the class $\mathcal{K}$. As explained for the class of planar graphs above, DSN$_{\mathcal{K}}$ can be thought of as the special case that lies between restricting the input to the class $\mathcal{K}$ and the general unrestricted case.

The DSN$_{\mathcal{K}}$ problem has been implicitly studied in several results before for various classes $\mathcal{K}$ (cf. Table 1), in particular when $\mathcal{K}$ contains either planar graphs, or graphs of bounded treewidth (here the *undirected* treewidth is meant, i.e., the treewidth of the underlying undirected graph). For these results, typically an algorithm is given that computes a solution for an input of a class $\mathcal{K}$, but the algorithm is in fact more general and can also be applied to the corresponding DSN$_{\mathcal{K}}$ problem. Our algorithms presented in this paper for the class $\mathcal{K}$ of planar graphs are also of this type. The reader may therefore want to think of the case when the input is planar for our algorithms. On the other hand, our corresponding hardness results are for the more general DSN$_{\mathcal{K}}$ problem, which means that they rule out algorithms of this general type. In particular, they can be interpreted as saying that if there are algorithms for input graphs from $\mathcal{K}$ that beat our lower bounds for the more general DSN$_{\mathcal{K}}$ problem, then they cannot be of the general type that seems prevalent in the study of the DSN problem for special input graphs.

Another special case we consider is the bi-DSN problem, where the input graph G is *bidirected*, i.e., for every edge uv of G the reverse edge vu exists in G as well and has the same weight as uv. This in turn can be understood as the case lying between undirected and directed graphs, since bidirected graphs are directed, but, similar to undirected graphs, a path can be traversed in either direction at the same cost. Bidirected graphs model the realistic setting [13, 54, 73, 78] when the cost of transmitting from a node u to a node v in a wireless network is the same in both directions, which for instance happens if the nodes all have the same transmitter model.

We systematically study several special cases of DSN resulting from the above restrictions, and prove several matching upper and lower bounds on runtimes parameterized by k. We now give a brief overview of the studied problems, and refer to Section 1.1 for a detailed exposition of our results.

bi-DSN$_{\text{Planar}}$, i.e., the DSN$_{\mathcal{K}}$ problem on bidirected inputs, where $\mathcal{K}$ is the class of planar graphs: For this problem we present our main result, which is that bi-DSN$_{\text{Planar}}$ admits a *parameterized approximation scheme (PAS)*, i.e., an algorithm that for any $\varepsilon > 0$ computes a $(1+\varepsilon)$-approximation in $f(\varepsilon,k) \cdot n^{g(\varepsilon)}$ time for some functions f and g. We also prove that, unless FPT=W[1], no *efficient parameterized approximation scheme (EPAS)* exists, i.e., there is no algorithm computing a $(1+\varepsilon)$-approximation in $f(\varepsilon,k) \cdot n^{O(1)}$ time for any computable function f. Thus the degree of the polynomial runtime dependence on n has to depend on ε.

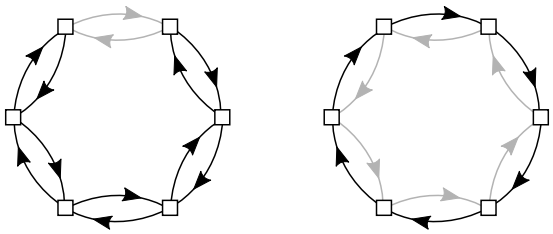

Figure 1: A BI-SCSS instance where all vertices are terminals. Left: Black edges show a solution which takes an undirected optimum twice. Right: The actual optimum solution is shown in black.

BI-DSN, i.e., the DSN problem on bidirected inputs: The above PAS for the rather restricted BI-DSN$_{\text{PLANAR}}$ problem begs the question of whether a PAS also exists for any more general problems, such as BI-DSN. In particular, one may at first think that BI-DSN closely resembles the undirected variant of DSN, i.e., the well-known STEINER FOREST (SF) problem, which is FPT [26, 34] for parameter k. Surprisingly however, we can show that BI-DSN is almost as hard as DSN (with almost-matching runtime lower bound under ETH), and moreover, no PAS exists under Gap-ETH.

Apart from the DST problem, another well-studied special case of DSN with restricted demands is when the demand pairs form a cycle, i.e., we are given a set $R = \{t_1, \ldots, t_k\}$ of k terminals and the set of demands is $\mathcal{D} = \{(t_i, t_{i+1})\}_{i=1}^{k}$ where $t_{k+1} = t_1$. Since this implies that any optimum solution is strongly connected, this problem is accordingly known as the STRONGLY CONNECTED STEINER SUBGRAPH (SCSS) problem. In contrast to DST, it is implicit from [42] (by a reduction from the CLIQUE problem) that optimum solutions to SCSS do not belong to any minor-closed graph class. Thus SCSS is not easily captured by some DSN$_{\mathcal{K}}$ problem for a restricted class $\mathcal{K}$. Nevertheless it is still possible to exploit the structure of the optimum solution to SCSS, which results in the following findings.

SCSS: It is known that a 2-approximation is obtainable [15] when parameterizing by k. We prove that the factor of 2 is best possible under Gap-ETH. To the best of our knowledge, this is the first example of a problem with a tight parameterized approximation result with non-trivial approximation factor (in this case 2), which also beats any approximation computable in polynomial time.

BI-SCSS, i.e., the SCSS problem on bidirected inputs: As for BI-DSN, one might think that BI-SCSS is easily solvable via its undirected version, i.e., the well-known STEINER TREE (ST) problem. In particular, the ST problem is FPT [26] for parameter k. However, it is not the case that simply taking an optimum undirected solution twice in a bidirected graph will produce a (near-)optimum solution to BI-SCSS (see Figure 1). Nevertheless we prove that BI-SCSS is FPT for parameter k as well, while also being NP-hard. Our algorithm is non-trivial and does not apply any methods used for undirected graphs. To the best of our knowledge, bidirected inputs are the first example where SCSS remains NP-hard but turns out to be FPT parameterized by k.

1.1 Our results

Bidirected inputs with planar solutions. Our main theorem implies the existence of a PAS for BI-DSN$_{\text{PLANAR}}$, where the parameter is the number k of demands.

Theorem 1.1. *There is a $2^{O(k^2)}n^{2^{O(1/\varepsilon)}}$ time algorithm for* BI-DSN$_{\text{PLANAR}}$*, that for any $\varepsilon > 0$ computes a $(1+\varepsilon)$-approximation.*

This result begs the question of whether the considered special case is not too restrictive. Should it not be possible to obtain better runtimes and/or should it not be possible to even compute the optimum solution when parameterizing by k for this very restricted problem? And could it not be that a similar result is true in more general settings, when for instance the input is bidirected but the optimum is not restricted to a planar graph? We prove that both questions can be answered in the negative.

First off, it is not hard to prove that a *polynomial time approximation scheme (PTAS)* is not possible for BI-DSN$_{\text{PLANAR}}$, i.e., it is necessary to parameterize by k in Theorem 1.1. This is implied by the following result, since (as mentioned before) a PTAS for BI-DSN$_{\text{PLANAR}}$ would also imply a PTAS for BI-DST, i.e., the DST problem on bidirected input graphs.

Theorem 1.2. *The* BI-DST *problem (and by extension also the* BI-DSN$_{\text{PLANAR}}$ *problem) is* APX-hard.

One may wonder however, whether parameterizing by k doesn't make the BI-DSN$_{\text{PLANAR}}$ problem FPT, so that approximating the planar optimum as in Theorem 1.1 would in fact be unnecessary. Furthermore, even if it is necessary to approximate, one may ask whether the runtime given in Theorem 1.1 can be improved. In particular, note that the runtime we obtain in Theorem 1.1 is similar to that of a PTAS, i.e., the exponent of n in the running time depends on ε. Ideally we would like an EPAS, which has a runtime of the form $f(k,\varepsilon) \cdot n^{O(1)}$, i.e., we would like to treat ε as a parameter as well. The following theorem shows that both approximating and runtime dependence on ε are in fact necessary in Theorem 1.1.

Theorem 1.3. *The* BI-DSN$_{\text{PLANAR}}$ *problem is* W[1]*-hard parameterized by* k*. Moreover, under ETH, for any computable functions* $f(k)$ *and* $f(k,\varepsilon)$*, the* BI-DSN$_{\text{PLANAR}}$ *problem*

- *has no* $f(k) \cdot n^{o(\sqrt{k})}$ *time algorithm to compute an optimum solution, i.e., a solution with cost at most that of the cheapest planar one, and*
- *has no* $f(k,\varepsilon) \cdot n^{o(\sqrt{k})}$ *time algorithm to compute a solution with cost at most* $(1+\varepsilon)$ *times that of the cheapest planar one, if* $\varepsilon > 0$ *is part of the input.*

It stands out that to compute optimum solutions, this theorem rules out runtimes for which the dependence of the exponent of n is $\sqrt{k}$, while for the general DSN problem, as mentioned above, the both necessary and sufficient dependence of the exponent is linear in k [17, 32]. Could it be that BI-DSN$_{\text{PLANAR}}$ is just as hard as DSN when computing optimum solutions? The answer is no, as the next theorem shows.

Theorem 1.4. *There is a* $2^{O(k^{3/2}\log k)} \cdot n^{O(\sqrt{k})}$ *time algorithm to compute the optimum solution for* BI-DSN$_{\text{PLANAR}}$*, i.e., a solution with cost at most that of the cheapest planar one.*

This result is an example of the so-called "square-root phenomenon": planarity often allows runtimes that improve the exponent by a square root factor in terms of the parameter when compared to the general case [37, 51, 52, 56, 63, 64, 65, 70, 71]. Interestingly though, Chitnis et al. [17] show that under ETH, no $f(k) \cdot n^{o(k)}$ time algorithm can compute the optimum solution to DSN$_{\text{PLANAR}}$. Thus assuming a bidirected input graph in Theorem 1.4 is necessary (under ETH) to obtain a factor of $O(\sqrt{k})$ in the exponent of n.

Bidirected inputs. Since in contrast to BI-DSN$_{\text{PLANAR}}$, the BI-DSN problem does not restrict the optimum solutions, one may wonder whether a parameterized approximation scheme as in Theorem 1.1 is possible for this more general case as well. We answer this in the negative by proving the following result, which implies that restricting the optima to planar graphs was necessary for Theorem 1.1.

Theorem 1.5. *Under Gap-ETH, there exists a constant $\alpha > 1$ such that for any computable function $f(k)$ there is no $f(k) \cdot n^{O(1)}$ time algorithm that computes an α-approximation for* BI-DSN.

Also for the other obvious generalization of BI-$\text{DSN}_{\text{PLANAR}}$, in which the input graph is unrestricted but we need to compute the planar optimum (i.e., the $\text{DSN}_{\text{PLANAR}}$ problem), no parameterized approximation scheme exists. This follows from a recent result [14], which shows that no $(2-\varepsilon)$-approximation can be computed for $\text{DSN}_{\text{PLANAR}}$ in $f(k) \cdot n^{O(1)}$ time for any $\varepsilon > 0$ and computable function f, under Gap-ETH.

What approximation factors can be obtained for BI-DSN when parameterizing by k, given the lower bound of Theorem 1.5 on one hand, and the before-mentioned result [23] that rules out a $k^{1/4-o(1)}$-approximation for DSN in time parameterized by k on the other? It turns out that it is not too hard to obtain a constant approximation for BI-DSN, given the similarity of bidirected graphs to undirected graphs. In particular, relying on the fact that for the undirected version of DSN, i.e. the SF problem, there is a polynomial time 2-approximation algorithm by Agrawal et al. [1], and an FPT algorithm based on Dreyfus and Wagner [26], we obtain the following theorem, which is also in contrast to Theorem 1.2.

Theorem 1.6. *The* BI-DSN *problem admits a 4-approximation in polynomial time, and a 2-approximation in $2^{O(k)} \cdot n^{O(1)}$ time.*

Even if Theorem 1.5 in particular shows that BI-DSN cannot be FPT under Gap-ETH, it does not give a strong lower bound on the runtime dependence in the exponent of n. However using the weaker ETH assumption we can obtain such a lower bound, as the next theorem shows. Interestingly, the obtained lower bound implies that when aiming for optimum solutions, the restriction to bidirected inputs does not make DSN easier than the general case, as also for BI-DSN the $n^{O(k)}$ time algorithm by Feldman and Ruhl [32] is essentially best possible. This is in contrast to the BI-$\text{DSN}_{\text{PLANAR}}$ problem where the square-root phenomenon takes effect as shown by Theorem 1.4.

Theorem 1.7. *The* BI-DSN *problem is* W[1]*-hard parameterized by k. Moreover, under ETH there is no $f(k) \cdot n^{o(k/\log k)}$ time algorithm for* BI-DSN*, for any computable function $f(k)$.*

Strongly connected solutions. Just like the more general DSN problem, the SCSS problem is W[1]-hard [42] parameterized by k, and is also hard to approximate as no polynomial time $O(\log^{2-\varepsilon} n)$-approximation is possible [43], unless NP $\subseteq$ ZTIME$(n^{\text{polylog}(n)})$. However it is possible to exploit the structure of the optimum to SCSS to obtain a 2-approximation algorithm parameterized by k, as observed by Chitnis et al. [15]. This is because any strongly connected graph is the union of two arborescences, and these form solutions to DST. The 2-approximation follows, since DST is FPT by the classic result of Dreyfus and Wagner [26]. Thus in contrast to DSN, for SCSS it is possible to beat any approximation factor obtainable in polynomial time when parameterizing by k.

Theorem 1.8 ([15])**.** *The* SCSS *problem admits a 2-approximation in $3^k \cdot n^{O(1)}$ time.*

An obvious question now is whether the approximation ratio of this rather simple algorithm can be improved. Interestingly we are able to show that this is not the case. To the best of our knowledge, this is the first example of a problem with a tight parameterized approximation result with non-trivial approximation factor (in this case 2), which also beats any approximation computable in polynomial time.

Theorem 1.9. *Under Gap-ETH, for any $\varepsilon > 0$ and any computable function $f(k)$, there is no $f(k) \cdot n^{O(1)}$ time algorithm that computes a $(2-\varepsilon)$-approximation for* SCSS.

We remark that our reduction for Theorem 1.9 uses edge weights, which however can be polynomially bounded. As a consequence an instance can be further reduced in polynomial time by first scaling the edge weights to polynomially bounded integers, and then subdividing each edge w times if its weight is w. This results in an equivalent unweighted instance, and thus the lower bound of Theorem 1.9 is also valid for unweighted instances of SCSS.

Bidirected inputs with strongly connected solutions. In light of the above results for restricted cases of DSN, what can be said about restricted cases of SCSS? It is implicit in the work of Chitnis et al. [17] that $\text{SCSS}_{\text{PLANAR}}$, i.e., the problem of computing a solution of cost at most that of the cheapest strongly connected planar solution, can be solved in $2^{O(k \log k)} \cdot n^{O(\sqrt{k})}$ time, while under ETH no $f(k) \cdot n^{o(\sqrt{k})}$ time algorithm is possible. Hence $\text{SCSS}_{\text{PLANAR}}$ is slightly easier than $\text{DSN}_{\text{PLANAR}}$ where the exponent of n needs to be linear in k, as mentioned before. On the other hand, the BI-SCSS problem turns out to be a lot easier to solve than BI-DSN. This is implied by the next theorem, which stands in contrast to Theorem 1.5 and Theorem 1.7.

Theorem 1.10. *There is a $2^{k^2+O(k)} \cdot n^{O(1)}$ time algorithm for* BI-SCSS*, i.e., it is FPT for parameter k.*

Could it be that BI-SCSS is even solvable in polynomial time? We prove that this is not the case, unless P = NP. To the best of our knowledge, the class of bidirected graphs is the first example where SCSS remains NP-hard but turns out to be FPT parameterized by k. Moreover, note that the above algorithm has an exponential runtime in k^2. We conjecture that a single exponential runtime should suffice, and we also obtain a lower bound result of this form.

Theorem 1.11. *The* BI-SCSS *problem is NP-hard. Moreover, under ETH there is no $2^{o(k)} \cdot n^{O(1)}$ time algorithm for* BI-SCSS*.*

Remark. For ease of notation, throughout this paper we chose to use the number of demands k uniformly as the parameter. Alternatively one might also consider the smaller parameter $|R|$, where $R = \bigcup_{i=1}^{k} \{s_i, t_i\}$ is the set of terminals (as also done in [29]). Note for instance that in case of the SCSS problem, $k = |R|$, while for DSN, k can be as large as $\Theta(|R|^2)$. However we always have $k \geq |R|/2$, since the demands can form a matching in the worst case. It is interesting to note that all our algorithms for DSN have the same running time for parameter $|R|$ as for parameter k. That is, we may set $k = |R|$ in Theorem 1.1, 1.4, and 1.6.

1.2 Our techniques

It is already apparent from the above exposition of our results, that understanding the structure of the optimum solution is a powerful tool when studying DSN and its related problems (cf. Table 1). This is also apparent when reading the literature on these problems, and we draw some of our inspiration from these known results, as described below.

Approximation scheme for $\text{BI-DSN}_{\text{PLANAR}}$. We generalize the insights on the structure of optimum solutions to the classical STEINER TREE (ST) problem for our main result in Theorem 1.1. For the ST problem, an *undirected* edge-weighted graph is given together with a terminal set R, and the task is to compute the cheapest tree connecting all k terminals. For this problem only polynomial-time 2-approximations were known [40, 75], until it was taken into account [48, 72, 74, 80] that any optimum Steiner tree can be decomposed into so-called *full components*, i.e., subtrees for which exactly the leaves are terminals. If a full component contains only a small subset of size k' of the terminals, it is the solution to an ST instance, for

problem	algorithms approx.	algorithms runtime	algorithms ref.	lower bounds approx.	lower bounds runtime	lower bounds ref.
DSN	–	$n^{O(k)}$	[32]	–	$f(k) \cdot n^{o(k)}$	[29, 42]
DSN	$O(k^{\frac{1}{2}+\varepsilon})$	$n^{O(1)}$	[11]	$k^{\frac{1}{4}-o(1)}$	$f(k) \cdot n^{O(1)}$	[23]
$\text{DSN}_{\text{TW: }\omega}$	–	$2^{O(k\omega \log \omega)} \cdot n^{O(\omega)}$	[34]	–	$f(k,\omega) \cdot n^{o(\omega)}$	[34]
$\text{BI-DSN}_{\text{PLANAR}}$	–	$2^{O(k^{3/2} \log k)} \cdot n^{O(\sqrt{k})}$	Thm 1.4	–	$f(k) \cdot n^{o(\sqrt{k})}$	Thm 1.3
$\text{BI-DSN}_{\text{PLANAR}}$	$1+\varepsilon$	$2^{O(k^2)} n^{2^{O(1/\varepsilon)}}$	Thm 1.1	$1+\varepsilon$	$f(\varepsilon,k) \cdot n^{o(\sqrt{k})}$	Thm 1.3
$\text{DSN}_{\text{PLANAR}}$	–	$n^{O(k)}$	[29, 32]	–	$f(k) \cdot n^{o(k)}$	[17]
$\text{DSN}_{\text{PLANAR}}$	$\alpha \geq 2$	(open)		$(2-\varepsilon)$	$f(k) \cdot n^{O(1)}$	[14]
BI-DSN	–	$n^{O(k)}$	[32]	–	$f(k) \cdot n^{o(k/\log k)}$	Thm 1.7
BI-DSN	2	$2^{O(k)} \cdot n^{O(1)}$	Thm 1.6	$\alpha \in \Theta(1)$	$f(k) \cdot n^{O(1)}$	Thm 1.5
BI-DSN	4	$n^{O(1)}$	Thm 1.6	$\alpha \in \Theta(1)$	$n^{O(1)}$	Thm 1.2
SCSS	–	$n^{O(k)}$	[32]	–	$f(k) \cdot n^{o(k/\log k)}$	[17, 42]
SCSS	2	$3^k \cdot n^{O(1)}$	[15]	$2-\varepsilon$	$f(k) \cdot n^{O(1)}$	Thm 1.9
$\text{SCSS}_{\text{PLANAR}}$	–	$2^{O(k \log k)} \cdot n^{O(\sqrt{k})}$	[17]	–	$f(k) \cdot n^{o(\sqrt{k})}$	[17]
BI-SCSS	–	$2^{k^2+O(k)} \cdot n^{O(1)}$	Thm 1.10	–	$2^{o(k)} \cdot n^{O(1)}$	Thm 1.11

Table 1: Summary of achievable runtimes for DSN and SCSS when parameterizing by k. A dash (–) refers to computing optimum solutions. Some of the previous results are implicit and in the papers are rather stated for the case when the input graphs are restricted to the same class as the optimum solutions.

which the optimum can be computed efficiently in time $3^{k'} \cdot n^{O(1)}$ using the algorithm of Dreyfus and Wagner [26]. A fundamental observation proved by Borchers and Du [7] is that for any k' there exists a solution to ST of cost at most $1 + \frac{1}{\lfloor \log_2 k' \rfloor}$ times the optimum, in which every full component contains at most k' terminals. Thus setting $k' = 2^{1/\varepsilon}$ for some constant $\varepsilon > 0$, all full-components with at most $2^{1/\varepsilon}$ terminals can be computed in polynomial time, and among them exists a collection forming a $(1+\varepsilon)$-approximation. The key to obtain approximation ratios smaller than 2 for ST is to cleverly select a good subset of all computed full-components. This is for instance done in [8] via an iterative rounding procedure, resulting in an approximation ratio of $\ln(4) + \varepsilon < 1.39$, which currently is the best one known.

Our main technical contribution is to generalize the Borchers and Du Theorem to $\text{BI-DSN}_{\text{PLANAR}}$. In particular, to obtain our approximation scheme of Theorem 1.1, we employ a similar approach by decomposing a $\text{BI-DSN}_{\text{PLANAR}}$ solution into sub-instances, each containing a small number of terminals. As $\text{BI-DSN}_{\text{PLANAR}}$ is W[1]-hard by Theorem 1.3, we cannot hope to compute optimum solutions to each sub-instance as efficiently as for ST. However, we provide an XP-algorithm with runtime $2^{O(k^{3/2} \log k)} \cdot n^{O(\sqrt{k})}$ for $\text{BI-DSN}_{\text{PLANAR}}$ in Theorem 1.4. Thus if every sub-instance contains at most $2^{1/\varepsilon}$ terminals, each can be solved in $n^{2^{O(1/\varepsilon)}}$ time, and this accounts for the "non-efficient" runtime of our approximation scheme. Since we allow runtimes parameterized by k, we can then search for a good subset of precomputed small optimum solutions to obtain a solution to the given demand set $\mathcal{D}$. For the latter solution to be a $(1+\varepsilon)$-approximation however, we need to generalize the Borchers and Du Theorem for ST to $\text{BI-DSN}_{\text{PLANAR}}$ (see Theorem 4.1 for the formal statement). This constitutes the bulk of the work to prove Theorem 1.1.

Exact algorithms for $\text{BI-DSN}_{\text{PLANAR}}$ and BI-SCSS. Also from a parameterized point of view, understanding the structure of the optimum solution to DSN has lead to useful insights in

the past. We will leverage one such recent result by Feldmann and Marx [34], where the above mentioned standard special case of restricting the patterns of the demands in $\mathcal{D}$ is studied in depth. The result is a complete dichotomy over which classes of restricted patterns define special cases of DSN that are FPT and which are W[1]-hard for parameter k. The high-level idea is that whenever the demand patterns imply optimum solutions of constant treewidth, there is an FPT algorithm computing such an optimum. In contrast, the problem is W[1]-hard whenever the demand patterns imply the existence of optimum solutions of arbitrarily large treewidth. The FPT algorithm from [34] lies at the heart of all our positive results, and therefore shows that the techniques developed in [34] to optimally solve special cases of DSN can be extended to find (near-)optimum solutions for other W[1]-hard special cases as well. It is important to note that the algorithm of [34] can also be used to compute the cheapest solution of treewidth at most ω, even if there is an even better solution of treewidth larger than ω (which might be hard to compute). Formally, the result leveraged in this paper is the following.

Theorem 1.12 (implicit in Theorem 5 of [34])**.** *If $\mathcal{K}$ is the class of graphs with treewidth at most ω, then the* DSN$_{\mathcal{K}}$ *problem can be solved in $2^{O(k\omega \log \omega)} \cdot n^{O(\omega)}$ time.*

We exploit the algorithm given by Theorem 1.12 to prove our algorithmic results of Theorem 1.4 and Theorem 1.10. In particular, we prove that any bi-DSN$_{\text{Planar}}$ solution has treewidth $O(\sqrt{k})$, from which Theorem 1.4 follows immediately. For bi-SCSS however, we give an example of an optimum solution of treewidth $\Omega(k)$. Hence we cannot exploit the algorithm of Theorem 1.12 directly to obtain Theorem 1.10. In fact on general input graphs, a treewidth of $\Omega(k)$ would imply that the problem is W[1]-hard by the hardness results in [34] (which was indeed originally shown by Guo et al. [42]). As this stands in stark contrast to Theorem 1.10, it is particularly interesting that the SCSS problem on bidirected input graphs is FPT. We prove this result by decomposing an optimum solution to bi-SCSS into sub-instances of bi-SCSS$_{\mathcal{K}}$, where $\mathcal{K}$ is a class of directed graphs of treewidth 1 (so-called *poly-trees*). For each such sub-instance we can compute a solution in $2^{O(k)} \cdot n^{O(1)}$ time by using Theorem 1.12 (for $\omega = 1$), and then combine them into an optimum solution to bi-SCSS.

W[1]-hardness and runtime lower bounds. Our hardness proofs for bi-DSN are based on reductions from the Grid Tiling problem [19]. This problem is particularly well-suited to prove hardness for problems on planar graphs, due to its grid-like structure. We first develop a specific gadget that can be exploited to show hardness for bidirected graphs. This gadget however is not planar. We only exploit the structure of Grid Tiling to show that the optimum solution is planar for Theorem 1.3. For Theorem 1.7 we modify this reduction to obtain a stronger runtime lower bound, but in the process we lose the property that the optimum is planar.

Parameterized inapproximability. Our hardness result for SCSS is proved by combining a variant of a known reduction by Guo et al. [42] with a recent parameterized hardness of approximation result for Densest k-Subgraph [9]. Our inapproximability result for bi-DSN is shown by combining our W[1]-hardness reduction with the same hardness of approximation result of Densest k-Subgraph.

1.3 Approximate kernelization

A topic closely related to parameterized algorithms is kernelization, which concerns efficient pre-processing algorithms. As formalized by Lokshtanov et al. [57], an *α-approximate kernel* for an optimization problem consists of a *reduction* and a *lifting* algorithm, both running in polynomial time. The reduction algorithm takes an instance I with parameter k and computes

a new instance I' and parameter k', such that the size $|I'| + k'$ of the new instance is bounded by some function $f(k)$ of the input parameter. This new instance I' is also called a *kernel* of I. The lifting algorithm takes as input any β-approximation for the kernel I' and computes an $\alpha\beta$-approximate solution for I.

It has long been known that a problem is FPT if and only if it admits an *exact* kernel, i.e., a 1-approximate kernel. Lokshtanov et al. [57] prove that this is also the case in general: a problem has a parameterized α-approximation algorithm if and only if it admits an α-approximate kernel. Note that the size of the kernel might in general be very large, even if it is bounded in the input parameter. Therefore a well-studied interesting question is whether a polynomial-sized kernel exists, which can be taken as evidence that a problem admits a very efficient pre-processing algorithm. If a problem admits a polynomial-sized $(1+\varepsilon)$-approximate kernel for every $\varepsilon > 0$, then we say that it admits a *polynomial-sized approximate kernelization scheme (PSAKS)*.

The ST problem is known to be FPT [26], while not admitting any polynomial-sized exact kernel [25] for parameter k, unless NP $\subseteq$ coNP/Poly. However, as noted by Lokshtanov et al. [57], the Borchers and Du Theorem implies the existence of a PSAKS for the ST problem. As a consequence of our generalization of the Borchers and Du Theorem we also obtain a PSAKS for BI-DSN$_{\text{PLANAR}}$ (see Corollary 4.2 for a formal statement). This is despite the fact that this problem does not admit *any* exact kernel for parameter k according to Theorem 1.3. Furthermore, we observe that the same kernel is in fact a polynomial-sized $(2+\varepsilon)$-approximate kernel for the BI-DSN problem. This nicely complements the existence of a parameterized 2-approximation algorithm according to Theorem 1.6.

1.4 Related work

The ST problem is one of the 21 NP-hard problems listed in the seminal paper of Karp [47]. Dreyfus and Wagner [26] showed that the problem is solvable in time $3^k \cdot n^{O(1)}$, which was later improved [39] to $(2+\varepsilon)^k \cdot n^{O(1)}$ for any constant $\varepsilon > 0$. For values $k > 2 \log n \log^3 \log n$ an even faster algorithm exists [77]. For unweighted graphs an algorithm with runtime $2^k \cdot n^{O(1)}$ can be obtained [6, 68]. An early LP-based 2-approximation algorithm for ST uses the so-called *bidirected cut relaxation (BCR)* [28, 35, 79], which formulates the problem by bidirecting the undirected input graph. Thus bidirected instances have implicitly been used even for the classical ST problem since the 1960s. For ST and SF there are PTASs on planar and bounded genus graphs [4, 30].

A recent result [29] investigates the complexity of DSN with respect to the stronger parameter $|R|$ (instead of the number of demands k; see remark above). It is shown that for bounded genus graphs the DSN problem can be solved in $f(|R|) \cdot n^{O(|R|)}$ time, while in general no $f(|R|) \cdot n^{o(|R|^2/\log|R|)}$ time algorithm exists, under ETH. The DST problem has an $O(k^\varepsilon)$-approximation in polynomial time [10], and an $O(\log^2 k/\log\log k)$-approximation in quasi-polynomial time [41]. Moreover, no better approximation is possible in quasi-polynomial time [41]. A long standing open problem is whether a polynomial-time algorithm with poly-logarithmic approximation guarantee exists for DST. The SCSS problem has also been studied in the special case when $R = V$. This case is commonly known as MINIMUM STRONGLY CONNECTED SPANNING SUBGRAPH, and the best approximation factor known is 2, which is also given by computing two spanning arborescences [38], and for $R = V$ can be done in polynomial time. For the unweighted case however, a 3/2-approximation is obtainable [76], which is contrast to unweighted SCSS, where the lower bound of Theorem 1.9 is also valid.

Bidirected input graphs have been studied in the context of radio and ad-hoc wireless networks [13, 54, 73, 78]. In the POWER ASSIGNMENT problem, nodes of a given bidirected network need to be activated in order to induce a network satisfying some connectivity condition.

For instance in [13], the problem of finding a strongly connected network is considered, but also other settings such as 2-(vertex/edge)-connectivity [78] or k-(vertex/edge)-connectivity [54] have been studied.

1.5 Organization of the paper

We give some preliminaries and basic observations on the structure of optimum solutions to BI-DSN in bidirected input graphs in Section 2. These are used throughout Section 4, where we present our approximation scheme for BI-DSN$_{\text{PLANAR}}$ of Theorem 1.1, and Section 5, where we show how to compute optimum solutions to BI-DSN$_{\text{PLANAR}}$ for Theorem 1.4 and BI-SCSS for Theorem 1.10. Before presenting our main result of Section 4 however, we first need to develop the approximation algorithms for BI-DSN of Theorem 1.6, which we do in Section 3 together with the hardness result for BI-DSN$_{\text{PLANAR}}$ of Theorem 1.2. The inapproximability results for BI-DSN of Theorem 1.5 and SCSS of Theorem 1.9 are given in Section 7, and the remaining runtime lower bounds for BI-DSN$_{\text{PLANAR}}$ of Theorem 1.3, BI-DSN of Theorem 1.7, and BI-SCSS of Theorem 1.11 can be found in Section 6. Finally, in Section 9 we list some open questions.

2 Structural properties of optimum solutions to BI-DSN

In this section we give some definitions relevant to directed and bidirected graphs, and some fundamental observations on solutions to BI-DSN that we will use throughout the paper.

Due to the similarity of bidirected graphs to undirected graphs, we will often exploit the structure of the underlying undirected graph of a given bidirected graph. More generally, for any directed graph G we denote the underlying undirected graph by $\overline{G}$. A *poly-graph* is obtained by directing the edges of an undirected graph, and analogously we obtain *poly-cycles*, *poly-paths*, and *poly-trees*. A strongly connected poly-cycle is a *directed cycle*, and a poly-tree for which all vertices can reach (or are reachable from) a designated *root* vertex r is called an *out-arborescence* (or *in-arborescence*). Note that for any edge uv of a poly-graph, the reverse edge vu does not exist, and so a poly-graph is in a sense the opposite of a bidirected graph. In between poly-graphs and bidirected graphs are general directed graphs.

2.1 Cycles of optimum solutions in bidirected graphs

We need the following observation, which has far reaching consequences for BI-DSN algorithms. An optimum BI-DSN solution may contain poly-cycles, which are not directed cycles: consider for instance a bidirected graph for which the underlying undirected graph is a cycle on four vertices and every edge has unit weight. If the vertices are numbered $1, 2, 3, 4$ along the cycle and the demands are $\{(1,2),(1,4),(3,2),(3,4)\}$, then it is not hard to see that an optimum solution is given by the poly-cycle with edges corresponding to the demands, i.e., the edge set $\{(1,2),(1,4),(3,2),(3,4)\}$. As the following lemma shows however, any such poly-cycle can be replaced by a directed cycle.

Lemma 2.1. *Let O be a poly-cycle of a subgraph $N \subseteq G$ in a bidirected graph G. Replacing O with a directed cycle on $V(O)$ in N results in a subgraph M of G with cost at most that of N, such that a $u \to v$ path exists in M for every vertex pair u, v for which N contained a $u \to v$ path.*

Proof. Removing all edges of O in N and replacing them with a directed cycle cannot increase the cost, as G is bidirected (the cost may decrease if an edge uv of O is replaced by an edge vu, which is already contained in N). Any $u \to v$ path that leads through O in N can be rerouted through the strongly connected directed cycle in M. □

From this we can deduce the following useful observation, which we will exploit for all of our algorithms. The intuitive meaning of it is that any poly-cycle of an optimum BI-DSN solution splits the solution into parts of which each contains at least one terminal.

Lemma 2.2. *Let $N \subseteq G$ be an optimum* BI-DSN *solution in a bidirected graph G, such that N contains a poly-cycle $O \subseteq N$. Every edge of N that is incident to two vertices of O is also part of O. Moreover, every connected component of the graph resulting from removing $V(O)$ from N contains at least one terminal.*

Proof. By Lemma 2.1 we may exchange O with a directed cycle O' without increasing the cost and maintaining all connections for the demands given by the BI-DSN instance. Since N has minimum cost, this means that the resulting network N' is also an optimum solution. Assume that N contained some edge e incident to two vertices of O but $e \notin E(O)$. The edge e cannot be a reverse edge of some edge f of O, as we could replace O with a cycle directed in the same direction as e. This would decrease the cost as N' only contains e, while N contains both e and f. We are left with the case that e is a *chord* of O, i.e., it connects two non-adjacent vertices of O. However in this case, the endpoints of e are strongly connected through O' in N' even after removing e. Thus we would be able to safely remove e and decrease the cost of N'.

Now assume that some connected component C of the graph obtained from N by removing $V(O)$ contains no terminal. Note that C also exists in the graph obtained from N' by removing O' and its vertices. As C contains no terminals, any $s \to t$ path in N' for a demand (s,t) that contains a vertex of C must contain a $u \to v$ subpath for some vertices $u, v \in V(O')$ with internal vertices from C. However the vertices u, v are strongly connected through O' and hence the $u \to v$ subpath can be rerouted via O'. This means we may safely remove C from N' without loosing any connections for the required demands. However this contradicts the optimality of N', and in turn also our assumption that N is an optimum solution. □

2.2 Reducing the vertex degrees

For our proofs, it will be convenient to assume that the degrees of the vertices in some given graph are bounded. More specifically, consider a (possibly planar) graph N connecting a terminal set R according to some set of demands. We use the standard procedure below, which assures that every terminal has only one neighbour in N, and every *Steiner vertex* of N, i.e. every non-terminal in $V(N) \setminus R$, has exactly three neighbours in N.

We execute the following steps on N in the given order. It is easy to see that these operations preserve planarity (if N is planar), the cost of N, and the connectivity according to the demands.

1. For every terminal $t \in R$ that has more than one neighbour in N, we introduce a new Steiner vertex v and add the edges vt and tv with cost 0 each. Thereafter every neighbour w of t different from v is made a neighbour of v instead. That is, the edges wt and tw are replaced by the edges wv and vw of the same cost. After this, every terminal in N has one neighbour only.
2. Then for every Steiner vertex v with more than 3 neighbours in N, we split v into two vertices as follows. In case N is planar we first fix a drawing of N. Then we introduce a new Steiner vertex u and edges uv and vu with cost 0 each. As the new vertex only has one neighbour, we may draw u in an arbitrary face of N that is incident to v. Let F be this face containing u, and let w_1 and w_2 be the two neighbours of v incident to F that are different from u. For each $j \in \{1,2\}$, we replace the edges vw_j and w_jv with edges uw_j and w_ju, respectively. We maintain the edge costs in each of these replacement steps. Note that N remains planar under this operation. In case N is not planar, we proceed in the same way but simply pick arbitrary neighbours w_1, w_2 of v that are different from

u. After repeating this for every Steiner vertex with more than 3 neighbours, all Steiner vertices of N have at most three neighbours.

3. Next we consider each Steiner vertex v that has exactly two neighbours u and w. If N contains the path uvw, we add an edge uw with the same cost as the path. Similarly, if N has a wvu path, we introduce the edge wu with the same cost. We then remove the vertex v. After this all Steiner vertices of N have exactly three neighbours.

3 Hardness and algorithms for BI-DSN via undirected graphs

In this section we present two results for problems on bidirected graphs that follow from corresponding results on undirected graphs. We first prove Theorem 1.2, which we restate below. In particular, it implies that BI-DSN$_{\text{PLANAR}}$ has no PTAS, unless P=NP.

Theorem 1.2. *The* BI-DST *problem (and by extension also the* BI-DSN$_{\text{PLANAR}}$ *problem) is* APX-hard.

Proof. Given a STEINER TREE (ST) instance on an undirected graph $\overline{G}$, we simply bidirect each edge to obtain the bidirected graph G. We then choose any of the terminals in G as the root to get an instance of BI-DST. It is easy to see that any solution to ST in $\overline{G}$ corresponds to a solution to BI-DST in G of the same cost, and vice versa. As the ST problem is APX-hard [18], the hardness carries over to BI-DST. □

Note that as the definition of the ST problem does not restrict the feasible solutions to trees, this hardness result does not restrict the approximate solutions to BI-DST to arborescences either. That is, it is also hard to compute an approximation N to the optimum BI-DSN$_{\text{PLANAR}}$ solution, even if we allow N to be a non-planar graph.

Next we turn to the positive result of Theorem 1.6, which we also restate below. Note that this theorem is in contrast to Theorem 1.2 and Theorem 1.5.

Theorem 1.6. *The* BI-DSN *problem admits a* 4*-approximation in polynomial time, and a* 2*-approximation in* $2^{O(k)} \cdot n^{O(1)}$ *time.*

Proof. Given a bidirected graph G and a demand set $\mathcal{D} = \{(s_i, t_i) \mid 1 \leq i \leq k\}$ of an instance to BI-DSN, we reduce it to an instance of the STEINER FOREST (SF) problem in the underlying undirected graph $\overline{G}$ with the corresponding unordered demand set $\overline{\mathcal{D}} = \{\{s_i, t_i\} \mid (s_i, t_i) \in \mathcal{D}\}$. The returned BI-DSN solution is the network $N \subseteq G$ that contains both edges uv and vu for any undirected edge between u and v of the SF solution computed for $\overline{G}$. Thus the cost of N is at most twice the cost of the SF solution. At the same time, the optimum SF solution in $\overline{G}$ has cost at most that of the optimum BI-DSN solution in G, since taking the underlying undirected graph of the latter is an SF solution in $\overline{G}$.

The first part of theorem now follows by using the polynomial time 2-approximation algorithm Agrawal et al. [1] for the SF problem. We now show how to solve the SF problem in $2^{O(k)} \cdot n^{O(1)}$ time using the FPT algorithm of Dreyfus and Wagner [26] for the ST problem which runs in $3^p \cdot n^{O(1)}$ time where p is number of terminals. However, as observed in [34], it can also easily be used for SF as well: the optimum solution to SF is a forest and therefore the terminal set can be partitioned so that each part is a tree in the optimum. We now use dynamic programming: for each $X \subseteq [k]$ let OPT$[X]$ denote the minimum cost solution for the instance of SF with the terminal pairs $\overline{\mathcal{D}}_X := \{\{s_i, t_i\} \mid i \in X\}$. We define OPT$[\emptyset] = 0$. Then, we have the following recurrence

$$\text{OPT}[X] = \min_{Y \subseteq X, Y \neq \emptyset} \Big\{ \text{DF}[Y] + \text{OPT}[X \setminus Y] \Big\}$$

where $\mathrm{DF}[Y]$ is the cost obtained by running the Dreyfus-Wagner algorithm on the instance of ST whose terminal set is Y. The correctness of the recurrence follows since the forest which forms the optimal solution of the instance of SF with terminal pairs $\overline{\mathcal{D}}_X$ must contain some non-empty subset of the terminal pairs in one of its trees. The final answer that we output is $\mathrm{OPT}\big[[k]\big]$. Since the running time of the Dreyfus-Wagner algorithm is $3^p \cdot n^{O(1)}$, the running time of the dynamic program is

$$\sum_{i=1}^{k} \binom{k}{i} \cdot \Big(\sum_{j=1}^{i} 3^{2j} \cdot n^{O(1)} \Big) \leq 9 \cdot n^{O(1)} \cdot \Big(\sum_{i=1}^{k} \binom{k}{i} \cdot 9^i \Big) = 2^{O(k)} \cdot n^{O(1)} \qquad \square$$

In Section 4 we will prove that a PSAKS exists for the BI-DSN$_{\text{PLANAR}}$ problem. One ingredient for this will be the existence of a polynomial time algorithm that gives a good estimate of the cost of a planar optimum solution. The 4-approximation algorithm of Theorem 1.6 provides a good lower bound on the cost of the overall optimum, and thus also for the planar optimum. However, a priori this does not provide a good upper bound, since the planar optimum might cost a lot more than the overall optimum, and thus than the approximation. Note though that both algorithms of Theorem 1.6 compute planar solutions, and hence they also upper bound the optimum planar solution. In particular, the existence of a planar 2-approximation of the optimum solution to BI-DSN implies that the planar optimum cannot cost more than twice the overall optimum, as summarized below.

Corollary 3.1. *For any* BI-DSN *instance with optimum solution N, there exists a planar solution N' such that* $\mathrm{cost}(N') \leq 2\,\mathrm{cost}(N)$.

4 An approximation scheme for BI-DSN$_{\text{PLANAR}}$

In this section we prove Theorem 1.1, which is restated below. Note that since we have k demand pairs, it follows that the number $|R|$ of terminals is at most $2k$, where $R = \bigcup_{i=1}^{k} \{s_i, t_i\}$. Henceforth in this section, we use the upper bound $2k$ on the number of terminals $|R|$ for ease of presentation (when instead we could replace k by $|R|$ in the running time of Theorem 1.1).

Theorem 1.1. *There is a $2^{O(k^2)} n^{2^{O(1/\varepsilon)}}$ time algorithm for* BI-DSN$_{\text{PLANAR}}$*, that for any $\varepsilon > 0$ computes a $(1+\varepsilon)$-approximation.*

The bulk of the proof is captured by the following result, which generalizes the corresponding theorem by Borchers and Du [7] for the ST problem, and which is our main technical contribution. In order to facilitate the definition of a sub-instance to DSN, we encode the demands of a DSN instance using a *pattern graph* H, as also done in [34]: the vertex set of H is the terminal set R, and H contains the directed edge st if and only if (s,t) is a demand. Hence the DSN problem asks for a minimum cost network $N \subseteq G$ having an $s \to t$ path for each edge st of H. Here $\mathrm{cost}(N)$ denotes the cost of a graph (solution) N, i.e., the sum of its edge weights.

Theorem 4.1. *Let G be a bidirected graph, and H a pattern graph on $R \subseteq V(G)$. Let $N \subseteq G$ be the cheapest planar solution to pattern H. For any $\varepsilon > 0$, there exists a set of patterns $\mathcal{H}$ such that*

1. *$V(H') \subseteq R$ with $|V(H')| \leq 2^{1+1/\varepsilon}$ for each $H' \in \mathcal{H}$,*
2. *given any feasible solutions $N_{H'} \subseteq G$ for all $H' \in \mathcal{H}$, the union $\bigcup_{H' \in \mathcal{H}} N_{H'}$ of the these solutions forms a feasible solution to H, and*
3. *there exist feasible planar solutions $N^*_{H'} \subseteq G$ for all $H' \in \mathcal{H}$ such that $\sum_{H' \in \mathcal{H}} \mathrm{cost}(N^*_{H'}) \leq (1+\varepsilon) \cdot \mathrm{cost}(N)$.*

Note that the pattern graphs H' of the set $\mathcal{H}$ in this theorem do not have to be subgraphs of the given pattern H. In fact, as the proof of Theorem 4.1 below shows, in general they are not. Before we give the proof, we describe the consequences of this theorem.

Consequences of Theorem 4.1. Our approximation scheme of Theorem 1.1 will compute optimum planar solutions to all patterns with at most $2^{1+1/\varepsilon}$ terminals using the XP algorithm of Theorem 1.4, and then essentially find the set $\mathcal{H}$ of Theorem 4.1 via a dynamic program. This is captured in the following proof.

Proof of Theorem 1.1. The first step of the algorithm is to consider every possible pattern graph on at most $g(\varepsilon) = 2^{1+1/\varepsilon}$ terminals from R. For each such pattern H' the algorithm computes an optimum BI-DSN$_{\text{PLANAR}}$ solution $N_{H'}$ (if any) using the algorithm of Theorem 1.4. Since any considered pattern graph has at most $2\binom{g(\varepsilon)}{2} < g(\varepsilon)^2$ edges, and (regardless of the input pattern H) there is a total of $2\binom{2k}{2} < 4k^2$ possible demands between the at most $2k$ terminals of R, the total number of considered pattern graphs is less than $\sum_{i=0}^{g(\varepsilon)^2} \binom{4k^2}{i} = k^{2^{O(1/\varepsilon)}}$. An optimum planar solution (if it exists) to each of these patterns is computed in $2^{g(\varepsilon)^{3/2}\log g(\varepsilon)} \cdot n^{O(\sqrt{g(\varepsilon)})} = n^{2^{O(1/\varepsilon)}}$ time via Theorem 1.4. Hence up to now the algorithm takes $n^{2^{O(1/\varepsilon)}}$ time, as $k \in O(n^2)$.

The next step is to use a dynamic program to compute a solution to the input pattern H by putting together these pre-computed planar solutions. More concretely, let $N_1, \ldots, N_p$ be all the solutions computed in the first step (given in any arbitrary order). For any subset $\mathcal{N}$ of these planar graphs, in the following we denote by $\text{cost}(\mathcal{N}) := \sum_{N\in\mathcal{N}} \text{cost}(N)$ their total cost and by $\bigcup\mathcal{N} := \bigcup_{N\in\mathcal{N}} N$ their union. For $1 \le i \le p$ and any pattern graph H', we define

$$\sigma(H', i) = \min\left\{\text{cost}(\mathcal{N}) \;\middle|\; \mathcal{N} \subseteq \{N_1, \ldots, N_i\} \text{ and } \bigcup\mathcal{N} \text{ feasible for } H'\right\} \tag{1}$$

to be the minimum total cost of a subset of the first i planar graphs $N_1, \ldots, N_i$ that forms a feasible solution to H'. Note that the total cost of a set $\mathcal{N}$ counts edges appearing in more than one graphs of $\mathcal{N}$ several times. If no feasible solution to H' can be obtained from any subset of $N_1, \ldots, N_i$, then we define $\sigma(H', i)$ to be ∞.

Let $\mathcal{H}$ be the set of patterns given by Theorem 4.1 for the optimum planar solution N to H, and let $N^*_{H'}$ be the planar solution to each $H' \in \mathcal{H}$ given by the theorem. The existence of $N^*_{H'}$ implies that for each $H' \in \mathcal{H}$ there is also a feasible planar solution $N_{H'}$ among $N_1, \ldots, N_p$, and by Theorem 4.1 their union $\bigcup_{H'\in\mathcal{H}} N_{H'}$ is a feasible solution to H. Thus

$$\sigma(H, p) \le \sum_{H'\in\mathcal{H}} \text{cost}(N_{H'}) \le \sum_{H'\in\mathcal{H}} \text{cost}(N^*_{H'}) = (1+\varepsilon)\,\text{cost}(N),$$

where the second inequality follows since each computed planar solution N_i (and thus each $N_{H'}$ where $H' \in \mathcal{H}$) is an optimum solution due to Theorem 1.4. We conclude that $\sigma(H, p)$ is the value of a $(1+\varepsilon)$-approximation to the optimum BI-DSN$_{\text{PLANAR}}$ solution.

To recursively compute $\sigma(H', i)$ for any pattern graph H' on R and any $1 \le i \le p$, we keep track of the subset $\mathcal{N}^i_{H'} \subseteq \{N_1, \ldots, N_i\}$ of planar graphs that obtain the cost stored by the following dynamic program in $\sigma(H', i)$. For $i = 1$ we just check whether N_1 is a feasible solution to H'. If so, we set $\sigma(H', 1) = \text{cost}(N_1)$ and $\mathcal{N}^1_{H'} = \{N_1\}$, while otherwise we set $\sigma(H', 1) = \infty$ and $\mathcal{N}^1_{H'} = \emptyset$. This obviously computes $\sigma(H', 1)$ correctly. To compute $\sigma(H', i)$ for any $i \ge 2$, we check for every pattern graph H'' whether $(\bigcup\mathcal{N}^{i-1}_{H''}) \cup N_i$ is a feasible solution to H'. Among all such solutions and the graph $\bigcup\mathcal{N}^{i-1}_{H'}$ we store the cost of the cheapest option. More formally,

we claim that for $i \geq 2$

$$\sigma(H', i) = \min \Big\{ \sigma(H', i-1), \sigma(H'', i-1) + \text{cost}(N_i) \;\Big|\; H'' \text{ is a pattern with } \Big(\bigcup \mathcal{N}^{i-1}_{H''}\Big) \cup N_i \text{ feasible for } H' \Big\}. \quad (2)$$

If the right-hand side of (2) is some finite value, we set $\mathcal{N}^i_{H'}$ to the subset obtaining the minimum (i.e., either $\mathcal{N}^{i-1}_{H'}$ or $\mathcal{N}^{i-1}_{H''} \cup \{N_i\}$ for some H''). Otherwise, we let $\mathcal{N}^i_{H'} = \emptyset$.

To show that the recursion given by (2) is correct, fix H' and $i \geq 2$, and let $\mathcal{N}^* \subseteq \{N_1, \ldots, N_i\}$ be the subset of planar graphs defining $\sigma(H', i)$, i.e., $\mathcal{N}^*$ minimizes the right-hand side of (1). We need to show that $\text{cost}(\mathcal{N}^i_{H'}) = \text{cost}(\mathcal{N}^*)$. First note that by (2), $\bigcup \mathcal{N}^i_{H'}$ is a feasible solution to H' and is the union of some subset of $N_1, \ldots, N_i$, so that $\text{cost}(\mathcal{N}^i_{H'}) \geq \text{cost}(\mathcal{N}^*)$ by definition of $\mathcal{N}^*$. In case $N_i \notin \mathcal{N}^*$, by induction we have $\text{cost}(\mathcal{N}^{i-1}_{H'}) = \text{cost}(\mathcal{N}^*)$, and so $\text{cost}(\mathcal{N}^i_{H'}) \leq \text{cost}(\mathcal{N}^*)$, since $\sigma(H', i-1) = \text{cost}(\mathcal{N}^{i-1}_{H'})$ is considered as one of the values over which (2) minimizes. In the other case when $N_i \in \mathcal{N}^*$, consider the graph $\bigcup(\mathcal{N}^* \setminus \{N_i\})$ obtained by taking the union of all planar graphs in $\mathcal{N}^*$ except N_i (note that it may still contain edges of N_i). Now let H'' be the pattern graph on R, which contains an edge st if and only if $\bigcup(\mathcal{N}^* \setminus \{N_i\})$ contains an $s \to t$ path. By induction we have $\text{cost}(\mathcal{N}^{i-1}_{H''}) \leq \text{cost}(\mathcal{N}^* \setminus \{N_i\})$, and adding $\text{cost}(N_i)$ to both sides of this inequality we get $\text{cost}(\mathcal{N}^{i-1}_{H''}) + \text{cost}(N_i) \leq \text{cost}(\mathcal{N}^*)$, since $\mathcal{N}^*$ contains N_i. Moreover, $(\bigcup \mathcal{N}^{i-1}_{H''}) \cup N_i$ is a feasible solution to H', since $\bigcup \mathcal{N}^{i-1}_{H''}$ is a feasible solution to H'' and adding N_i we obtain an $s \to t$ path between terminals $s, t \in R$ if and only if $\bigcup \mathcal{N}^*$ contains some $s \to t$ path as well. Hence $\text{cost}(\mathcal{N}^i_{H'}) \leq \text{cost}(\mathcal{N}^{i-1}_{H''}) + \text{cost}(N_i)$, as the latter term is equal to $\sigma(H'', i-1) + \text{cost}(N_i)$ and is considered as one of the values over which (2) minimizes. In conclusion, also if $N_i \in \mathcal{N}^*$ we have $\text{cost}(\mathcal{N}^i_{H'}) \leq \text{cost}(\mathcal{N}^*)$ and so $\text{cost}(\mathcal{N}^i_{H'}) = \text{cost}(\mathcal{N}^*)$. Thus the recursion given in (2) correctly computes the value of $\sigma(H', i)$ according to its definition in (1).

To bound the runtime of the dynamic program, recall that there are $2\binom{2k}{2} < 4k^2$ possible demands between the at most $2k$ terminals of R. Hence the number of considered pattern graphs H' is less than 2^{4k^2}. Recall also that the first step of the algorithm computes (at most) one planar solution to each pattern on at most $g(\varepsilon)$ terminals of which there are $n^{2^{O(1/\varepsilon)}}$ as $k \in O(n^2)$. Thus the asymptotic size of the table given by all entries $\sigma(H', i)$ (with $1 \leq i \leq p \leq n^{2^{O(1/\varepsilon)}}$) is $2^{O(k^2)} n^{2^{O(1/\varepsilon)}}$. To compute one entry of the table via (2), we need to consider every pattern H'' for each of which we perform a feasibility check, which can be done in polynomial time. Thus the runtime per entry is $2^{O(k^2)} \cdot n^{O(1)}$, and the total runtime of the algorithm (including the first step) is bounded by $2^{O(k^2)} n^{2^{O(1/\varepsilon)}}$. □

Note that even though the output of the algorithm is a $(1+\varepsilon)$-approximation to the cheapest planar solution, the computed solution may not be planar if the input graph is not. Theorem 4.1 shows though that the Borchers and Du Theorem can be extended to a much more general case, while the inapproximability results of Theorem 1.5 for bi-DSN and of [14] for $\text{DSN}_{\text{Planar}}$ show that no further generalizations in this direction are possible. Another consequence of the Borchers and Du Theorem for the ST problem is the existence of a PSAKS for ST, as recently shown by Lokshtanov et al. [57]. By similar arguments this is also true for bi-$\text{DSN}_{\text{Planar}}$, due to Theorem 4.1. A simple observation is that the same kernel is also a $(2+\varepsilon)$-approximate kernel for bi-DSN due to Corollary 3.1, which complements the parameterized 2-approximation of Theorem 1.6. We defer the proof of the following corollary to the end of this section, since we will utilize some of the insights gained to prove Theorem 4.1 (in particular those from Lemma 4.3 below).

Corollary 4.2. *The* BI-DSN$_{\text{PLANAR}}$ *problem admits a polynomial-size approximate kernelization scheme (PSAKS) of size* $k^{2^{O(1/\varepsilon)}}$. *The same kernel is also a polynomial-sized* $(2+\varepsilon)$*-approximate kernel for* BI-DSN.

Proving Theorem 4.1. We first use the transformations of Section 2.2 on the cheapest planar solution $N \subseteq G$, so that each terminal has only 1 neighbour, and each Steiner vertex has exactly 3 neighbours. Furthermore, let G_N be the graph spanned by the edge set $\{uv, vu \mid uv \in E(N)\}$, i.e., it is the underlying bidirected graph of N after performing the transformations of Section 2.2 on N. In particular, also in G_N each terminal has only 1 neighbour, and each Steiner vertex has exactly 3 neighbours. It is not hard to see that proving Theorem 4.1 for the solution N in G_N implies the same result for the original solution in G, by reversing all transformations given in Section 2.2.

The proof consists of two parts, of which the first exploits the bidirectedness of G_N, while the second exploits the planarity of N. The first part will identify paths connecting each Steiner vertex to some terminal in such a way that the paths do not overlap much. This will enable us to select a subset of these paths in the second part, so that the total weight of the selected paths is an ε-fraction of the cost of the solution N. This subset of paths will be used to connect terminals to the boundary vertices of small regions into which we divide N. These regions extended by the paths then form solutions to sub-instances, which together have a cost of $1+\varepsilon$ times the optimum. The first part is captured by the next lemma.

Lemma 4.3. *Let* $N \subseteq G_N$ *be the cheapest planar solution to a pattern graph* H *on* $R \subseteq V(G_N)$. *For every Steiner vertex* $v \in V(N) \setminus R$ *of* N *there is a path* P_v *in* G_N, *such that* P_v *is a* $v \to t$ *path to some terminal* $t \in R$, *and the total cost* $\sum_{v \in V(N) \setminus R} \text{cost}(P_v)$ *of these paths is* $O(\text{cost}(N))$.

For the second part we give each vertex v of N a weight $c(v)$, which is zero for terminals and equal to $\text{cost}(P_v)$ for each Steiner vertex $v \in V(N) \setminus R$ and corresponding path P_v given by Lemma 4.3. We now divide the optimum solution N into regions of small size, such that the boundaries of the regions have small total weight.

Definition 4.4. A *region* is a subgraph of N, and given a set of regions, a *boundary vertex* is a vertex that lies in at least two regions. Given a planar graph N and a value r, a *weighted weak r-division* is a set of regions of N inducing a partition on the edges of N, such that each region has at most r vertices, and the total weight of all boundary vertices is an $O(1/\log r)$-fraction of the total weight $\sum_{v \in V(N)} c(v)$.

Unweighted weak r-divisions of planar graphs have found many applications in approximation algorithms, and are for instance defined in [46]. For these, the total number of boundary vertices is at most $O(n/\sqrt{r})$ (they are called "weak" since they do not bound the boundary vertices of each region individually). Additionally, the number of regions is bounded by $O(n/r)$ in this case [38, 46]. To prove the existence of such weak r-divisions for planar graphs, a separator theorem is applied recursively until each resulting region is small enough. The bound on the number of boundary vertices follows from the well-known fact that any planar graph has a small separator of size $O(\sqrt{n})$.

We however need to bound the total weight of the boundary vertices to obtain weighted weak r-divisions. Unfortunately, separator theorems are not helpful here, since they only bound the number of vertices in the separator but cannot bound their weight. Instead we leverage techniques developed for the Klein-Plotkin-Rao (KPR) Theorem [31, 50, 55]. Even though the obtained $O(1/\log r)$-fraction for the weighted case is exponentially worse than the $O(1/\sqrt{r})$-fraction for unweighted graphs obtained in [38, 46], it follows from a lower bound result of Borchers and Du [7] that for weighted graphs this is best possible, even if the graph is a tree. In contrast to

the unweighted case, we also do not guarantee any bound on the number of regions, and we do not need such a bound either. Our proof follows the outlines of the proof given by Lee [55] for the KPR Theorem.

Lemma 4.5. *Let N be a directed planar graph for which each vertex has at most 3 neighbours, and let each vertex v of N have a weight $c(v) \in \mathbb{R}$. For any $r \in \mathbb{N}$ there is a weighted weak r-division.*

We fist show how to put Lemma 4.3 and Lemma 4.5 together in order to prove Theorem 4.1, before proving the lemmas.

Proof of Theorem 4.1. To identify the pattern set $\mathcal{H}$, we first construct a graph $N_E \subseteq G_N$ from every edge set $E \in \mathcal{E}$ given by Lemma 4.5 and the paths given by Lemma 4.3, after which we extract a pattern from it. Recall that if v is a Steiner vertex then we set the weight $c(v)$ to $\text{cost}(P_v)$, i.e., the path costs of the paths P_v of Lemma 4.3, and otherwise we set the weight to 0. We let $r = 2^{1/\varepsilon}$ in Lemma 4.5, so that each region has at most $2^{1/\varepsilon}$ vertices and the total weight of the boundary vertices is an $O(\varepsilon)$-fraction of the total weight.

We first include the graph spanned by E in N_E. For every Steiner vertex v that is a boundary vertex of the r-division $\mathcal{E}$ and is incident to some edge of E, we also include the $v \to t$ path P_v given by Lemma 4.3 in N_E. As G_N is bidirected, the reverse $t \to v$ path of P_v also exists in G_N, and we include this path in N_E as well. Let H_E be the pattern that has the terminal set of N_E as its vertices, and an edge st if and only if there is an $s \to t$ path in N_E. The pattern set $\mathcal{H}$ contains all patterns H_E constructed in this way for the edge sets $E \in \mathcal{E}$. We need to show (1) that each pattern H_E contains a bounded number of terminals, (2) that the union of any solutions to these patterns is feasible for the input pattern H, and (3) that there are solutions to the patterns with total cost at most $(1 + O(\varepsilon)) \cdot \text{cost}(N)$. Making ε sufficiently small, this implies Theorem 4.1.

For the first part, the bound on the terminals in a pattern H_E follows from the bound on the vertices spanned by the edges of E, as given in Lemma 4.5: the graph N_E contains all terminals spanned by the edges of E, and one terminal for each boundary vertex that is a Steiner vertex spanned by E. Thus the total number of terminals of N_E, and therefore also of H_E, is at most $2r = 2^{1+1/\varepsilon}$.

For the second part, consider any solutions N'_E to the patterns $H_E \in \mathcal{H}$. We need to show that for every edge st of H there is an $s \to t$ path in the union $\bigcup_{H_E \in \mathcal{H}} N'_E$. As N is a feasible solution to H, it contains an $s \to t$ path $P \subseteq N$. Consider the sequence $P_1, P_2, \ldots, P_\ell$ of subpaths of P, such that the edges of each subpath belong to the same edge set of $\mathcal{E}$ and the subpaths are of maximal length under this condition. We construct a sequence $t_0, t_1, \ldots, t_\ell$ of terminals from these subpaths as follows. As it has maximal length, the endpoints of each subpath P_i is either a Steiner vertex that is also a boundary vertex of $\mathcal{E}$, or a terminal (e.g. s and t). First we set $t_0 = s$. For any $i \geq 1$, let $E \in \mathcal{E}$ be the set that contains the edges of P_i. If the last vertex of P_i is a terminal, then t_i is that terminal, while if the last vertex is a Steiner vertex v, then t_i is the terminal that the path P_v included in N_E connects to. If the first vertex of P_i is a terminal, then clearly it is equal to t_{i-1}. Moreover, if the first vertex of P_i is a Steiner vertex v, then by construction the graph N_E contains the reverse $t_{i-1} \to v$ path of P_v. Thus N_E contains a $t_{i-1} \to t_i$ path, and so the pattern H_E contains the edge $t_{i-1}t_i$. This implies that also any arbitrary solution N'_E to H_E contains a $t_{i-1} \to t_i$ path, and therefore the union $\bigcup_{E \in \mathcal{E}} N'_E$ of arbitrary solutions contains a $t_0 \to t_\ell$ path via the intermediate terminals t_i where $i \in \{1, \ldots, \ell - 1\}$. As $t_0 = s$ and $t_\ell = t$, this means that the union is feasible for H.

For the third part we just bound $\sum_{E \in \mathcal{E}} \text{cost}(N_E)$, i.e., the special solutions $N^*_{H'}$ of the theorem statement are exactly the solutions N_E constructed above, which are subgraphs of the planar

graph G_N and thus are planar as well. Note that the cost of each N_E is the cost of the edge set E plus the cost of the paths P_v and their reversed paths attached to the boundary Steiner vertices v incident to E. The sum of the costs of all edge sets $E \in \mathcal{E}$ contribute exactly the cost of N to $\sum_{E\in\mathcal{E}} \text{cost}(N_E)$, since $\mathcal{E}$ is a partition of the edges of N. As we assume that each boundary vertex v of $\mathcal{E}$ has at most three neighbours, v is incident to a constant number of edge sets of $\mathcal{E}$. Thus $\sum_{E\in\mathcal{E}} \text{cost}(N_E)$ also contains the cost of path P_v only a constant number times: twice for each set $E \in \mathcal{E}$ incident to boundary vertex v, due to P_v and its reverse path, which in a bidirected instance has the same cost as P_v. By Lemma 4.5, $\sum_{v\in B} c(v) \leq O(\varepsilon) \cdot \sum_{v\in V(N)} c(v)$, where B is the set of boundary vertices of $\mathcal{E}$, and the cost $c(v)$ of a vertex is the cost of the path P_v if v is a Steiner vertex, and 0 otherwise. Hence all paths P_v and their reverse paths contained in all the graphs N_E for $E \in \mathcal{E}$ contribute at most $O(\varepsilon) \cdot \sum_{v\in V(N)} c(v) = O(\varepsilon) \cdot \sum_v \text{cost}(P_v)$ to $\sum_{E\in\mathcal{E}} \text{cost}(N_E)$. By Lemma 4.3, $\sum_v \text{cost}(P_v) = O(\text{cost}(N))$, and so $\sum_{E\in\mathcal{E}} \text{cost}(N_E) = (1 + O(\varepsilon)) \cdot \text{cost}(N)$. □

We now turn to proving the two remaining lemmas, starting with finding paths for Steiner vertices for Lemma 4.3.

Proof of Lemma 4.3. We begin by analysing the structure of optimal DSN solutions in bidirected graphs, based on Lemma 2.1. Here a *condensation graph* of a directed graph results from contracting each strongly connected component, which hence is a DAG.

Claim 4.6. *For any solution $N \subseteq G_N$ to a pattern H, there is a solution $M \subseteq G_N$ to H with $\text{cost}(M) \leq \text{cost}(N)$ and $\overline{N} = \overline{M}$, such that the condensation graph of M is a poly-forest.*

Proof. Let C be a subgraph of N, which induces a maximal 2-connected component in $\overline{N}$. Assume there is a vertex pair $u, v \in V(C)$ for which no $u \to v$ path exists in C. As C induces a 2-connected component in $\overline{N}$, by Menger's Theorem [20] there are two internally disjoint poly-paths P and Q between u and v in C, which together form a poly-cycle O. By Lemma 2.1 we may replace O by a directed cycle without increasing the cost, and so that there is a directed path for every pair of vertices for which such a path existed before. Additionally, this step introduces a $u \to v$ path along this new directed cycle in C. Repeating this for any pair of vertices for which no directed path exists in C will eventually result in a strongly connected component. Hence we can make every component of N, which induces a maximal 2-connected component in $\overline{N}$, strongly connected without increasing the cost. Note also that $\overline{N}$ does not change.

After this procedure we obtain the graph $M \subseteq G_N$. The maximal 2-connected components in $\overline{M}$ induce subgraphs of the strongly connected components of M (they may be subgraphs due to cycles of length 2 in M, which are bridges in $\overline{M}$). Contracting all strongly connected components of M must therefore result in a poly-forest, as any poly-cycle in the condensation graph would also induce a cycle in $\overline{M}$. ⌟

By Claim 4.6 we may assume w.l.o.g. that the condensation graph of the optimum solution N is a poly-forest such that every terminal has one neighbour and every Steiner vertex has three neighbours. Consider a weakly connected component C of N, i.e., inducing a connected component of $\overline{N}$. We first extend C to a strongly connected graph C' as follows. Let F be the edges of C that do not lie in a strongly connected component, i.e., they are the edges of the condensation graph of C. Let $\widetilde{F} = \{uv \mid vu \in F\}$ be the set containing the reverse edges of F, and let C' be the strongly connected graph spanned by all edges of C in addition to the edges in $\widetilde{F}$. Note that adding $\widetilde{F}$ to C increases the cost by at most a factor of two as G_N is bidirected, while the number of neighbours of any vertex does not change. We claim that in fact C' is a *minimal* SCSS solution to the terminal set $R_C \subseteq R$ contained in C, that is, removing any edge of C' will disconnect some terminal pair of R_C.

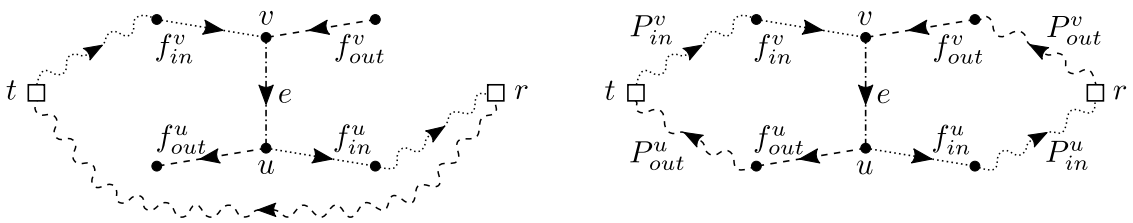

Figure 2: The case when $e = vu$ lies in both A_{in} (dotted) and A_{out} (dashed). Edges are straight lines, paths are wavy and may intersect. On the left: assuming that A_{out} contains a path from the root r to terminal t that does not contain e leads to a contradiction, since f^v_{out} would then be redundant. On the right: otherwise, the paths connecting r and t with v and u imply the existence of a poly-cycle, which due to e has a chord, again leading to a contradiction.

For this, consider any $s \to t$ path of C' containing an edge $e \in \widetilde{F}$ for some terminal pair $s, t \in R_C$. As the edges F of the condensation graph of C form a poly-tree, every path from s to t in C' must pass through e. In particular there is no $s \to t$ path in C, and thus there is no edge st in the pattern graph H. Or equivalently, for any terminal pair $s, t \in R_C$ for which there is a demand $st \in E(H)$, no $s \to t$ path in C' passes through an edge of $\widetilde{F}$. Thus for such a terminal pair the set of paths from s to t is the same in C' and C. Since N is an optimum solution so that every edge e of C is necessary for some pair $s, t \in R_C$ with $st \in E(H)$, the edge e is still necessary in C'. Moreover, for any of the added edges $uv \in \widetilde{F}$ the reverse edge $vu \in F$ was necessary in C to connect some $s \in R_C$ to some $t \in R_C$. As observed above, uv is necessary to connect t to s in C', since the edges F of the condensation graph form a poly-tree.

As C' is a minimal SCSS solution to the terminals R_C contained within, it is the union of an in-arborescence A_{in} and out-arborescence A_{out}, both with the same root $r \in R_C$ and leaf set $R_C \setminus \{r\}$, since every terminal only has one neighbour in G_N. A *branching point* of an arborescence A is a vertex with at least two children in A. We let $W \subseteq V(C')$ be the set consisting of all terminals R_C and all branching points of A_{in} and A_{out}. We will need that any vertex of C' has a vertex of W in its close vicinity. That is, if $N[v] = N(v) \cup \{v\}$ denotes the closed neighbourhood of a vertex v and $N^2[v] = \bigcup_{u \in N[v]} N[v]$, we prove the following.

Claim 4.7. *For every vertex v of C', there is a vertex of W in $N^2[v]$.*

Proof. Assume that $v \notin W$, since otherwise we are done. Such a vertex must be a Steiner vertex, and hence has exactly three neighbours in C'. As v is not a branching point of A_{in} or A_{out}, this means that v is incident to two edges of A_{in} and two edges of A_{out}. This can either mean that there are three edges incident to v of which one lies in both A_{in} and A_{out}, or there are four edges incident to v of which two connect to the same neighbour of v but point in opposite directions. Consider the former case first, i.e., there is one of the edges e incident to v that lies in the intersection of the two arborescences, another incident edge f^v_{in} that lies in A_{in} but not in A_{out}, and a third incident edge f^v_{out} that lies in A_{out} but not in A_{in}. Now assume for a contradiction that the neighbour u of v incident to e also does not belong to W, and w.l.o.g., let $e = vu$ (for the symmetric when $e = uv$, an analogous argument to the following exists). In particular, both f^v_{in} and f^v_{out} are incoming edges to v. By the same observations as for v, there must be an incident edge f^u_{in} to u that lies in A_{in} but not in A_{out}, and an incident edge f^u_{out} that lies in A_{out} but not in A_{in}. Both these edges must be outgoing of u. See Figure 2.

The in-arborescence A_{in} contains a $t \to r$ path from some terminal $t \in R_C$ to the root r passing through e. We claim that A_{out} must contain an $r \to t$ path to the same terminal t passing through e as well. If this were not the case there would be some other $r \to t$ path of

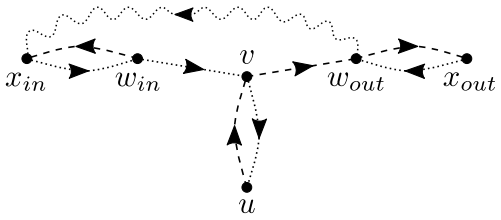

Figure 3: The case when vu lies in A_{in} (dotted) and uv lies in A_{out} (dashed). Assuming that neither w_{in} nor w_{out} has an incident edge lying in both arborescences leads to a contradiction, since vu being a bridge in $\overline{C'}$ implies a path of A_{in} (wavy) connecting w_{out} with x_{in}, but $x_{in}w_{in}$ is also a bridge in $\overline{C'}$.

A_{out} not containing e. Together with the $t \to v$ subpath of the $t \to r$ path in A_{in}, this implies an $r \to v$ path not containing f^v_{out}: the latter edge is not contained in A_{in} and therefore cannot be part of the $t \to v$ subpath. However this means that every terminal reachable from r via v in C' is reachable by a path not containing f^v_{out}. As this edge is not contained in A_{in}, it could safely be removed from C' without disconnecting any terminal pair. This would contradict the minimality of C', which means there must be an $r \to t$ path in A_{out} that passes through v.

For this terminal t, we can conclude that there is a $t \to v$ path $P^v_{in} \subseteq A_{in}$ ending in f^v_{in}, a $u \to r$ path $P^u_{in} \subseteq A_{in}$ starting in f^u_{in}, but also an $r \to v$ path $P^v_{out} \subseteq A_{out}$ ending in f^v_{out}, and a $u \to t$ path $P^u_{out} \subseteq A_{out}$ starting in f^u_{out}. Moreover, none of these four paths contains e. Note that the union $P^v_{in} \cup P^u_{in} \cup P^v_{out} \cup P^u_{out}$ of the four paths contains a poly-cycle O for which e is a *chord*, i.e., it connects two non-adjacent vertices of O.

The strongly connected component C' was constructed from the component C of the optimum solution N by adding the set $\widetilde{F}$ of reverse edges to some existing edge set F of C. Hence, even if O and/or e do not exist in N, there still exists a poly-cycle O' in N with the same vertex set and underlying undirected graph as O, and an edge e' that is a chord to O', which may be e or its reverse edge. This contradicts the optimality of N by Lemma 2.2, and thus $u \in N(v)$ is in W.

It remains to consider the case when v has four incident edges. This means that for one neighbour u of v there are two edges uv and vu in C' of which one belongs to A_{in} and the other to A_{out}. W.l.o.g., let uv belong to A_{out} (for the other case when uv belongs to A_{in}, by symmetry an analogous argument to the following exists). Now let w_{out} and w_{in} be the other two neighbours of v, for which the edge vw_{out} is in A_{out}, while the edge $w_{in}v$ is in A_{in}. If either w_{in} or w_{out} is in W, we are done. Hence assuming that $w_{in}, w_{out} \notin W$, just as v, both w_{in} and w_{out} are Steiner vertices with three neighbours, each incident to two edges of A_{in} and two edges of A_{out}. If either w_{in} or w_{out} has an incident edge that lies in the intersection of A_{in} and A_{out}, by the same argument as for v above, some vertex of $N(w_{in}) \cup N(w_{out})$ must lie in W. As $N(w_{in}) \cup N(w_{out}) \subseteq N^2[v]$ this would conclude the proof.

Hence assume that neither w_{in} nor w_{out} has an incident edge lying in both arborescences. Thus w_{out} has a neighbour $x_{out} \neq v$ such that $w_{out}x_{out} \in E(A_{out})$ and $x_{out}w_{out} \in E(A_{in})$, and w_{in} has a neighbour $x_{in} \neq v$ such that $x_{in}w_{in} \in E(A_{in})$ and $w_{in}x_{in} \in E(A_{out})$. See Figure 3. Note that $x_{in} \neq x_{out}$ as otherwise A_{in} would have a vertex of out-degree more than one. Moreover, by the following argument, we can conclude that in $\overline{C'}$, all three undirected edges vu, $x_{out}w_{out}$, and $x_{in}w_{in}$ are bridges. Consider any edge e in the component C of the optimum solution N from which C' was constructed. By Lemma 2.2, the reverse edge of e can only exist in C if e does not lie on any poly-cycle. That is, if e and its reverse edge exist in C then the corresponding edge in $\overline{C}$ is a bridge. To obtain C' from C we added $\widetilde{F}$, which contains all reverse edges of the condensation graph of C. From Claim 4.6 we concluded that the condensation graph of C is a

poly-forest. Thus any edge of C' for which the reverse edge exists in C' as well, must correspond to a bridge in $\overline{C'}$, including vu, $x_{out}w_{out}$, and $x_{in}w_{in}$, which all lie in A_{in}. Note also that by the same observations, v, w_{out}, and w_{in} lie in the same 2-connected component of $\overline{C'}$, as the reverse edges of vw_{out} and $w_{in}v$ do not exist in C'.

This means that A_{in} contains a path starting in $x_{out}w_{out}$, which reaches the root of A_{in} by passing through vu, as the latter is a bridge of $\overline{C'}$ while v and w_{out} lie in the same 2-connected component of $\overline{C'}$. Since neither v nor w_{in} is a branching point of A_{in} while $x_{in}w_{in}, w_{in}v, vu \in E(A_{in})$, this path of A_{in} contains the subpath given by the sequence $x_{in}w_{in}vu$. But this means that there is a path from $x_{out}w_{out}$ to x_{in} that does not pass through $w_{in}x_{in} \in A_{out}$. This contradicts the fact that $x_{in}w_{in}$ is a bridge of $\overline{C'}$, and thus concludes the proof. ⌟

As the graph G_N is bidirected, for any v-u path P in the underlying undirected graph $\overline{G}_N$ of G_N, there exists a corresponding directed $v \to u$ path in G_N of the same cost. Therefore, we can ignore the directions of the edges in C' and the arborescences A_{out} and A_{in} to identify the paths P_v for Steiner vertices v of N. Thus we will only consider paths in the graphs $\overline{C'}$, $\overline{A}_{out}$, and $\overline{A}_{in}$ from now on. In particular, we exploit the following observation found in [27] (and also used by Borchers and Du [7]) on undirected trees.[4]

Claim 4.8 ([27, Lemma 3.2])**.** *For any undirected tree T we can find a path $P_v \subseteq T$ for every branching point v, such that P_v leads from v to some leaf of T, and all these paths P_v are pairwise edge-disjoint.*

If a Steiner vertex v of C' is a branching point of A_{out} (A_{in}), we let P_v be the corresponding path in $\overline{A}_{out}$ ($\overline{A}_{in}$) given by Claim 4.8 from v to some leaf of A_{out} (A_{in}), which is a terminal. Note that paths in $\overline{A}_{in}$ may overlap with paths in $\overline{A}_{out}$. However any edge in the union of all the paths P_v chosen so far is contained in at most two such paths, one for a branching point of A_{out} and one for a branching point of A_{in}.

It remains to choose a path P_v for every Steiner vertex v that is neither a branching point of A_{out} nor of A_{in}, i.e., for every vertex not in W. By Claim 4.7 for any such vertex $v \notin W$ there is a vertex $u \in N^2[v]$ for which $u \in W$. If u is a terminal, then the path P_v is simply the edge vu if $u \in N(v)$ or the corresponding path vwu for some $w \in N(v)$ otherwise. If u is not a terminal but a branching point of A_{out} or A_{in}, then we chose a path P_u for u above. In this case, P_v is the path contained in the walk given by extending the path P_u by the edge vu or the path vwu, respectively. Note that, as any vertex of C' has at most 3 neighbours, any terminal or branching point $u \in W$ can be used in this way for some vertex $v \notin W$ at most nine times. Therefore any edge in the union of all chosen paths is contained in $O(1)$ paths. Consequently the total cost $\sum_{v \in V(N) \setminus R} \operatorname{cost}(P_v)$ is $O(\operatorname{cost}(C'))$, and as $\operatorname{cost}(C') \leq 2\operatorname{cost}(C)$ we also get $\sum_{v \in V(N) \setminus R} \operatorname{cost}(P_v) = O(\operatorname{cost}(C))$.

We may repeat these arguments for every weakly connected component of N to obtain the lemma. □

Next we give the proof of Lemma 4.5, which shows that there are weighted weak r-divisions for planar graphs.

Proof of Lemma 4.5. We will not be concerned with the edge weights of N and accordingly define the distance function $d_M(u, v)$ for any subgraph M of N to be the *hop-distance* between u and v in $\overline{M}$, i.e., the minimum number of edges on any path from u to v in $\overline{M}$. The idea (as outlined in [31, 55]) is to iteratively "chop" the vertices of N into disjoint sets that induce annuli of bounded thickness measured in the hop-distance, using the following random process.

[4] In [7, 27] the claim is stated for binary trees, but this is an assumption that can be made w.l.o.g. using similar vertex degree transformations as presented in Section 2.2.

For a fixed value τ, if we are given some connected graph M, then we first choose an offset $\tau_0 \in \{1, \ldots, \tau\}$ uniformly at random and an arbitrary vertex v_0 of M. A so-called *τ-chop* then is the partition of the vertices of M defined by the sets

$$A_0 = \{v \in V(M) \mid d_M(v_0, v) < \tau_0\} \text{ and}$$
$$A_i = \{v \in V(M) \mid \tau_0 + (i-1)\tau \leq d_M(v_0, v) < \tau_0 + i\tau\} \text{ for } i \geq 1.$$

We define a τ-chop of a disconnected graph as the partition given by the union of τ-chops on each of the connected components, where for each component we choose an offset τ_0 uniformly at random and an arbitrary vertex v_0. Finally, a τ-chop of a partition $\mathcal{P}$ is the refined partition given by the union of τ-chops on each subgraph induced by a set in $\mathcal{P}$, again choosing a v_0 and a τ_0 for every component of the subgraphs. Hence we may start with N and iteratively perform τ-chops to obtain smaller and smaller subsets of vertices.

Lee [55] now proves the following claim, where the *weak diameter* of a subgraph $M \subseteq N$ is the maximum hop-distance of any two vertices of M measured in the underlying graph N, i.e., $\max_{u,v \in V(M)} d_N(u, v)$. Note that this claim holds independent of the choices of the vertices v_0 and the offsets τ_0.

Claim 4.9 (Lemma 2 in [55])**.** *If N excludes K_h as a minor, then any sequence of $h-1$ iterated τ-chops on N results in a partition $\mathcal{P}$ of $V(N)$, such that each graph induced by a set $S \in \mathcal{P}$ has weak diameter $O(h\tau)$.*

Let $\mathcal{P}$ be the partition of $V(N)$ from Claim 4.9. Since N is planar, it excludes K_5 as a minor, and so the weak diameter of each set $S \in \mathcal{P}$ is $O(\tau)$. We define a partition $\overline{\mathcal{E}}$ of the edges of $\overline{N}$, consisting of sets $E_S \subseteq E(\overline{N})$ for each $S \in \mathcal{P}$. In particular, if S is the set containing the lexicographically smaller vertex incident to an edge e of $\overline{N}$, then e is contained in E_S. Note that the weak diameter of a region M_S spanned by an edge set E_S is at most the weak diameter of the graph induced by S plus 2, i.e., also the weak diameter of M_S is $O(\tau)$. Since $\overline{N}$ has maximum degree 3, the weak diameter bounds the number of vertices in each region by $|V(M_S)| = 2^{O(\tau)}$ for every $S \in \mathcal{P}$. As $\overline{\mathcal{E}}$ corresponds to a partition $\mathcal{E}$ of the edges of N, for some $\tau = \Theta(\log r)$ we obtain an r-division given by $\mathcal{E}$ with the required bound on the sizes of the regions.

It remains to bound the weight of the boundary vertices, for which we bound the expected weight among the random choices of offsets. More concretely, note that when performing a single τ-chop on a connected graph M from a fixed vertex v_0, two adjacent vertices u, v end up in different sets S with probability at most $1/\tau$ by the choice of the offset τ_0 and the definition of the sets A_i, $i \geq 0$. We assign the edge uv of $\overline{N}$ to the set E_S containing the lexicographically smaller vertex among u and v. Thus any vertex w, which has degree at most 3 in $\overline{N}$, is a boundary vertex of a region spanned by some set E_S with probability at most $3/\tau$ when performing a single τ-chop from a fixed vertex v_0. As we perform $h-1$ iterative τ-chops, the expected weight of the boundary vertices is at most $\frac{3(h-1)}{\tau} \sum_{v \in V(N)} c(v)$. Hence, since N is planar and by our choice of $\tau = \Theta(\log r)$, there exists an r-division with only a $O(1/\log r)$-fraction of the total vertex weight in the boundary vertices. □

Proving Corollary 4.2. Finally, we can also prove that Theorem 4.1 implies a PSAKS for BI-DSN$_{\text{PLANAR}}$, by utilizing some of the insights of the above proofs. The proof essentially follows the same lines as the one given for the ST problem by Lokshtanov et al. [57] based on the Borchers and Du Theorem.

Proof of Corollary 4.2. To obtain a polynomial-sized $(1+\varepsilon)$-approximate kernel we proceed similar to the algorithm described at the beginning of this section, by first computing an optimum solution for every possible pattern graph on at most $g(\varepsilon) = 2^{1+1/\varepsilon}$ terminals from R. Using the

XP algorithm of Theorem 1.4, this takes $n^{2^{O(1/\varepsilon)}}$ time, as determined before. If ε is a constant, this amounts to a polynomial runtime. Taking the union of all precomputed solutions gives a graph, which due to Theorem 4.1 contains a $(1+\varepsilon)$-approximation to the optimum of the input graph G (by the same arguments showing that the algorithm computes a $(1+\varepsilon)$-approximation). However the union is not a kernel, since its size is not necessarily bounded as a function of the parameter k. In particular, it may contain many vertices and the edge weights might be large.

To reduce the number of vertices, we apply the vertex degree transformations from Section 2.2 to each computed optimum solution $N_{H'}$ to patterns H' on at most $g(\varepsilon)$ terminals. In particular, every Steiner vertex of $N_{H'}$ now has exactly three neighbours. We use the insights from the proof of Lemma 4.3 to argue that $N_{H'}$ has a bounded number of vertices. By Claim 4.6 we may assume that $N_{H'}$ is an optimum solution to H' for which the condensation graph is a poly-forest. Now consider a weakly connected component C of $N_{H'}$. As argued in the proof of Lemma 4.3, if we add to C the edge set $\widetilde{F}$, which contains the reverse edges to those of the condensation graph of C, then we obtain a minimal SCSS solution C' for the terminal set $R_C \subseteq R$ contained in C. This means that C' is the union of an in-arborescence A_{in} and an out-arborescence A_{out}, both rooted at some terminal $r \in R_C$ and with leaves from R_C. The number of branching points of each of these arborescences is at most $|R_C|$. Hence the set W (as defined earlier) of branching points and terminals R_C contains at most $3|R_C|$ vertices. Due to Claim 4.7 we can map any vertex not contained in W to a vertex in W at hop-distance at most 2. Since every vertex not in W is a Steiner vertex and has three neighbours, at most 9 vertices map to any particular vertex of W. Thus the number of vertices of C not in W is at most $27|R_C|$, which brings the total to at most $30|R_C|$ after adding W. This means that the number of vertices of $N_{H'}$ is at most $30g(\varepsilon) = 2^{O(1/\varepsilon)}$. As calculated earlier, the total number of pattern graphs H' on at most $g(\varepsilon)$ terminals is $k^{2^{O(1/\varepsilon)}}$. Hence taking the union of all computed solutions $N_{H'}$ after applying the vertex degree transformations of Section 2.2 gives a graph G' with $2^{O(1/\varepsilon)} \cdot k^{2^{O(1/\varepsilon)}} = k^{2^{O(1/\varepsilon)}}$ vertices, which is polynomial if ε is constant.

For the edge weights, Lokshtanov et al. [57] show how to round them in such a way that each edge weight can be stored using $O(\log(k/\varepsilon))$ bits for the ST problem. Here we will need slightly more bits. As an ingredient we use that a polynomial time constant approximation algorithm exists, which is provided by Theorem 1.6. In particular, let $M \subseteq G$ be a 4-approximate solution computed by this algorithm for the input graph G. If the weight of an edge e of the union graph G' currently is $w(e)$, then we define a rounded integer weight

$$\hat{w}(e) = \left\lfloor \frac{k^{g(\varepsilon)} w(e)}{\varepsilon \operatorname{cost}(M)} \right\rfloor,$$

and set the edge weights of the union graph G' to $\hat{w}(e)$ instead. By Corollary 3.1 we have $\operatorname{cost}(N) \leq 2\operatorname{cost}(M)$ for the optimum planar solution N to the input instance, and so we may remove any edge of cost more than $2\operatorname{cost}(M)$. This implies that $\hat{w}(e) \leq 2k^{g(\varepsilon)}/\varepsilon$, so that each edge can be encoded using $2^{O(1/\varepsilon)} \log k$ bits. As the graph G' has $k^{2^{O(1/\varepsilon)}}$ vertices, its number of edges is asymptotically the same, and so the size of the kernel including the edge weights also is $k^{2^{O(1/\varepsilon)}}$.

It remains to show that rounding the edge weights does not distort the solution costs by too much. Let $N' \subseteq G'$ be a β-approximation of the optimum planar solution in the kernel, i.e., using weights $\hat{w}(e)$. Let also $N^\star$ be the optimum planar solution in G' when using the original weights $w(e)$. In particular, we get $\sum_{e \in E(N')} \hat{w}(e) \leq \beta \sum_{e \in E(N^\star)} \hat{w}(e)$, since the optimum planar solution in the kernel has cost at most that of $N^\star$ according to weights $\hat{w}(e)$. Since the number of edges of G' is $k^{2^{O(1/\varepsilon)}}$, also N' has at most this many edges, and the cost of this solution

measured by the original edge weights $w(e)$ compared to $N^\star$ and M is

$$\begin{aligned}\sum_{e\in E(N')} w(e) &\leq \sum_{e\in E(N')} \left(\frac{\varepsilon\,\mathrm{cost}(M)}{k^{2^{O(1/\varepsilon)}}}(1+\hat{w}(e))\right)\\ &\leq \varepsilon\,\mathrm{cost}(M) + \frac{\varepsilon\,\mathrm{cost}(M)}{k^{2^{O(1/\varepsilon)}}} \sum_{e\in E(N')} \hat{w}(e)\\ &\leq \varepsilon\,\mathrm{cost}(M) + \frac{\varepsilon\,\mathrm{cost}(M)}{k^{2^{O(1/\varepsilon)}}} \cdot \beta \sum_{e\in E(N^\star)} \hat{w}(e)\\ &\leq \varepsilon\,\mathrm{cost}(M) + \frac{\varepsilon\,\mathrm{cost}(M)}{k^{2^{O(1/\varepsilon)}}} \cdot \beta \sum_{e\in E(N^\star)} \frac{k^{2^{O(1/\varepsilon)}} w(e)}{\varepsilon\,\mathrm{cost}(M)}\\ &= \varepsilon\,\mathrm{cost}(M) + \beta \sum_{e\in E(N^\star)} w(e).\end{aligned}$$

Each of N' and $N^\star$ can clearly be lifted to a solution in the input instance with the same or lower cost (when using weights $w(e)$) in polynomial time. By Theorem 4.1, we have $\mathrm{cost}(N^\star) \leq (1+\varepsilon)\,\mathrm{cost}(N)$ for the optimum planar solution N of the input instance. At the same time, M is a 4-approximation for the input instance, which means that $\mathrm{cost}(M) \leq 4\,\mathrm{cost}(N)$. By the above calculations we hence get that $\mathrm{cost}(N') \leq (1+5\varepsilon)\beta\,\mathrm{cost}(N)$. By making ε sufficiently small, this implies the desired approximation bound for BI-DSN$_{\text{PLANAR}}$. Moreover, due to Corollary 3.1, the planar solution N' is also a $2(1+5\varepsilon)\beta$-approximation of the overall optimum of the input instance, and thus the claimed approximation bound for BI-DSN follows as well. □

5 Computing optimum solutions in bidirected graphs

In this section we show how to compute optimum solutions to BI-SCSS and to BI-DSN$_{\text{PLANAR}}$, and we start with the latter.

5.1 An XP algorithm for BI-DSN$_{\text{PLANAR}}$

In this section we prove Theorem 1.4, which is restated below.

Theorem 1.4. *There is a $2^{O(k^{3/2}\log k)} \cdot n^{O(\sqrt{k})}$ time algorithm to compute the optimum solution for* BI-DSN$_{\text{PLANAR}}$*, i.e., a solution with cost at most that of the cheapest planar one.*

Proof. The proof hinges on the fact that an optimum solution $N \subseteq G$ to BI-DSN$_{\text{PLANAR}}$ has treewidth less than $6\sqrt{k}$. To show this, assume to the contrary that the treewidth of N is at least $6\sqrt{k}$. It is well-known that this implies that N contains a $6\sqrt{k} \times 6\sqrt{k}$ grid minor. Consider a planar drawing of N. Since there are only k terminal pairs, we can have at most $2k$ terminals. By the pigeon-hole principle, the grid minor contains some 3×3 grid minor M for which no terminal touches any of the faces in the interior of M in the drawing. We can see M as consisting of a poly-cycle O with all other vertices of M touching faces in the interior of O in the drawing. In particular, removing O from M will leave a non-empty connected component (the interior of O) which contains no terminals. This however contradicts Lemma 2.2. Note that such a solution would also be planar, and thus the treewidth of the optimum planar solution is $O(\sqrt{k})$.

Since by Theorem 1.12 there is an algorithm to compute the optimum among all solution of treewidth at most ω in time $2^{O(k\omega\log\omega)} \cdot n^{O(\omega)}$, the above treewidth bound implies Theorem 1.4. □

Note that the algorithm in Theorem 1.4 does not necessarily compute a planar solution, if the input is not planar, but the found solution will still have cost at most that of the cheapest planar solution. Theorem 1.3 shows that the running time obtained in Theorem 1.4 for BI-DSN$_{\text{PLANAR}}$ is asymptotically optimal under ETH.

5.2 FPT algorithm for BI-SCSS

We now turn to BI-SCSS (without restricting the optimum) and show that this problem is FPT for parameter k (recall that for SCSS the number of demands equals the number of terminals). The formal theorem is restated below:

Theorem 1.10. *There is a $2^{k^2+O(k)} \cdot n^{O(1)}$ time algorithm for* BI-SCSS*, i.e., it is FPT for parameter k.*

An optimum solution to BI-SCSS can have treewidth $\Omega(k)$, as the following lemma shows. This is particularly interesting, since the results in [34] show that any problem with optima of unbounded treewidth on general input graphs is W[1]-hard. Note that no solution with larger treewidth can exist, as by [34] any optimum solution to DSN has treewidth $O(k)$.

Lemma 5.1. *There are instances of* BI-SCSS *in which the optimum solution has treewidth $\Omega(k)$.*

Proof. We will describe the underlying undirected graph $\overline{G}$ of an instance G to BI-SCSS. We begin with a constant degree expander graph with k vertices, for which we subdivide each edge twice. The resulting graph is going to be $\overline{G}$, where each edge has unit weight. All vertices of the graph are going to be terminals, which means that the number of terminals is $\Theta(k)$, since the number of edges in a constant degree expander is linear in the number of vertices. Also, the treewidth of the expander graph is $\Theta(k)$, which is not changed by subdividing edges.

Consider any of the twice subdivided edges, i.e. let P be a path of length 3 in $\overline{G}$ for which both internal vertices u, v have degree 2 in $\overline{G}$. Let e be one of the edges of P. If a strongly connected SCSS solution containing all vertices of the bidirected graph G does not use any of the two edges corresponding to e in G, then it needs to use all four of the other edges of G corresponding to the two edges of P different from e: this is the only way in which all other terminals can reach u and v, and u and v can reach all other terminals. Note also that it is not possible for a strongly connected solution to only use two directed edges corresponding to edges of P.

We can however construct a solution N in which for every edge of P we use exactly one directed edge of G, and this must then be optimal: the solution N initially contains one of the directed edges of G corresponding to an edge of $\overline{G}$ each. As the underlying undirected graph of N would be exactly $\overline{G}$, its treewidth is $\Omega(k)$, as claimed. However N might not yet be strongly connected. If there are two vertices u and v, for which no $u \to v$ path exists in N, we introduce such a path as follows. An expander cannot contain any bridge, and so $\overline{G}$ is 2-edge-connected. Thus by Menger's Theorem [20] there are two edge-disjoint paths P and Q between u and v in $\overline{G}$. Consider any poly-cycle O formed by edges of the paths in N corresponding to P and Q. By Lemma 2.1 we may replace O by a directed cycle without losing the connectivity between any pair of vertices of N for which a directed path already existed. Also the underlying undirected graph of the resulting solution N is still $\overline{G}$. After replacing every poly-cycle formed by edges corresponding to those of P and Q in this way, there will be a $u \to v$ path in N. We may repeat this procedure for any pair of vertices that does not have a path between them, until the solution is strongly connected. □

We prove that BI-SCSS is FPT via a similar decomposition to the one of Theorem 4.1 for BI-DSN$_{\text{PLANAR}}$ (or the Borchers and Du Theorem for ST). More concretely, we show that

any optimum solution to BI-SCSS can be decomposed into non-overlapping (i.e., edge-disjoint) poly-trees, each of which is a feasible solution to some demand pairs of the terminals. As a consequence, similar to the PAS of Theorem 1.1, we can compute optimum poly-tree solutions via Theorem 1.12, among which we can find a solution to BI-SCSS. Since, compared to Theorem 4.1, here we have the stronger property that poly-tree solutions in the decomposition for BI-SCSS do not overlap, we obtain an optimum solution this way. However, in contrast to Theorem 4.1 the number of terminals in each poly-tree is not bounded by any constant, and thus applying the algorithm of Theorem 1.12 needs FPT time instead of polynomial time. Due to this weaker property compared to Theorem 4.1, also no kernelization is implied by our decomposition for BI-SCSS, as this would require a polynomial time algorithm.[5] For the following statement we reuse the formulation of DSN (and thus in particular for BI-SCSS) in terms of pattern graphs, as introduced in Section 4.

Lemma 5.2. *Let G be a bidirected graph with terminal set R, and $N \subseteq G$ be the cheapest strongly connected subgraph containing R. There exists a set of patterns $\mathcal{H}$ such that*

1. *$V(H) \subseteq R$ for each $H \in \mathcal{H}$,*
2. *given any feasible solutions $N_H \subseteq G$ for all $H \in \mathcal{H}$, the union $\bigcup_{H\in\mathcal{H}} N_H$ of the these solutions strongly connects R, and*
3. *there exist feasible solutions $T^*_H \subseteq G$ for all $H \in \mathcal{H}$ where each T^*_H is a poly-tree and $\sum_{H\in\mathcal{H}} \text{cost}(T^*_H) = \text{cost}(N)$.*

Before proving this lemma we show that it implies the claimed FPT algorithm of Theorem 1.10. Note that we do not know the set $\mathcal{H}$ of Lemma 5.2 without knowing the solution N, which we wish to compute. However, a simple dynamic programming approach can be used to find N.

Proof of Theorem 1.10. We present an algorithm that is very similar to the one for Theorem 1.1. The first step of the algorithm is to compute the optimum poly-tree solutions to all patterns on the terminals R. Note that poly-trees are exactly the directed graphs with treewidth 1, so that for every pattern graph H on R we can set $\omega = 1$ in Theorem 1.12 to compute the best poly-tree solution (if any) in $2^{O(k)} \cdot n^{O(1)}$ time. Since there are $2\binom{k}{2} = k^2 - k$ possible edges for any pattern on R, and any subset of these may span a pattern H, there are 2^{k^2-k} possible patterns. Thus up to now the algorithm uses $2^{k^2+O(k)} \cdot n^{O(1)}$ time.

The next step is to use a dynamic program to compute a solution strongly connecting all of R by putting together these poly-tree solutions. More concretely, let $T_1, \ldots, T_p$ be all the poly-tree solutions computed in the first step (given in any arbitrary order). For any subset $\mathcal{T}$ of these poly-trees, in the following we denote by $\text{cost}(\mathcal{T}) := \sum_{T\in\mathcal{T}} \text{cost}(T)$ their total cost and by $\bigcup\mathcal{T} := \bigcup_{T\in\mathcal{T}} T$ their union. For $1 \leq i \leq p$ and any pattern graph H, we define

$$\sigma(H, i) = \min\left\{\text{cost}(\mathcal{T}) \;\middle|\; \mathcal{T} \subseteq \{T_1, \ldots, T_i\} \text{ and } \bigcup\mathcal{T} \text{ feasible for } H\right\} \tag{3}$$

to be the minimum total cost of a subset of the first i poly-trees $T_1, \ldots, T_i$ that forms a feasible solution to H. Note that the total cost of a set $\mathcal{T}$ counts edges appearing in more than one poly-tree of $\mathcal{T}$ several times. If no feasible solution to H can be obtained from any subset of $T_1, \ldots, T_i$, then we define $\sigma(H, i)$ to be ∞.

Let $\mathcal{H}$ be the set of patterns given by Lemma 5.2 for the optimum BI-SCSS solution N, and let T^*_H be the poly-tree solution to each $H \in \mathcal{H}$ given by the lemma. The existence of T^*_H in particular implies that for each $H \in \mathcal{H}$ there is a feasible poly-tree solution N_H among

[5] For this reason Lemma 5.2 is "merely" a lemma, while Theorem 4.1 is a theorem.

$T_1, \ldots, T_p$, and by Lemma 5.2 their union $\bigcup_{H \in \mathcal{H}} N_H$ strongly connects R. Thus if H^* is any strongly connected pattern graph on R (e.g., a directed cycle on R), then

$$\sigma(H^*, p) \leq \sum_{H \in \mathcal{H}} \text{cost}(N_H) \leq \sum_{H \in \mathcal{H}} \text{cost}(T^*_H) = \text{cost}(N),$$

where the second inequality follows since each computed poly-tree T_i (and thus each N_H where $H \in \mathcal{H}$) is an optimum poly-tree solution according to Theorem 1.12. We conclude that $\sigma(H^*, p)$ is the value of the optimum BI-SCSS solution we wish to compute.

To recursively compute $\sigma(H, i)$ for any pattern graph H on R and any $1 \leq i \leq p$, we keep track of the subset $\mathcal{T}^i_H \subseteq \{T_1, \ldots, T_i\}$ of poly-trees that obtain the cost stored by the following dynamic program in $\sigma(H, i)$. For $i = 1$ we just check whether T_1 is a feasible solution to H. If so, we set $\sigma(H, 1) = \text{cost}(T_1)$ and $\mathcal{T}^1_H = \{T_1\}$, while otherwise we set $\sigma(H, 1) = \infty$ and $\mathcal{T}^1_H = \emptyset$. This obviously computes $\sigma(H, 1)$ correctly. To compute $\sigma(H, i)$ for any $i \geq 2$, we check for each pattern graph H' whether $(\bigcup \mathcal{T}^{i-1}_{H'}) \cup T_i$ is a feasible solution to H. Among all such solutions and the graph $\bigcup \mathcal{T}^{i-1}_H$ we store the cost of the cheapest option. More formally, we claim that for $i \geq 2$

$$\sigma(H, i) = \min \Big\{ \sigma(H, i-1), \sigma(H', i-1) + \text{cost}(T_i) \;\Big|\; H' \text{ is a pattern with } \Big(\bigcup \mathcal{T}^{i-1}_{H'}\Big) \cup T_i \text{ feasible for } H \Big\}. \quad (4)$$

If the right-hand side of (4) is some finite value, we set $\mathcal{T}^i_H$ to the subset obtaining the minimum (i.e., either $\mathcal{T}^{i-1}_H$ or $\mathcal{T}^{i-1}_{H'} \cup \{T_i\}$ for some H'). Otherwise, we let $\mathcal{T}^i_H = \emptyset$.

To show that the recursion given by (4) is correct, fix H and $i \geq 2$, and let $\mathcal{T}^* \subseteq \{T_1, \ldots, T_i\}$ be the subset of poly-trees defining $\sigma(H, i)$, i.e., $\mathcal{T}^*$ minimizes the right-hand side of (3). We need to show that $\text{cost}(\mathcal{T}^i_H) = \text{cost}(\mathcal{T}^*)$. First note that by (4), $\bigcup \mathcal{T}^i_H$ is a feasible solution to H and is the union of some subset of $T_1, \ldots, T_i$, and so $\text{cost}(\mathcal{T}^i_H) \geq \text{cost}(\mathcal{T}^*)$ by definition of $\mathcal{T}^*$. In case $T_i \notin \mathcal{T}^*$, we have $\text{cost}(\mathcal{T}^{i-1}_H) = \text{cost}(\mathcal{T}^*)$ by induction, and so $\text{cost}(\mathcal{T}^i_H) \leq \text{cost}(\mathcal{T}^*)$, since $\sigma(H, i-1) = \text{cost}(\mathcal{T}^{i-1}_H)$ is considered as one of the values over which (4) minimizes. In the other case when $T_i \in \mathcal{T}^*$, consider the graph $\bigcup(\mathcal{T}^* \setminus \{T_i\})$ obtained by taking the union of all poly-trees in $\mathcal{T}^*$ except T_i (note that it may still contain edges of T_i). Now let H' be the pattern graph on R, which contains an edge st if and only if $\bigcup(\mathcal{T}^* \setminus \{T_i\})$ contains an $s \to t$ path. By induction we have $\text{cost}(\mathcal{T}^{i-1}_{H'}) \leq \text{cost}(\mathcal{T}^* \setminus \{T_i\})$, and adding $\text{cost}(T_i)$ to both sides of this inequality we get $\text{cost}(\mathcal{T}^{i-1}_{H'}) + \text{cost}(T_i) \leq \text{cost}(\mathcal{T}^*)$, since $\mathcal{T}^*$ contains T_i. Moreover, $(\bigcup \mathcal{T}^{i-1}_{H'}) \cup T_i$ is a feasible solution to H, since $\bigcup \mathcal{T}^{i-1}_{H'}$ is a feasible solution to H' and adding T_i we obtain an $s \to t$ path between terminals $s, t \in R$ if and only if $\bigcup \mathcal{T}^*$ contains some $s \to t$ path as well. Hence $\text{cost}(\mathcal{T}^i_H) \leq \text{cost}(\mathcal{T}^{i-1}_{H'}) + \text{cost}(T_i)$, as the latter term is equal to $\sigma(H', i-1) + \text{cost}(T_i)$ and is considered as one of the values over which (4) minimizes. In conclusion, also if $T_i \in \mathcal{T}^*$ we have $\text{cost}(\mathcal{T}^i_H) \leq \text{cost}(\mathcal{T}^*)$ and so $\text{cost}(\mathcal{T}^i_H) = \text{cost}(\mathcal{T}^*)$. Thus the recursion given in (4) correctly computes the value of $\sigma(H, i)$ according to its definition in (3).

To bound the runtime of the dynamic program, recall that there are 2^{k^2-k} possible pattern graphs H on R, and the first step of the algorithm computes at most one poly-tree solution to each pattern H, i.e., $p \leq 2^{k^2-k}$. Thus the size of the table given by all entries $\sigma(H, i)$ (with $1 \leq i \leq p$) is at most $2 \cdot 2^{k^2-k}$. To compute one entry of the table via (4), we need to consider every pattern H' for each of which we perform a feasibility check, which can be done in polynomial time. Thus the runtime per entry is $2^{k^2-k} \cdot n^{O(1)}$, and the total runtime of the algorithm (including the first step) is bounded by $2^{k^2+O(k)} \cdot n^{O(1)}$. □

To complete this section we now prove Lemma 5.2 and show how to decompose any strongly connected solution in a bidirected graph.

Proof of Lemma 5.2. We will assume w.l.o.g. that in the cheapest solution $N \subseteq G$ each terminal has only 1 neighbour, and each Steiner vertex has exactly 3 neighbours. We may assume this according the transformation given in Section 2.2, just as for our earlier proof of Theorem 4.1. Furthermore, let G_N again be the graph spanned by the edge set $\{uv, vu \in E(G) \mid uv \in E(N)\}$. It is not hard to see that proving Lemma 5.2 for the obtained optimum solution N in G_N implies the same result for the original optimum solution in G.

We will first reduce the claim to solutions that have a 2-connected underlying undirected graph. In particular, consider a maximal 2-connected component $\overline{C}$ of $\overline{N}$, and the set of articulation points W of $\overline{N}$ contained in $\overline{C}$, i.e., $w \in W$ if and only if $w \in V(C)$ and w is adjacent to some vertex of $\overline{N}$ that is not in $\overline{C}$. We now claim that the directed subgraph C of N corresponding to $\overline{C}$ is an optimum strongly connected solution for the terminal set given by W. First off, note that C cannot contain any terminals from R, as we assume that every terminal in R has only one neighbour, while $\overline{C}$ is 2-connected. Since $\overline{C}$ is a maximal 2-connected component of $\overline{N}$, no path leaving C can return to C. So any $u \to v$ path connecting a pair of vertices $u, v \in W$ must be entirely contained in C. This means that C strongly connects W, since N is strongly connected. If C was not an optimum strongly connected solution for W, we could replace it by a cheaper one in N. This would result in a feasible solution to R but with smaller cost than N, which would contradict the optimality of N.

Since C is an optimum strongly connected solution for W, we are able to prove the next claim, which essentially follows from our main observation on solutions in bidirected graphs given by Lemma 2.2.

Claim 5.3. *Every cycle of $\overline{C}$ contains at least two vertices of W.*

Proof. Assume $\overline{C}$ contains a cycle O with at most one vertex from W. As $\overline{C}$ is 2-connected and C is a minimum cost solution for W, there are at least two vertices in W, and at least one of these does not lie on O. In particular, on the cycle O there must be vertices that have degree more than 2 in $\overline{C}$ where paths lead to vertices of $\overline{C}$ not on O. As $\overline{C}$ is 2-connected, by Menger's Theorem [20] any such path leading away from O from a vertex $u \in V(O)$ must eventually lead back to some vertex $v \in V(O)$. We assume that every vertex of $\overline{N}$ has degree at most 3, and so $u \neq v$. This means that there exists a non-empty path $P \subseteq O$ between u and v along O that contains no vertex of W as an internal vertex, since O contains at most one vertex from W.

We will now fix such a pair of vertices $u, v \in V(O)$ of degree 3 in $\overline{C}$, such that there is a u–v path $Q \subseteq \overline{C}$, which contains no edge of O. We choose the pair u, v under the minimality condition that the u–v path $P \subseteq O$ not containing an internal vertex from W is of minimum length. That is, there is no pair u', v' of vertices on P, so that at least one of u' and v' is an internal vertex of P, and so that there is a u'–v' path in $\overline{C}$, which contains no edge of O: otherwise the u'–v' subpath of P would be a shorter path not containing an internal vertex from W than P for the pair u, v. In particular, this means that any path from an internal vertex of P that leads away from O must lead back to a vertex of O that does not lie on P.

Assume that P has internal vertices of degree 3 in $\overline{C}$, and let w be the closest one to u on P. That is, there is a w–w' path Q' not containing any edge of O, such that w' lies on O but not on P, by our choice of u and v. Furthermore, the w–u subpath P' of P has no internal vertex of degree 3 in $\overline{C}$ by our choice of w, but it has length at least 1, as w is an internal vertex of P. Now consider the cycle O' formed by the u–v path Q, the v–w subpath of P (with edges not on P'), the w–w' path Q', and the w'–u path on O not containing v. As P' does not lie on O' but connects the vertices w and v of O', by Lemma 2.2 the path P' cannot be a single edge, since C is an optimum solution for W. Thus removing O' from $\overline{C}$ results in a connected component that consists of the subpath of P' connecting the non-empty set of internal vertices of P'. This is because w is the closest internal vertex of P with degree 3 to u, so that each internal vertex

of P' has degree 2 in $\overline{C}$. However none of the vertices of this connected component is from W, as P, and therefore P', has no internal vertex from W. This contradicts Lemma 2.2, as C is an optimum SCSS solution for the set W.

Thus we are left with the case when all internal vertices of P have degree 2 in $\overline{C}$. In this case we consider the cycle O' formed by the u–v path Q and the v–u path $Q' \subseteq O$ containing no edge of P. Again, note that P connects the two vertices u and v of the cycle O' but P does not lie on O'. Thus, as before, P cannot be a single edge by Lemma 2.2, so that the non-empty set of internal vertices of P induce a connected component after removing O' from $\overline{C}$. This connected component contains no vertex from W, which once more contradicts Lemma 2.2 since C is optimum. ⌟

This claim implies that we can partition the edges of $\overline{C}$ into sets spanning edge-disjoint trees with leaves from W and internal vertices not in W, as follows. Take any edge e of $\overline{C}$ and consider the set of paths $\mathcal{P}$ in $\overline{C}$ that contain e, have two vertices of W as endpoints, and only vertices not in W as internal vertices. Assume the paths in $\mathcal{P}$ together span a graph containing a cycle. By Claim 5.3 there is a vertex $w \in W$ on this cycle. This vertex w is the endpoint of two paths in $\mathcal{P}$, each of which contains a different edge incident to w on the cycle. Since both these paths also contain e, they span a cycle O containing w (O may be different from the former cycle). As none of the internal vertices of the two paths is from W while the endpoints are, the cycle O also contains no vertex from W apart from w (otherwise the paths could not share e). Hence we found a cycle O with only one vertex from W, which contradicts Claim 5.3, and so the set $\mathcal{P}$ spans a tree. As we can find such a set of paths for every edge of $\overline{C}$, we can also find the desired edge partition for which each set spans a tree with leaves from W and internal vertices not from W. Let $\mathcal{T}_C$ be the set containing the graphs in C of treewidth 1 corresponding to these trees in $\overline{C}$.

We now extend the graphs of $\mathcal{T}_C$ of all 2-connected components $\overline{C}$ into edge-disjoint poly-trees of N, for which the leaves are terminals in R, as follows. Each graph $T \in \mathcal{T}_C$ is a poly-tree of C, since every edge of C lies on a cycle, for which by Lemma 2.2 no reverse edge exists in N. However a leaf w of T is not a terminal from R but an articulation point of $\overline{N}$, i.e a vertex of the corresponding set W. The 2-connected components of $\overline{N}$ are connected through these articulation points by trees, for which the leaves are terminals or articulation points of $\overline{N}$. As N is strongly connected, such a tree corresponds to a bidirected graph T of treewidth 1 in N. This means that fixing one of the leaves r of T, the graph T is the edge-disjoint union of an in- and an out-arborescence on the same vertex set both with root r. We denote by $\mathcal{A}$ the set of edge-disjoint in- and out-arborescences connecting the earlier-defined components C of N for which $\overline{C}$ is 2-connected. Note that N is the disjoint union of all poly-trees in the sets $\mathcal{T}_C$ and the arborescences in $\mathcal{A}$.

Since we assume that every vertex of N has at most 3 neighbours and $\overline{C}$ is 2-connected, an articulation point w of $\overline{N}$ in $\overline{C}$ has two neighbours in C and one neighbour outside of C. Thus w is either the root or a leaf of the two arborescences of $\mathcal{A}$ containing w, and it is a leaf of two edge-disjoint poly-trees of $\mathcal{T}_C$. In particular there are exactly four edges incident to w. One of the arborescences $A \in \mathcal{A}$ has an edge for which w is the tail, while the other $A' \in \mathcal{A}$ has an edge for which w is the head. This means that w must be the head of an edge of a poly-tree $T \in \mathcal{T}_C$, and the tail of an edge of a poly-tree $T' \in \mathcal{T}_C$. Taking the union of T and A, and also the union of T' and A', results in two edge-disjoint poly-trees in each of which every directed path of maximal length has two leaves of the resulting poly-tree as endpoints. These endpoints are either terminals or articulation points of $\overline{N}$ different from w. We can repeat this procedure at every articulation point of $\overline{N}$ to form two new edge-disjoint poly-trees, each from the union of two smaller poly-trees and/or arborescences. This will result in larger and larger poly-trees,

until we obtain a partition of the edges of N into sets, each of which spans a poly-tree in which every maximal length directed path connects two terminals of R that are leaves of the poly-tree. Let $\mathcal{T}_N$ denote the set of all these poly-trees.

For each poly-tree $T \in \mathcal{T}_N$, we introduce a pattern graph H to $\mathcal{H}$ having the subset of R contained in T as its vertex set, and having an edge st whenever T contains an $s \to t$ path. The solution T^*_H to H is exactly the poly-tree T. Now the lemma follows, since each pattern H has only terminals of R as vertices, and the solutions $T^*_H \subseteq N$ being edge-disjoint implies $\sum_{H \in \mathcal{H}} \text{cost}(T^*_H) = \text{cost}(N)$. Also, by construction of the pattern set $\mathcal{H}$ from N, any union of feasible solutions N_H for all $H \in \mathcal{H}$ will have the same connectivity between terminals as N, and thus the union strongly connects R. □

Theorem 1.11 shows that BI-SCSS is NP-hard, and even has a $2^{o(k)} \cdot n^{O(1)}$ lower bound under ETH. Hence, to the best of our knowledge, the class of bidirected graphs is the first example where SCSS remains NP-hard but turns out to be FPT parameterized by the number of terminals k.

6 Runtime lower bounds

This section is devoted to proving runtime lower bounds. First, in Section 6.1 we describe a general gadget which is used in both Theorem 1.3 and Theorem 1.7. Then Section 6.2 and Section 6.3 contain the proof of W[1]-hardness of BI-DSN$_{\text{PLANAR}}$ and BI-DSN respectively. Finally, Section 6.4 contains the proof of NP-hardness and the $2^{o(k)} \cdot n^{O(1)}$ runtime lower bound for BI-SCSS$_{\text{PLANAR}}$.

6.1 Constructing a "uniqueness" gadget

For every integer n we define the following gadget U_n which contains $4n+4$ vertices (see Figure 4). Since we need many of these gadgets later on, we will denote vertices of U_n by $U_n(v)$ etc., in order to be able to distinguish vertices of different gadgets. All edges will have the same weight M, which we will fix later during the reductions. The gadget U_n is constructed as follows (we first construct an undirected graph, and then bidirect each edge):

- Introduce two source vertices $U_n(s_1), U_n(s_2)$, two target vertices $U_n(t_1), U_n(t_2)$, and for each $i \in [n]$ the four vertices $U_n(0_i), U_n(1_i), U_n(2_i), U_n(3_i)$.
- U_n has a path of three edges corresponding to each $i \in [n]$.
  - Let $i \in [n]$. Then we denote the path in U_n corresponding to i by $P_{U_n}(i) := U_n(0_i) - U_n(1_i) - U_n(2_i) - U_n(3_i)$.
  - Each of these edges is called a *base* edge and has weight M
- Finally we add the following edges:
  - $U_n(s_1)U_n(1_i)$ and $U_n(t_1)U_n(1_i)$ for each $i \in [n]$.
  - $U_n(s_2)U_n(2_i)$ and $U_n(t_2)U_n(2_i)$ for each $i \in [n]$.
  - Each of these edges is called a *connector* edge and has weight M.

After bidirecting all above undirected edges in the gadgets, we give the following definitions for the directed graph U_n.

Definition 6.1. The set of *boundary* vertices of U_n is $\bigcup_{i=1}^{n} \{U_n(0_i), U_n(3_i)\}$. For each $R \in \{0, 1, 2, 3\}$ the set of R-vertices of U_n is $\{U_n(R_i) : 1 \le i \le n\}$.

Definition 6.2. A set of edges E' of U_n satisfies the *in-out* property if each of the following four conditions is satisfied

- $U_n(s_1)$ can reach some boundary vertex

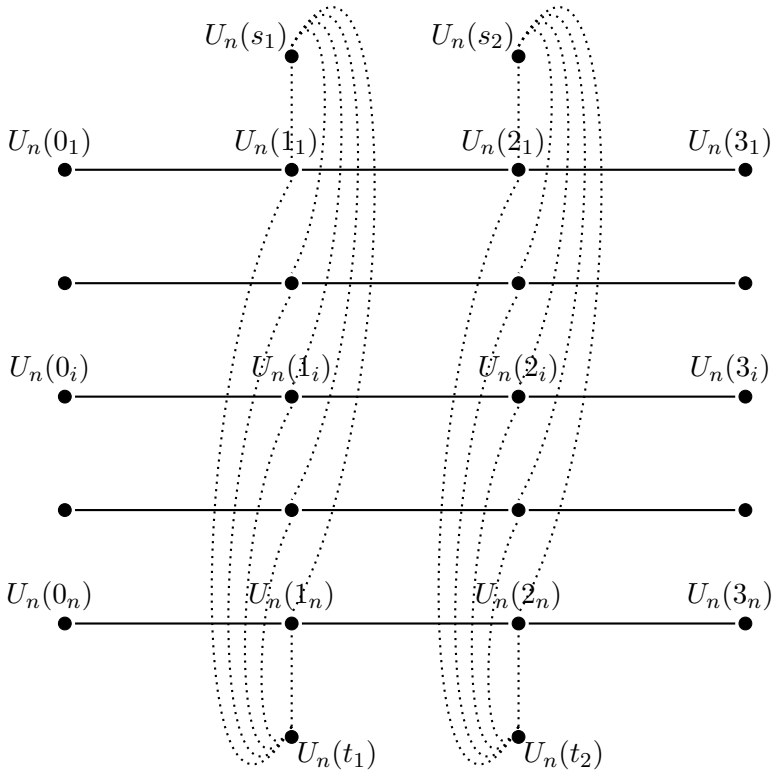

Figure 4: The construction of the uniqueness gadget U_n. Note that the gadget has $4n+4$ vertices. Each *base* edge is denoted by a filled edge and each connector edge is denoted by a dotted edge in the figure.

- $U_n(s_2)$ can reach some boundary vertex
- $U_n(t_1)$ can be reached from some boundary vertex
- $U_n(t_2)$ can be reached from some boundary vertex

Definition 6.3. A set of edges E' of U_n is *represented* by $i \in [n]$ and *right-oriented* (resp. *left-oriented*) if

- the connector edges in E' are $U_n(s_1) \to U_n(1_i), U_n(s_2) \to U_n(2_i), U_n(t_1) \leftarrow U_n(1_i)$, and $U_n(t_2) \leftarrow U_n(2_i)$, and
- base edges in E' are the directed path $P^{\text{right}}_{U_n}(i) := U_n(0_i) \to U_n(1_i) \to U_n(2_i) \to U_n(3_i)$ (resp. the path $P^{\text{left}}_{U_n}(i) := U_n(0_i) \leftarrow U_n(1_i) \leftarrow U_n(2_i) \leftarrow U_n(3_i)$).

We now show a lower bound on the weight of edges we need to pick from U_n to satisfy the in-out property.

Lemma 6.4. *Let E' be a set of edges of U_n which satisfies the in-out property. Then we have that either*
(i) the weight of E' is at least $8M$, or
(ii) the weight of E' is exactly $7M$ and there is an integer $i \in [n]$ such that E' is represented by i and is either left-oriented or right-oriented.

Proof. We clearly need at least four connector edges in E':

- one outgoing edge from $U_n(s_1)$ so that it can reach some boundary vertex,
- one outgoing edge from $U_n(s_2)$ so that it can reach some boundary vertex,
- one incoming edge into $U_n(t_1)$ so that it can be reached from some boundary vertex, and
- one incoming edge into $U_n(t_2)$ so that it can be reached from some boundary vertex.

This incurs a cost of $4M$ in E'. We now see how many base edges we must have in E'. We define the following:

- "0-1" edges: this is the set of edges $\{U_n(0_i) \leftrightarrow U_n(1_i) \ : \ 1 \le i \le n\}$
- "1-2" edges: this is the set of edges $\{U_n(1_i) \leftrightarrow U_n(2_i) \ : \ 1 \le i \le n\}$
- "2-3" edges: this is the set of edges $\{U_n(2_i) \leftrightarrow U_n(3_i) \ : \ 1 \le i \le n\}$

In each of the following four cases, we show that weight of E' is at least $8M$ (note that we have already shown that E' must contain at least four connector edges, and hence to show the lower bound of $8M$ on weight of E' we just need to show that it contains at least four base edges):

1. E' has no "0-1" edges: This implies that E' has at least 4 base edges from U_n: two rightward edges (one "1-2" and one "2-3") so that $U_n(s_1)$ can reach some boundary vertex, and two leftward edges (one "1-2" and one "2-3") so that $U_n(t_1)$ can be reached from some boundary vertex.
2. E' has no "2-3" edges: This implies that E' has at least 4 base edges from U_n: two leftward edges (one "0-1" and one "1-2") so that $U_n(s_2)$ can reach some boundary vertex, and two rightward edges (one "0-1" and one "1-2") so that $U_n(t_2)$ can be reached from some boundary vertex.
3. E' has no "1-2" edges: This implies that E' has at least 4 base edges from U_n: a leftward "0-1" edge so that $U_n(s_1)$ can reach some boundary vertex, a rightward "0-1" edge so that $U_n(t_1)$ can be reached from some boundary vertex, a leftward "2-3" edge so that $U_n(t_1)$ can be reached from some boundary vertex and a rightward "2-3" edge so that $U_n(s_2)$ can reach some boundary vertex.
4. E' has more than one edge of at least one of "0-1", "1-2" and "2-3" types: If E' does not contain at least one edge from each of the types "0-1", "1-2" and "2-3", then we are done by the three previous cases. Hence, E' contains at least one edge from each of the types "0-1", "1-2" and "2-3". Now, in the given case, if E' has more than one edge of at least one

of "0-1", "1-2" and "2-3" types then E' contains at least four base edges which is what we had to prove.

In each of the aforementioned four cases we have shown that weight of E' is at least $8M$. The only case that remains to be considered is when E' has *exactly one* edge of each of the types "0-1", "1-2" and "2-3". In this case, E' has weight exactly $7M$ and contains exactly four connector edges (one incident on each source and target vertex) and exactly three base edges (one each from "0-1", "1-2", and "2-3"). Let the four connector edges in E' be given by

- $U_n(s_1) \to U_n(1_{\beta_1})$
- $U_n(s_2) \to U_n(2_{\beta_2})$
- $U_n(t_1) \leftarrow U_n(1_{\beta_3})$
- $U_n(t_2) \leftarrow U_n(2_{\beta_4})$

Suppose that the (only) "1-2" edge of E' is rightward and given by $U_n(1_\beta) \to U_n(2_\beta)$. We will now show that $\beta = \beta_1 = \beta_2 = \beta_3 = \beta_4$ and that the three base edges of E' are exactly given by the path $P^{\text{right}}_{U_n}(\beta) := U_n(0_\beta) \to U_n(1_\beta) \to U_n(2_\beta) \to U_n(3_\beta)$.

- The unique "0-1" edge is rightward and given by $U_n(0_{\beta_3}) \to U_n(1_{\beta_3})$. Suppose the (unique) "0-1" edge is leftward: however this implies there is no incoming path to $U_n(t_1)$ which contradicts the fact that it can be reached from some boundary vertex. Since the unique "1-2" edge is rightward, it follows that the path in E', which connects some boundary vertex to $U_n(t_1)$, must use the unique "0-1" rightward edge which is hence forced to be $U_n(0_{\beta_3}) \to U_n(1_{\beta_3})$.
- The unique "2-3" edge is rightward and given by $U_n(2_{\beta_2}) \to U_n(3_{\beta_2})$. Suppose the (unique) "2-3" edge is leftward: however this implies there is no outgoing path from $U_n(s_2)$ (since the unique "1-2" edge is rightward and the unique "2-3" edge is leftward), which contradicts the fact that $U_n(s_2)$ can reach some boundary vertex. Since both the unique "1-2" edge and the unique "2-3" edge is rightward, it follows that the path in E' from $U_n(s_2)$ to some boundary vertex must use the unique "2-3" rightward edge which is hence forced to be $U_n(2_{\beta_2}) \to U_n(3_{\beta_2})$.

Hence, we have that the only base edges in E' are given by

- the rightward "0-1" edge $U_n(0_{\beta_3}) \to U_n(1_{\beta_3})$,
- the rightward "1-2" edge $U_n(1_\beta) \to U_n(2_\beta)$, and
- the rightward "2-3" edge $U_n(2_{\beta_2}) \to U_n(3_{\beta_2})$.

Now the existence of a path in E' from boundary vertex to $U_n(t_2)$ implies $\beta_3 = \beta = \beta_4$. Similarly, the existence of a path in E' from $U_n(s_1)$ to some boundary vertex implies $\beta_1 = \beta = \beta_2$. Hence, we have that $\beta = \beta_1 = \beta_2 = \beta_3 = \beta_4$, i.e., the three base edges in E' are exactly given by the path $P^{\text{right}}_{U_n}(\beta) := U_n(0_\beta) \to U_n(1_\beta) \to U_n(2_\beta) \to U_n(3_\beta)$. If the unique "1-2" edge in E' is leftward, then the arguments are symmetric. □

The following corollary follows immediately from the second part of proof of Lemma 6.4.

Corollary 6.5. *For every $i \in [n]$ there is a set of edges $E^{\text{right}}_{U_n}(i)$ $\left(\text{resp. } E^{\text{left}}_{U_n}(i)\right)$ of cost exactly $7M$ which represents i, is right-oriented (resp. left-oriented) and satisfies the "in-out" property.*

6.2 W[1]-hardness for BI-DSN$_{\text{PLANAR}}$

The goal of this section is to prove Theorem 1.3. We reduce from the GRID TILING problem introduced by Marx [61]:

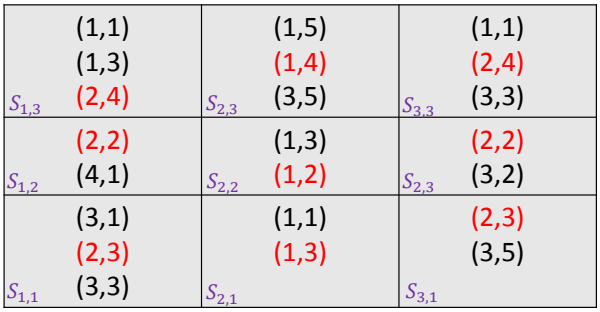

Figure 5: An instance of Grid Tiling with $k = 3, n = 5$ with a solution highlighted in red. Note that in a solution, all entries from a row agree in the second coordinate and all entries from a column agree in the first coordinate.

$k \times k$ Grid Tiling
Input: integers k, n, and a collection of k^2 non-empty sets $S_{i,j} \subseteq [n] \times [n]$ where $1 \leq i, j \leq k$.
Question: for each $1 \leq i, j \leq k$ does there exist a value $\gamma_{i,j} \in S_{i,j}$ such that
- if $\gamma_{i,j} = (x, y)$ and $\gamma_{i,j+1} = (x', y')$ then $x = x'$, and
- if $\gamma_{i,j} = (x, y)$ and $\gamma_{i+1,j} = (x', y')$ then $y = y'$.

See Figure 5 for example of an instance of Grid Tiling. Under ETH [44, 45], it was shown by Chen et al. [12] that k-Clique does not admit an algorithm running in time $f(k) \cdot n^{o(k)}$ for any computable function f. There is a simple reduction [19, Theorem 14.28] from k-Clique to $k \times k$ Grid Tiling implying the same runtime lower bound for the latter problem. To prove Theorem 1.3, we give a reduction which transforms an instance $\left(k, n, \{S_{i,j}\}_{1 \leq i,j \leq k}\right)$ of Grid Tiling into an instance $(G^*, \mathcal{D})$ of bi-DSN$_{\text{Planar}}$ which has a planar optimum and the number of terminals is $|\mathcal{D}| = O(k^2)$. We design two types of gadgets: the *main gadget* and the *secondary gadget*. The reduction from Grid Tiling represents each cell of the grid with a copy of the main gadget, and each main gadget is surrounded by four secondary gadgets: on the top, right, bottom and left. Each of these gadgets are actually copies of the "uniqueness gadget" from Section 6.1 with $M = k^4$: each secondary gadget is a copy of U_n and for each $1 \leq i, j \leq k$ the main gadget $\mathrm{M}_{i,j}$ (corresponding to the set $S_{i,j}$) is a copy of $U_{|S_{i,j}|}$. We refer to Figure 6 (bird's-eye view) and Figure 7 (zoomed-in view) for an illustration of the reduction.

Fix some $1 \leq i, j \leq k$. The main gadget $\mathrm{M}_{i,j}$ has four secondary gadgets[6] surrounding it:
- above $\mathrm{M}_{i,j}$ is the vertical secondary gadget $\mathrm{VS}_{i,j+1}$,
- on the right of $\mathrm{M}_{i,j}$ is the horizontal secondary gadget $\mathrm{HS}_{i+1,j}$,
- below $\mathrm{M}_{i,j}$ is the vertical secondary gadget $\mathrm{VS}_{i,j}$, and
- on the left of $\mathrm{M}_{i,j}$ is the horizontal secondary gadget $\mathrm{HS}_{i,j}$.

Hence, there are $k(k+1)$ horizontal secondary gadgets and $k(k+1)$ vertical secondary gadgets. Recall that $\mathrm{M}_{i,j}$ is a copy of $U_{|S_{i,j}|}$ and each of the secondary gadgets are copies of U_n (with $M = k^4$). With slight abuse of notation, we assume that the rows of $\mathrm{M}_{i,j}$ are indexed by the set $\{(x, y) : (x, y) \in S_{i,j}\}$. We add the following edges (in red color) of weight 1. For each $(x, y) \in S_{i,j}$ add an edge connecting
- $\mathrm{VS}_{i,j+1}(3_x)$ and $\mathrm{M}_{i,j}(0_{(x,y)})$,
- $\mathrm{HS}_{i,j}(3_y)$ and $\mathrm{M}_{i,j}(0_{(x,y)})$,

[6] Half of the secondary gadgets are called "horizontal" since their base edges are horizontal (as seen by the reader), and the other half of the secondary gadgets are called "vertical".

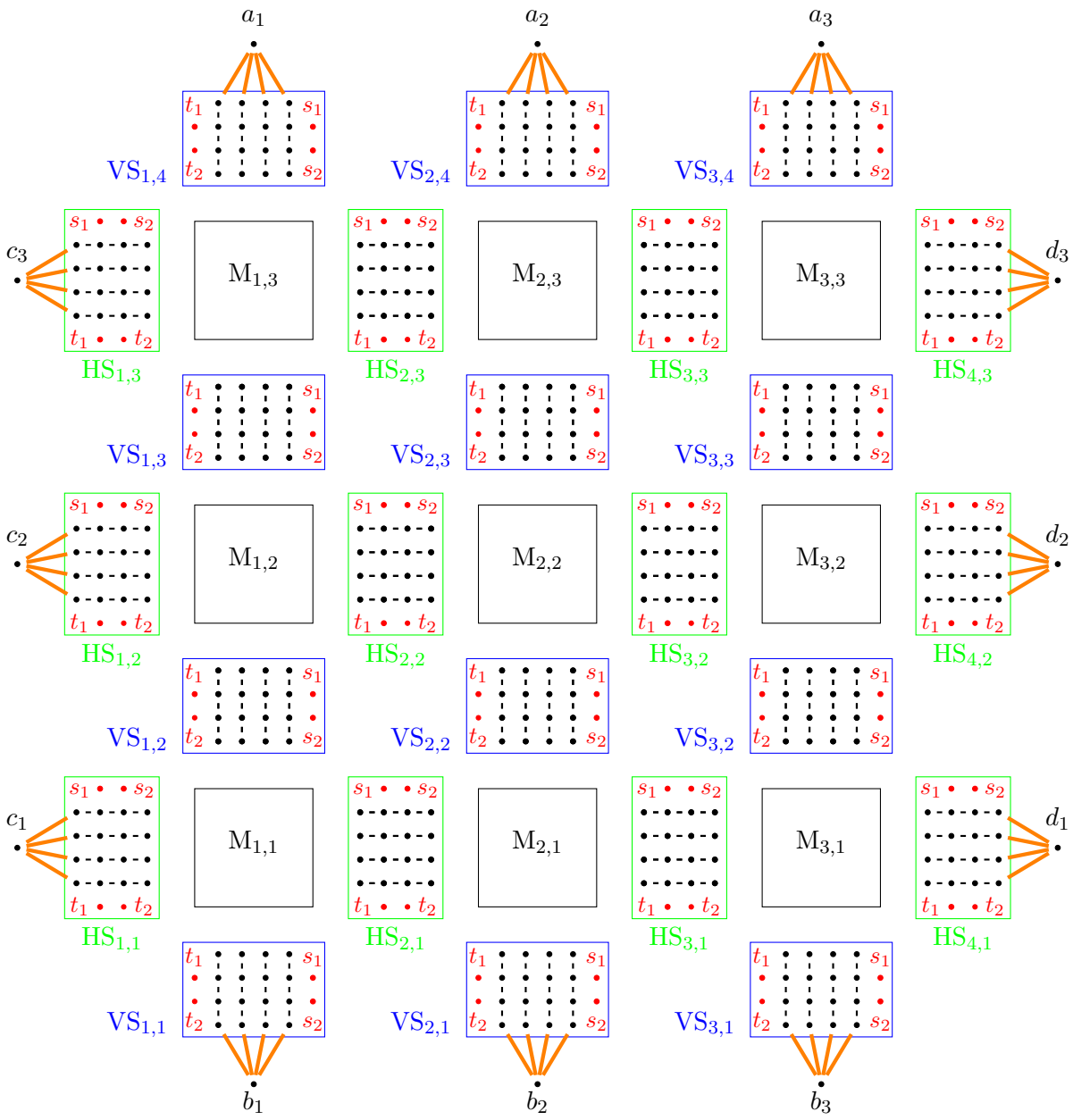

Figure 6: A bird's-eye view of the instance of G^* with $k = 3$ and $n = 4$ (see Figure 7 for a zoomed-in view). The connector edges within each main and secondary gadget are not shown. Similarly, the vertices and edges within each main gadget are not shown here either. Additionally we have some red edges between each main gadget and the four secondary gadgets surrounding it which are omitted in this figure for clarity (they are shown in Figure 7 which gives a more zoomed-in view).

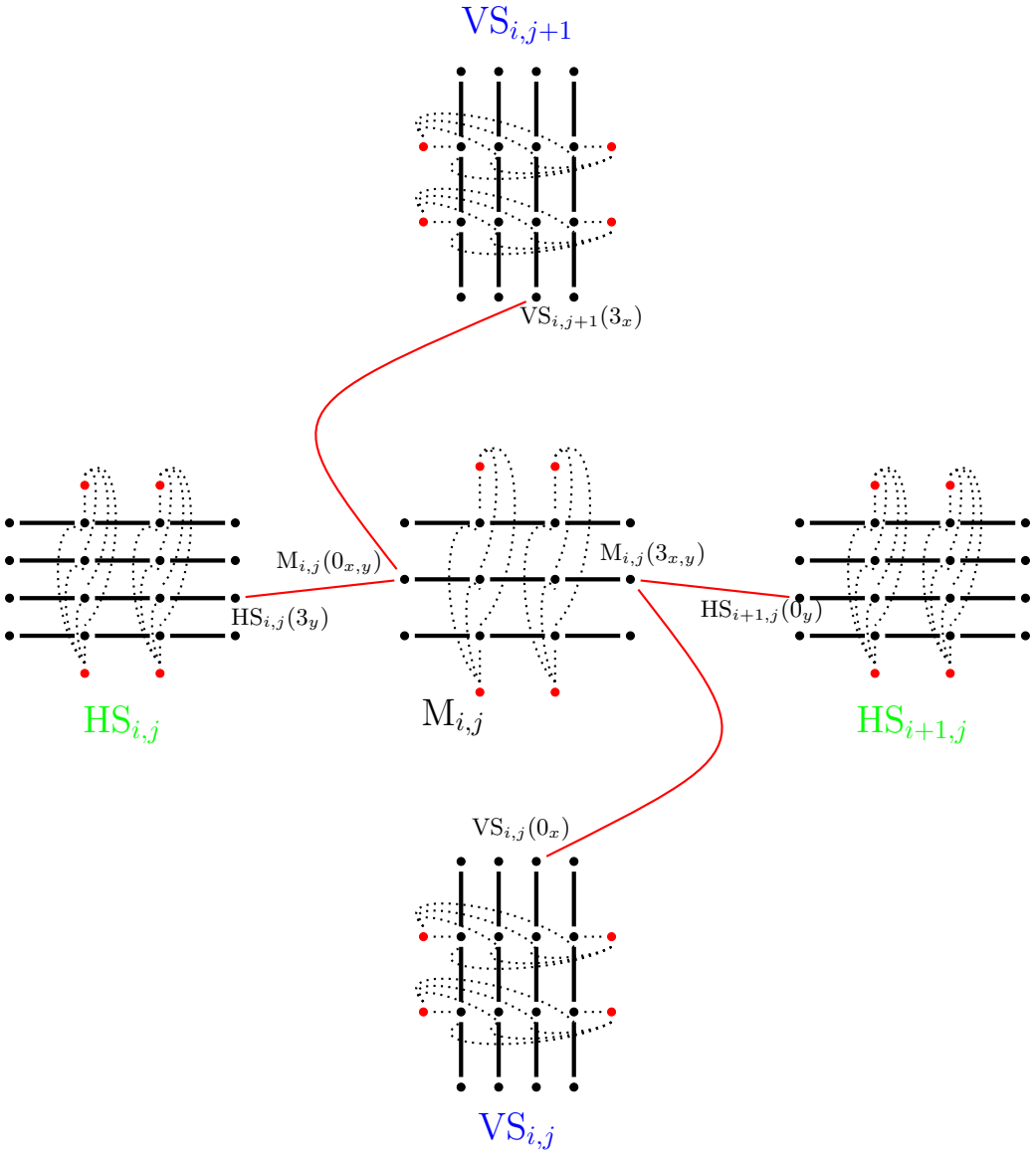

Figure 7: A zoomed-in view of the main gadget $M_{i,j}$ surrounded by four secondary gadgets: vertical gadget $VS_{i,j+1}$ on the top, horizontal gadget $HS_{i,j}$ on the left, vertical gadget $VS_{i,j}$ on the bottom and horizontal gadget $HS_{i+1,j}$ on the right. Each of the secondary gadgets is a copy of the uniqueness gadget U_n (see Section 6.1) and the main gadget $M_{i,j}$ is a copy of the uniqueness gadget $U_{|S_{i,j}|}$. The only inter-gadget edges are the red edges: they have one end-point in a main gadget and the other end-point in a secondary gadget. We have shown four such red edges which are introduced for every $(x, y) \in S_{i,j}$.

- $\mathrm{HS}_{i+1,j}(0_y)$ and $\mathrm{M}_{i,j}(3_{(x,y)})$, and
- $\mathrm{VS}_{i,j}(0_x)$ and $\mathrm{M}_{i,j}(3_{(x,y)})$.

Introduce the following $4k$ vertices (which we call *border* vertices):

- $a_1, a_2, \ldots, a_k$,
- $b_1, b_2, \ldots, b_k$,
- $c_1, c_2, \ldots, c_k$,
- $d_1, d_2, \ldots, d_k$.

For each $i \in [k]$ add an edge (in orange color in Figure 6) with weight 1 connecting

- a_i and $\mathrm{VS}_{i,k+1}(0_j)$ for each $j \in [n]$,
- b_i and $\mathrm{VS}_{i,1}(3_j)$ for each $j \in [n]$,
- c_i and $\mathrm{HS}_{1,i}(0_j)$ for each $j \in [n]$, and
- d_i and $\mathrm{HS}_{k+1,i}(3_j)$ for each $j \in [n]$.

We follow the convention that:

- $\mathrm{M}_{0,j}(s_1) = c_j = \mathrm{M}_{0,j}(s_2)$ and $\mathrm{M}_{k+1,j}(t_1) = d_j = \mathrm{M}_{k+1,j}(t_2)$ for each $j \in [k]$, and
- $\mathrm{M}_{i,0}(t_1) = b_i = \mathrm{M}_{i,0}(t_2)$ and $\mathrm{M}_{i,k+1}(s_1) = a_i = \mathrm{M}_{i,k+1}(s_2)$ for each $i \in [k]$.

This concludes the construction of the graph G^*. Note that we bidirect each edge of G^*. Finally, the set of demand pairs $\mathcal{D}$ is given by:

- Type I: Let $1 \le i \le k+1, 1 \le j \le k$. Consider the horizontal secondary gadget $\mathrm{HS}_{i,j}$. We add the pairs $\big(\mathrm{M}_{i-1,j}(s_1), \mathrm{HS}_{i,j}(t_1)\big)$ and $\big(\mathrm{M}_{i-1,j}(s_2), \mathrm{HS}_{i,j}(t_2)\big)$ in addition to the pairs $\big(\mathrm{HS}_{i,j}(s_1), \mathrm{M}_{i,j}(t_1)\big)$ and $\big(\mathrm{HS}_{i,j}(s_2), \mathrm{M}_{i,j}(t_2)\big)$.
- Type II: Let $1 \le j \le k+1, 1 \le i \le k$. Consider the vertical secondary gadget $\mathrm{VS}_{i,j}$. We add the pairs $(\mathrm{M}_{i,j}(s_1), \mathrm{VS}_{i,j}(t_1))$ and $(\mathrm{M}_{i,j}(s_2), \mathrm{VS}_{i,j}(t_2))$ in addition to the pairs $(\mathrm{VS}_{i,j}(s_1), \mathrm{M}_{i,j-1}(t_1))$ and $(\mathrm{VS}_{i,j}(s_2), \mathrm{M}_{i,j-1}(t_2))$

We have $O(k^2)$ vertical and horizontal secondary gadgets and we add $O(1)$ demand pairs corresponding to each of these gadgets. Hence, the total number of demand pairs is

$$|\mathcal{D}| = O(k^2) \tag{5}$$

Fix the budget $B^* = 4k + k(k+1) \cdot B + k(k+1) \cdot B + k^2 \cdot (B+4)$ where $B = 7M = 7k^4$. The high-level intuition is the following: we need $4k$ from the budget (via orange edges) just to include one edge incident on each of the $4k$ border vertices (each of which is part of a demand pair). We have $k(k+1)$ horizontal and vertical secondary gadgets each. We argue that any solution for BI-DSN must satisfy the in-out property in each of the secondary gadgets, and then invoke Lemma 6.4. Finally, for each main gadget, we again show that it must satisfy the in-out property and hence has cost at least B. However, here we show that we additionally need at least four red edges and hence the cost of any BI-DSN solution restricted to a main gadget is at least $B+4$. Since we have k^2 main gadgets, this completely uses up the budget B^*.

We now show the correctness of our reduction by showing that the instance $\big(k, n, \{S_{i,j}\}_{1 \le i,j \le k}\big)$ of Grid Tiling has a solution if and only if the instance $(G^*, \mathcal{D})$ of BI-DSN$_{\text{PLANAR}}$ has a planar solution of cost $\le B^*$. First we show that the forward direction:

Lemma 6.6. *If the instance $\big(k, n, \{S_{i,j}\}_{1 \le i,j \le k}\big)$ of Grid Tiling has a solution then the instance $(G^*, \mathcal{D})$ of BI-DSN$_{\text{PLANAR}}$ has a planar solution of cost $\le B^*$*

Proof. Suppose that Grid Tiling has a solution, i.e., there exist $1 \le \alpha_1, \alpha_2, \ldots, \alpha_k, \beta_1, \beta_2, \ldots, \beta_k \le n$ such that for each $1 \le i, j \le k$ we have $(\alpha_i, \beta_j) \in S_{i,j}$. We now build an edge set $E' \subseteq E(G^*)$ of weight $\le B^*$ such that the network $\big(V(G^*), E'\big)$ is a *planar* solution for the BI-DSN instance $(G^*, \mathcal{D})$. In the edge set E', we take the following edges:

1. The orange edges $(c_j, \mathrm{HS}_{1,j}(0_{\beta_j}))$ and $(\mathrm{HS}_{k+1,j}(3_{\beta_j}), d_j)$ for each $j \in [k]$. This uses up $2k$ from the budget since each of these edges has weight 1.
2. The orange edges $(a_i, \mathrm{VS}_{i,k+1}(0_{\alpha_i}))$ and $(\mathrm{VS}_{i,1}(3_{\alpha_i}), b_i)$ for each $i \in [k]$. This uses up $2k$ from the budget since each of these edges has weight 1.
3. For each $1 \le i, j \le k$ for the main gadget $\mathrm{M}_{i,j}$, use Corollary 6.5 to pick a set of edges $E^{\mathrm{right}}_{\mathrm{M}_{i,j}}\big((\alpha_i, \beta_j)\big)$ which is right-oriented, represented by (α_i, β_j) and has weight exactly B. Additionally we also pick the following four red edges (each of which has weight 1):
   - $\mathrm{VS}_{i,j+1}(3_{\alpha_i}) \to \mathrm{M}_{i,j}(0_{\alpha_i,\beta_j})$
   - $\mathrm{HS}_{i,j}(3_{\beta_j}) \to \mathrm{M}_{i,j}(0_{\alpha_i,\beta_j})$
   - $\mathrm{VS}_{i,j}(0_{\alpha_i}) \leftarrow \mathrm{M}_{i,j}(3_{\alpha_i,\beta_j})$
   - $\mathrm{HS}_{i+1,j}(0_{\beta_j}) \leftarrow \mathrm{M}_{i,j}(3_{\alpha_i,\beta_j})$
4. For each $1 \le j \le k+1$ and $1 \le i \le k$ for the vertical secondary gadget $\mathrm{VS}_{i,j}$, use Corollary 6.5 to pick a set of edges $E^{\mathrm{right}}_{\mathrm{VS}_{i,j}}(\alpha_i)$ which is right-oriented, represented by α_i and has weight exactly B.
5. For each $1 \le j \le k$ and $1 \le i \le k+1$ for the horizontal secondary gadget $\mathrm{HS}_{i,j}$, use Corollary 6.5 to pick a set of edges $E^{\mathrm{right}}_{\mathrm{HS}_{i,j}}(\beta_j)$ which is right-oriented, represented by β_j and has weight exactly B.

It is easy to see that $\big(V(G^*), E'\big)$ is planar, as each application of Corollary 6.5 gives a planar graph, separate applications give vertex-disjoint graphs, and moreover, the red and orange edges do not destroy planarity either. The weight of the edge set E' is exactly $4k + k(k+1)\cdot B + k(k+1)\cdot B + k^2\cdot(B+4) = B^*$. We now show that the network $\big(V(G^*), E'\big)$ is indeed a solution for the BI-DSN instance $(G^*, \mathcal{D})$. Fix i, j such that $1 \le i \le k+1, 1 \le j \le k$. Consider the four demand pairs of Type I.

- $\big(\mathrm{M}_{i-1,j}(s_1), \mathrm{HS}_{i,j}(t_1)\big)$. This path is obtained by concatenation of the following paths:
  - from $E^{\mathrm{right}}_{\mathrm{M}_{i-1,j}}\big((\alpha_{i-1}, \beta_j)\big)$ use the path $\mathrm{M}_{i-1,j}(s_1) \to \mathrm{M}_{i-1,j}(1_{\alpha_{i-1},\beta_j}) \to \mathrm{M}_{i-1,j}(2_{\alpha_{i-1},\beta_j}) \to \mathrm{M}_{i-1,j}(3_{\alpha_{i-1},\beta_j})$
  - use the red edge $\mathrm{M}_{i-1,j}(3_{\alpha_{i-1},\beta_j}) \to \mathrm{HS}_{i,j}(0_{\beta_j})$
  - from $E^{\mathrm{right}}_{\mathrm{HS}_{i,j}}(\beta_j)$ use the path $\mathrm{HS}_{i,j}(0_{\beta_j}) \to \mathrm{HS}_{i,j}(1_{\beta_j}) \to \mathrm{HS}_{i,j}(t_1)$.
- $\big(\mathrm{M}_{i-1,j}(s_2), \mathrm{HS}_{i,j}(t_2)\big)$. This path is obtained by concatenation of the following paths:
  - from $E^{\mathrm{right}}_{\mathrm{M}_{i-1,j}}\big((\alpha_{i-1}, \beta_j)\big)$ use the path $\mathrm{M}_{i-1,j}(s_2) \to \mathrm{M}_{i-1,j}(2_{\alpha_{i-1},\beta_j}) \to \mathrm{M}_{i-1,j}(3_{\alpha_{i-1},\beta_j})$
  - use the red edge $\mathrm{M}_{i-1,j}(3_{\alpha_{i-1},\beta_j}) \to \mathrm{HS}_{i,j}(0_{\beta_j})$
  - from $E^{\mathrm{right}}_{\mathrm{HS}_{i,j}}(\beta_j)$ use the path $\mathrm{HS}_{i,j}(0_{\beta_j}) \to \mathrm{HS}_{i,j}(1_{\beta_j}) \to \mathrm{HS}_{i,j}(2_{\beta_j}) \to \mathrm{HS}_{i,j}(t_2)$.
- $\big(\mathrm{HS}_{i,j}(s_1), \mathrm{M}_{i,j}(t_1)\big)$. This path is obtained by concatenation of the following paths:
  - from $E^{\mathrm{right}}_{\mathrm{HS}_{i,j}}(\beta_j)$ use the path $\mathrm{HS}_{i,j}(s_1) \to \mathrm{HS}_{i,j}(1_{\beta_j}) \to \mathrm{HS}_{i,j}(2_{\beta_j}) \to \mathrm{HS}_{i,j}(3_{\beta_j})$,
  - use the red edge $\mathrm{HS}_{i,j}(3_{\beta_j}) \to \mathrm{M}_{i,j}(0_{\alpha_i,\beta_j})$
  - from $E^{\mathrm{right}}_{\mathrm{M}_{i,j}}\big((\alpha_i, \beta_j)\big)$ use the path $\mathrm{M}_{i,j}(0_{\alpha_i,\beta_j}) \to \mathrm{M}_{i,j}(1_{\alpha_i,\beta_j}) \to \mathrm{M}_{i,j}(t_1)$.
- $\big(\mathrm{HS}_{i,j}(s_2), \mathrm{M}_{i,j}(t_2)\big)$. This path is obtained by concatenation of the following paths:
  - from $E^{\mathrm{right}}_{\mathrm{HS}_{i,j}}(\beta_j)$ use the path $\mathrm{HS}_{i,j}(s_2) \to \mathrm{HS}_{i,j}(2_{\beta_j}) \to \mathrm{HS}_{i,j}(3_{\beta_j})$
  - use the red edge $\mathrm{HS}_{i,j}(3_{\beta_j}) \to \mathrm{M}_{i,j}(0_{\alpha_i,\beta_j})$
  - from $E^{\mathrm{right}}_{\mathrm{M}_{i,j}}\big((\alpha_i, \beta_j)\big)$ use the path $\mathrm{M}_{i,j}(0_{\alpha_i,\beta_j}) \to \mathrm{M}_{i,j}(1_{\alpha_i,\beta_j}) \to \mathrm{M}_{i,j}(2_{\alpha_i,\beta_j}) \to \mathrm{M}_{i,j}(t_2)$.

The analysis for demand pairs of Type II is analogous, and therefore omitted here. □

Our next lemma shows the reverse direction of the correctness of the reduction: a planar solution of small cost for the BI-DSN$_{\text{PLANAR}}$ instance implies a solution for the GRID TILING instance.

Lemma 6.7. *If the instance $(G^*, \mathcal{D})$ of* BI-DSN$_{\text{PLANAR}}$ *has a planar solution of cost $\leq B^*$ then the instance $\left(k, n, \{S_{i,j}\}_{1 \leq i,j \leq k}\right)$ of* GRID TILING *has a solution.*

Proof. Suppose that the instance $(G^*, \mathcal{D})$ of BI-DSN$_{\text{PLANAR}}$ has a planar solution, say $N := \left(V(G^*), E^*\right)$, of cost at most B^*.

Claim 6.8. *For every $1 \leq i \leq k+1, 1 \leq j \leq k$ the edge set E^* restricted to the horizontal secondary gadget $HS_{i,j}$ satisfies the in-out property. Hence, $HS_{i,j}$ uses up weight of at least B from the budget.*

Proof. Looking at the demand pairs in $\mathcal{D}$ of Type I, we observe that

- $\text{HS}_{i,j}(s_1)$ is the source of some demand pair whose other end-point lies outside of $\text{HS}_{i,j}$,
- $\text{HS}_{i,j}(s_2)$ is the source of some demand pair whose other end-point lies outside of $\text{HS}_{i,j}$,
- $\text{HS}_{i,j}(t_1)$ is the target of some demand pair whose other end-point lies outside of $\text{HS}_{i,j}$,
- $\text{HS}_{i,j}(t_2)$ is the target of some demand pair whose other end-point lies outside of $\text{HS}_{i,j}$.

Since the network $\left(V(G^*), E^*\right)$ is a solution of the BI-DSN instance, it follows that there is a path starting at $\text{HS}_{i,j}(s_1)$ which must leave the gadget $\text{HS}_{i,j}$, i.e., $\text{HS}_{i,j}(s_1)$ can reach either a 0-vertex or a 3-vertex. The other three conditions of Definition 6.2 follow by similar reasoning. By Lemma 6.4, it follows that $\text{HS}_{i,j}$ uses up weight of at least B from the budget. □

Analogous claims hold also for the vertical secondary gadgets and main gadgets:

Claim 6.9. *For every $1 \leq i \leq k, 1 \leq j \leq k+1$ the edge set E^* restricted to the vertical secondary gadget $VS_{i,j}$ satisfies the in-out property. Hence, $VS_{i,j}$ uses up weight of at least B from the budget.*

Claim 6.10. *For every $1 \leq i, j \leq k$ the edge set E^* restricted to the main gadget $M_{i,j}$ satisfies the in-out property. Hence, $M_{i,j}$ uses up weight of at least B from the budget.*

From Claim 6.8, Claim 6.9 and Claim 6.10 we know that each of the gadgets (horizontal secondary, vertical secondary and main) use up at least weight B in E^*. We now claim that E^* restricted to each vertical secondary gadget, horizontal secondary gadget and main gadget has weight exactly B. Suppose there is at least one gadget where E^* has weight more than B. By Lemma 6.4, the weight of E^* in this gadget is at least $8M = B + M$. Since $M = k^4$ and $B^* = 4k + k(k+1) \cdot B + k(k+1) \cdot B + k^2 \cdot (B+4)$, where $B = 7M$, the weight of E^* is at least $\Big(k(k+1) \cdot B + k(k+1) \cdot B + k^2 \cdot B\Big) + M = \Big(k(k+1) \cdot B + k(k+1) \cdot B + k^2 \cdot B\Big) + k^4 > \Big(k(k+1) \cdot B + k(k+1) \cdot B + k^2 \cdot B\Big) + 4(k+k^2) = B^*$ which is a contradiction (since $k^4 > 4k + 4k^2$ for $k \geq 3$).

Therefore, together with Lemma 6.4, we have the following:

Claim 6.11. *The weight of E^* restricted to each main gadget and each secondary gadget is exactly $B = 7M$. Moreover,*

- *for each $1 \leq i \leq k+1, 1 \leq j \leq k$, the horizontal secondary gadget $HS_{i,j}$ is represented by some $y_{i,j} \in [n]$ and is either right-oriented or left-oriented,*
- *for each $1 \leq i \leq k, 1 \leq j \leq k+1$, the vertical secondary gadget $VS_{i,j}$ is represented by some $x_{i,j} \in [n]$ and is either right-oriented or left-oriented, and*
- *for each $1 \leq i, j \leq k$, the main gadget $M_{i,j}$ is represented by some $(\lambda_{i,j}, \delta_{i,j}) \in S_{i,j}$ and is either right-oriented or left-oriented.*

We now show that for each $1 \leq i, j \leq k$, the edge set E^* must also contain some red edges which have exactly one end-point in vertices of $\text{M}_{i,j}$.

Claim 6.12. *For each $1 \leq i, j \leq k$, the edge set E^* must contain at least one red edge of each of the following four types:*

- *an edge with one end-point in the set of 3-vertices of $M_{i,j}$ and other end-point in the set of 0-vertices of $VS_{i,j}$*
- *an edge with one end-point in the set of 3-vertices of $M_{i,j}$ and other end-point in the set of 0-vertices of $HS_{i+1,j}$*
- *an edge with one end-point in the set of 0-vertices of $M_{i,j}$ and other end-point in the set of 3-vertices of $HS_{i,j}$*
- *an edge with one end-point in the set of 0-vertices of $M_{i,j}$ and other end-point in the set of 3-vertices of $VS_{i,j+1}$*

Proof. We show that E^* must use at least one red edge which has one end-point in the set of 3-vertices of $\mathrm{M}_{i,j}$ and the other end-point in the set of 0-vertices of $\mathrm{VS}_{i,j}$. Analogous arguments hold for the other three secondary gadgets surrounding the main gadget $\mathrm{M}_{i,j}$. Note that these four red edges are distinct since the vertex sets of the secondary gadgets are pairwise disjoint.

We know by Claim 6.11 that $\mathrm{VS}_{i,j}$ is either right-oriented or left-oriented. Suppose $\mathrm{VS}_{i,j}$ is right-oriented (the other case is analogous). By Lemma 6.4 and Claim 6.11, we know that the only base edges of $\mathrm{VS}_{i,j}$ picked in E^* are $\mathrm{VS}_{i,j}(0_{x_{i,j}}) \to \mathrm{VS}_{i,j}(1_{x_{i,j}}) \to \mathrm{VS}_{i,j}(2_{x_{i,j}}) \to \mathrm{VS}_{i,j}(3_{x_{i,j}})$ and the only connector edges of $\mathrm{VS}_{i,j}$ picked in E^* are $\mathrm{VS}_{i,j}(s_1) \to \mathrm{VS}_{i,j}(1_{x_{i,j}}), \mathrm{VS}_{i,j}(s_2) \to \mathrm{VS}_{i,j}(2_{x_{i,j}}), \mathrm{VS}_{i,j}(t_1) \leftarrow \mathrm{VS}_{i,j}(1_{x_{i,j}})$ and $\mathrm{VS}_{i,j}(t_2) \leftarrow \mathrm{VS}_{i,j}(2_{x_{i,j}})$. But there is a Type II demand pair whose target is $\mathrm{VS}_{i,j}(t_1)$: the path satisfying this demand pair has to enter $\mathrm{VS}_{i,j}$ through the vertex $\mathrm{VS}_{i,j}(0_{x_{i,j}})$. The only edges (which are not base or connector edges of $\mathrm{VS}_{i,j}$) incident on the 0-vertices of $\mathrm{VS}_{i,j}$ have their other end-point in the set of 3-vertices of $\mathrm{M}_{i,j}$. That is, E^* contains a red edge whose start vertex is a 3-vertex of $\mathrm{M}_{i,j}$ and end vertex is a 0-vertex of $\mathrm{VS}_{i,j}$. □

Claim 6.13. *For each $1 \leq i, j \leq k$, the edge set E^* must contain at least one orange edge of each of the following types:*

- *an outgoing edge from a_i,*
- *an incoming edge into b_i,*
- *an outgoing edge from c_j, and*
- *an incoming edge into d_j.*

Proof. Note that for each $i \in [k]$ the vertex a_i is the source of two Type II demand pairs, viz. $(a_i, \mathrm{VS}_{i,k+1}(t_1))$ and $(a_i, \mathrm{VS}_{i,k+1}(t_2))$. Hence E^* must contain at least one outgoing edge from a_i. The other three statements follow similarly. □

We show now that we have no slack, i.e., the weight of E^* must be exactly B^*.

Claim 6.14. *The weight of E^* is exactly B^*, and is inclusion-wise minimal.*

Proof. From Claim 6.11, we know that each main gadget and each secondary gadget contributes towards a weight of B in E^*. Claim 6.12 says that each main gadget needs at least 4 red edges, and Claim 6.13 says that the orange edges contribute at least $4k$ to weight of E^*. Since all these edges are distinct, we have that the weight of E^* is at least $k(k+1)\cdot B + k(k+1)\cdot B + k^2 \cdot B + 4(k+k^2) = B^*$. Hence, the weight of E^* is exactly B^*, and it is minimal (under edge deletions) since no edges have zero weights. □

Consider a main gadget $\mathrm{M}_{i,j}$. The main gadget has four secondary gadgets surrounding it: $\mathrm{VS}_{i,j}$ below it, $\mathrm{VS}_{i,j+1}$ above it, $\mathrm{HS}_{i,j}$ to the left and $\mathrm{HS}_{i+1,j}$ to the right. By Claim 6.11, these gadgets are represented by $x_{i,j}, x_{i,j+1}, y_{i,j}$ and $y_{i+1,j}$ respectively. The main gadget $\mathrm{M}_{i,j}$ is represented by $(\lambda_{i,j}, \delta_{i,j})$.

Claim 6.15 (propagation)**.** *For every main gadget $M_{i,j}$, we have $x_{i,j} = \lambda_{i,j} = x_{i,j+1}$ and $y_{i,j} = \delta_{i,j} = y_{i+1,j}$.*

Proof. Due to symmetry, it suffices to only argue that $x_{i,j} = \lambda_{i,j}$. Let us assume for the sake of contradiction that $x_{i,j} \neq \lambda_{i,j}$. We will now show that there is a vertex such that (1) there is exactly one edge adjacent to it in the solution E^* and (2) it does not belong to any demand pair. Observe that removing its only adjacent edge from E^* does not effect the validity of the solution. This contradicts Claim 6.14 which states that E^* is minimal.

From Claim 6.12 and Claim 6.14, it follows that E^* contains exactly one red edge, say e_1, which has one end-point in the set of 3-vertices of $\mathrm{M}_{i,j}$ and the other end-point in set of 0-vertices of $\mathrm{VS}_{i,j}$. Also, E^* contains exactly one red edge, say e_2, which has one end-point in the set of 3-vertices of $\mathrm{M}_{i,j}$ and the other end-point in set of 0-vertices of $\mathrm{HS}_{i+1,j}$.

Observe that if e_1 does not have one endpoint at $\mathrm{VS}_{i,j}(0_{x_{i,j}})$, then the vertex $\mathrm{VS}_{i,j}(0_{x_{i,j}})$ is the desired vertex. Hence, suppose that one endpoint of the edge e_1 is $\mathrm{VS}_{i,j}(0_{x_{i,j}})$. The other endpoint of e_1 must be $\mathrm{M}_{i,j}(3_{x_{i,j},y})$ for some $y \in V_j$. Since $x_{i,j} \neq \lambda_{i,j}$, we have $\mathrm{M}_{i,j}(3_{x_{i,j},y}) \neq \mathrm{M}_{i,j}(3_{\lambda_{i,j},\delta_{i,j}})$. Suppose that one of the end-points of e_2 is $\mathrm{M}_{i,j}(3_{x',y'})$. Since $\mathrm{M}_{i,j}(3_{x_{i,j},y}) \neq \mathrm{M}_{i,j}(3_{\lambda_{i,j},\delta_{i,j}})$, at least one of the following must be true: $\mathrm{M}_{i,j}(3_{x_{i,j},y}) \neq \mathrm{M}_{i,j}(3_{x',y'})$ or $\mathrm{M}_{i,j}(3_{\lambda_{i,j},\delta_{i,j}}) \neq \mathrm{M}_{i,j}(3_{x',y'})$. If $\mathrm{M}_{i,j}(3_{x_{i,j},y}) \neq \mathrm{M}_{i,j}(3_{\lambda_{i,j},\delta_{i,j}})$, then $\mathrm{M}_{i,j}(3_{x_{i,j},y})$ is the desired vertex. Otherwise, if $\mathrm{M}_{i,j}(3_{\lambda_{i,j},\delta_{i,j}}) \neq \mathrm{M}_{i,j}(3_{x',y'})$, then $\mathrm{M}_{i,j}(3_{\lambda_{i,j},\delta_{i,j}})$ is the desired vertex. In all cases, we have found a vertex with the desired properties. Hence, we have arrived at a contradiction. □

By Claim 6.15, it follows that for each $1 \leq i,j \leq k$ we have $x_{i,j} = \lambda_{i,j} = x_{i,j+1}$ and $y_{i,j} = \delta_{i,j} = y_{i+1,j}$ in addition to $(\lambda_{i,j}, \delta_{i,j}) \in S_{i,j}$ (by the definition of the main gadget). This implies that the instance $\left(k, n, \{S_{i,j}\}_{1 \leq i,j \leq k}\right)$ of Grid Tiling has a solution. This concludes the proof of Lemma 6.7. □

Finally we are ready to prove Theorem 1.3 which is restated below:

Theorem 1.3. *The* bi-DSN$_{\text{Planar}}$ *problem is* W[1]*-hard parameterized by k. Moreover, under ETH, for any computable functions $f(k)$ and $f(k,\varepsilon)$, the* bi-DSN$_{\text{Planar}}$ *problem*

- *has no $f(k) \cdot n^{o(\sqrt{k})}$ time algorithm to compute an optimum solution, i.e., a solution with cost at most that of the cheapest planar one, and*
- *has no $f(k,\varepsilon) \cdot n^{o(\sqrt{k})}$ time algorithm to compute a solution with cost at most $(1+\varepsilon)$ times that of the cheapest planar one, if $\varepsilon > 0$ is part of the input.*

Proof. Each main gadget has $O(n^2)$ vertices and G^* has $O(k^2)$ main gadgets. Each secondary gadget has $O(n)$ vertices and G^* has $O(k^2)$ secondary gadgets. The number of border vertices in G^* is $4k$. Hence, the total number of vertices in the graph G^* is $O(n^2k^2) = \mathrm{poly}(n,k)$. It is easy to see that G^* can be constructed in $\mathrm{poly}(n,k)$ time from a given instance $\left(k, n, \{S_{i,j}\}_{1 \leq i,j \leq k}\right)$ of Grid Tiling. It is known [19, Theorem 14.28] that $k \times k$ Grid Tiling is W[1]-hard parameterized by k, and under ETH cannot be solved in $f(k) \cdot n^{o(k)}$ time for any computable function f. Combining the two directions from Lemma 6.6 and Lemma 6.7, we get a parameterized reduction from $k \times k$ Grid Tiling to an instance of bi-DSN$_{\text{Planar}}$ with $|\mathcal{D}| = O(k^2)$ terminal pairs (Equation 5). Hence, it follows that bi-DSN$_{\text{Planar}}$ is W[1]-hard parameterized by the number k of terminal pairs and under ETH cannot be solved in $f(k) \cdot n^{o(\sqrt{k})}$ time for any computable function f.

Suppose now that there is an algorithm $\mathbb{A}$ which runs in time $f(k,\varepsilon) \cdot n^{o(\sqrt{k})}$ and computes an $(1+\varepsilon)$-approximate solution for bi-DSN$_{\text{Planar}}$. Recall that our reduction works as follows: the instance $\left(k, n, \{S_{i,j}\}_{1 \leq i,j \leq k}\right)$ of Grid Tiling is a YES instance if and only if the instance $(G^*, \mathcal{D})$ of bi-DSN$_{\text{Planar}}$ has a planar solution of cost $B^* = 4k + k(k+1) \cdot B + k(k+1) \cdot B + k^2 \cdot (B+4) = O(k^6)$ since $B = O(k^4)$. Consequently, consider running $\mathbb{A}$ with ε set to a value such that

$(1+\varepsilon)\cdot B^* < B^*+1$ with ε being a function of the parameter k independent of n. Every edge of our constructed graph G^* has weight at least 1, and hence a $(1+\epsilon)$-approximation is in fact forced to find a solution of cost at most B^*, i.e., $\mathbb{A}$ finds an optimum solution. By the previous paragraph, this is not possible. □

6.3 W[1]-hardness for bi-DSN

The goal of this section is to prove Theorem 1.7. We reduce from the Colored Subgraph Isomorphism problem[7] introduced by Marx [62].

Colored Subgraph Isomorphism (PSI)
Input: An undirected graph $G=(V_G,E_G)$ and $H=(V_H=\{1,2,\ldots,\ell\},E_H)$, and a partition of V_G into disjoint subsets $V_1,V_2,\ldots,V_\ell$
Question: Is there a function $\phi: V_H \to V_G$ such that
1. for every $i\in[\ell]$ we have $\phi(i)\in V_i$, and
2. for every edge $ij\in E_H$ we have $\phi(i)\phi(j)\in E_G$.

The W[1]-hardness of Colored Subgraph Isomorphism parameterized by $|E_H|$ follows since the W[1]-hard problem Multicolored Clique [36, 69] is a special case when H is a clique. Marx [62, Corollary 6.3] showed the following stronger lower bound: under ETH, Colored Subgraph Isomorphism cannot be solved in time $f(|E_H|)\cdot|V_G|^{o\left(|E_H|/\log|E_H|\right)}$ for any computable function f. To prove Theorem 1.7, we give a reduction which transforms an instance (G,H) of Colored Subgraph Isomorphism into an instance $(G^*,\mathcal{D})$ of bi-DSN which has $|\mathcal{D}|=O(|E_H|)$ demand pairs. This reduction is a modification of that given for Theorem 1.3 in Section 6.2: there we had the special case when H is a clique on ℓ vertices and hence could ensure planarity (Figure 6) of the optimum at the cost of a quadratic blowup in the number of demand pairs as compared to $|E_H|$. In this reduction, we lose that structure but achieve the condition that number of demand pairs in the instance of bi-DSN is linear in $|E_H|$.

Consider an instance of Colored Subgraph Isomorphism given by two undirected graphs $G=(V_G,E_G)$ and $H=(V_H=\{1,2,\ldots,\ell\},E_H)$, and a partition of V_G into disjoint subsets $V_1,V_2,\ldots,V_\ell$. We now build an instance $(G^*,\mathcal{D})$ of bi-DSN. Let $|E_H|=k$. We define the following quantities:

- For each $1\le i\le\ell$, $E_H^* := E_H\cup\{ii\ :1\le i\le\ell\}$, i.e., we introduce self-loops
- For each $1\le i\ne j\le\ell$, $E_{\{i,j\}}\subseteq E_G$ is the set of edges which have one endpoint in V_i and the other endpoint in V_j. For $1\le j\le\ell$ let $E_{\{j,j\}}=\{ii\ :\ i\in V_j\}$.
- For each $1\le i\le\ell$, $N_H(i):=\{j\ :\ ij\in E_H^*\}$
- For each $1\le i\le\ell$, $\alpha_i=\min\{j\ :\ j\in N_H(i)\}$ and $\beta_i=\max\{j\ :\ j\in N_H(i)\}$.
- For each $1\le i\le\ell$, $N'_H(i):=N_H(i)\cup\{\ell+1\}$
- For each $1\le i,j\le\ell$ such that $ij\in E_H^*$ we define $\mathbf{next}_j(i)=\min\big\{\{\ell+1\}\cup\{r\ :r>j, ri\in E_H^*\}\big\}$. For each $i\in[\ell]$ we define $\mathbf{next}_0(i)=\alpha_i$ and $\mathbf{next}_{\ell+1}(i)=\ell+1$.
- For each $1\le i,j\le\ell$ such that $ij\in E_H^*$ we define $\mathbf{prev}_j(i)=\max\big\{\{0\}\cup\{r\ :r<j, ri\in E_H^*\}\big\}$. For each $i\in[\ell]$ we define $\mathbf{prev}_{\ell+1}(i)=\beta_i$ and $\mathbf{prev}_0(i)=0$.

Remark 6.16. *Note that the quantities α_i and β_i are well-defined since we can assume that H is connected (and hence has no isolated vertices) since otherwise we can solve the* Colored Subgraph Isomorphism *instance separately for each component of H.*

The graph G^* has two types of gadgets: the *main gadget* and the *secondary gadget*. Each of these gadgets are copies of the "uniqueness gadget" from Section 6.1 with $M=k^4$.

[7] This is sometimes also known as Colored Subgraph Isomorphism

- For each $1 \leq i, j \leq \ell$ such that $ij \in E_H^*$ the gadget $\mathrm{M}_{i,j}$ (corresponding to the set $E_{\{i,j\}}$) is a copy of $U_{|E_{\{i,j\}}|}$.
- For each $i \in [\ell]$ and each $j \in N'_H(i)$, the vertical secondary gadget $\mathrm{VS}_{i,j}$ is a copy of $U_{|V_i|}$.
- For each $j \in [\ell]$ and each $i \in N'_H(j)$, the horizontal secondary gadget $\mathrm{HS}_{i,j}$ is a copy of $U_{|V_j|}$.

Hence, we have a total of $2k + \ell$ main gadgets, and $2k + 2\ell$ horizontal and vertical secondary gadgets each. For each $1 \leq i, j \leq \ell$ such that $ij \in E_H^*$, the main gadget $\mathrm{M}_{i,j}$ is surrounded (Figure 8) by the following four secondary gadgets:

- $\mathrm{HS}_{i,j}$ on the left,
- $\mathrm{VS}_{i,j}$ on the bottom,
- $\mathrm{HS}_{\mathbf{next}_i(j),j}$ on the right, and
- $\mathrm{VS}_{i,\mathbf{next}_j(i)}$ on the top.

Recall that for each $1 \leq i, j \leq \ell$ such that $ij \in E_H^*$, the main gadget $\mathrm{M}_{i,j}$ is a copy of $U_{|E_{\{i,j\}}|}$ with $M = k^4$. With slight abuse of notation, we assume that the rows of $\mathrm{M}_{i,j}$ are indexed by the set $\left\{\{x, y\} \ : \ \{x, y\} \in E_{\{i,j\},x \in V_i, y \in V_j}\right\}$. For each $1 \leq i, j \leq \ell$ such that $ij \in E_H^*$, we add an edge (in red color) of weight 1 for each $\{x, y\} \in E_{\{i,j\}}$ connecting

- $\mathrm{VS}_{i,\mathbf{next}_j(i)}(3_x)$ and $\mathrm{M}_{i,j}(0_{(x,y)})$,
- $\mathrm{HS}_{i,j}(3_y)$ and $\mathrm{M}_{i,j}(0_{(x,y)})$,
- $\mathrm{HS}_{\mathbf{next}_i(j),j}(0_y)$ and $\mathrm{M}_{i,j}(3_{(x,y)})$, and
- $\mathrm{VS}_{i,j}(0_x)$ and $\mathrm{M}_{i,j}(3_{(x,y)})$.

Introduce the following 4ℓ vertices (which we call *border* vertices):

- $a_1, a_2, \ldots, a_\ell$
- $b_1, b_2, \ldots, b_\ell$
- $c_1, c_2, \ldots, c_\ell$
- $d_1, d_2, \ldots, d_\ell$

We follow the convention that:
- $\mathrm{M}_{0,j}(s_1) = c_j = \mathrm{M}_{0,j}(s_2)$ and $\mathrm{M}_{\ell+1,j}(t_1) = d_j = \mathrm{M}_{\ell+1,j}(t_2)$ for each $j \in [\ell]$, and
- $\mathrm{M}_{i,0}(t_1) = b_i = \mathrm{M}_{i,0}(t_2)$ and $\mathrm{M}_{i,\ell+1}(s_1) = a_i = \mathrm{M}_{i,\ell+1}(s_2)$ for each $i \in [\ell]$.

For each $i \in [\ell]$ add an edge (in orange color in Figure 6) with weight 1 connecting

- a_i and $\mathrm{VS}_{i,\ell+1}(0_x)$ for each $x \in V_i$,
- b_i and $\mathrm{VS}_{i,1}(3_x)$ for each $x \in V_i$,
- c_i and $\mathrm{HS}_{1,i}(0_x)$ for each $x \in V_i$, and
- d_i and $\mathrm{HS}_{\ell+1,i}(3_x)$ for each $x \in V_i$.

This concludes the construction of the graph G^*. Note that we bidirect each edge of G^*. We now define the set of demand pairs:

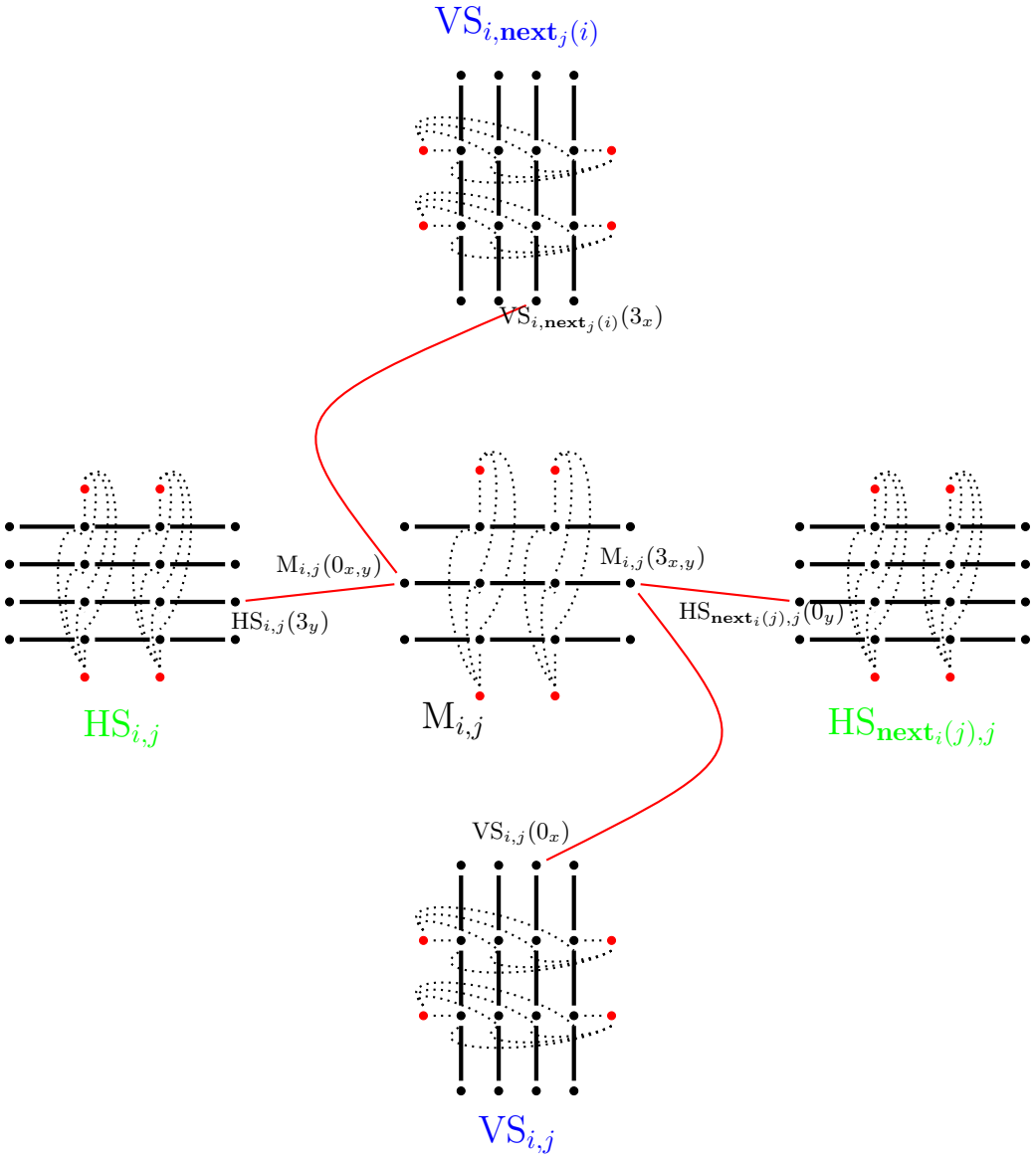

Figure 8: A zoomed-in view of the main gadget $\mathrm{M}_{i,j}$ surrounded by four secondary gadgets: vertical gadget $\mathrm{VS}_{i,\mathbf{next}_j(i)}$ on the top, horizontal gadget $\mathrm{HS}_{i,j}$ on the left, vertical gadget $\mathrm{VS}_{i,j}$ on the bottom and horizontal gadget $\mathrm{HS}_{\mathbf{next}_i(j),j}$ on the right. The main gadget $\mathrm{M}_{i,j}$ is a copy of the uniqueness gadget $U_{|E_{\{i,j\}}|}$ (see Section 6.1). The vertical gadgets $\mathrm{VS}_{i,j}$ and $\mathrm{VS}_{i,\mathbf{next}_j(i)}$ are copies of $U_{|V_i|}$. The horizontal gadgets $\mathrm{HS}_{i,j}$ and $\mathrm{HS}_{\mathbf{next}_i(j),j}$ are copies of $U_{|V_j|}$. The only inter-gadget edges are the red edges: they have one end-point in a main gadget and the other end-point in a secondary gadget. We have shown four such red edges which are introduced for every $(x, y) \in E_{\{i,j\}}$.

The set of demand pairs $\mathcal{D}$ is given by:

- Type I: For each $j \in [\ell]$ and each $i \in N'_H(j)$, we add the following four pairs involving the vertices of the horizontal secondary gadget $\mathrm{HS}_{i,j}$:
  - $\left(\mathrm{M}_{\mathbf{prev}_i(j),j}(s_1), \mathrm{HS}_{i,j}(t_1)\right)$
  - $\left(\mathrm{M}_{\mathbf{prev}_i(j),j}(s_2), \mathrm{HS}_{i,j}(t_2)\right)$
  - $\left(\mathrm{HS}_{i,j}(s_1), \mathrm{M}_{i,j}(t_1)\right)$
  - $\left(\mathrm{HS}_{i,j}(s_2), \mathrm{M}_{i,j}(t_2)\right)$
- Type II: For each $i \in [\ell]$ and each $j \in N'_H(i)$, we add the following four pairs involving the vertices of the vertical secondary gadget $\mathrm{VS}_{i,j}$:
  - $\left(\mathrm{M}_{i,j}(s_1), \mathrm{VS}_{i,j}(t_1)\right)$
  - $\left(\mathrm{M}_{i,j}(s_2), \mathrm{VS}_{i,j}(t_2)\right)$
  - $\left(\mathrm{VS}_{i,j}(s_1), \mathrm{M}_{i,\mathbf{prev}_j(i)}(t_1)\right)$
  - $\left(\mathrm{VS}_{i,j}(s_2), \mathrm{M}_{i,\mathbf{prev}_j(i)}(t_2)\right)$

We have $2k + 2\ell$ vertical and horizontal secondary gadgets each, and we add $O(1)$ demand pairs corresponding to each of these gadgets. Hence, the total number of demand pairs is

$$|\mathcal{D}| = O(2k + 2\ell) = O(k) \tag{6}$$

since we can assume that H is connected (6.16) which implies $k \geq \ell - 1$.

Fix the budget $B^* = 4\ell + (2k+2\ell) \cdot B + (2k+2\ell) \cdot B + (2k+\ell) \cdot (B+4)$ where $B = 7M = 7k^4$. The high-level intuition is the following: we need 4ℓ (via orange edges) from the budget just to include one edge incident on each of the 4ℓ border vertices (each of which is part of a demand pair). We have $(2k + 2\ell)$ horizontal and vertical secondary gadgets each. We argue that any solution for BI-DSN must satisfy the in-out property in each of the secondary gadgets, and then invoke Lemma 6.4. Finally, for each main gadget, we again show that it must satisfy the in-out property and hence has cost at least B. However, here we show that we additionally need at least four red edges (which have exactly one end-point in a main gadget) and hence the cost of any BI-DSN solution restricted to edges having at least one end-point in each main gadget is at least $B + 4$. Since we have $2k + \ell$ main gadgets, this completely uses up the budget B^*.

We now prove the correctness of our reduction by showing that the instance (G, H) of Colored Subgraph Isomorphism is a YES instance if and only if there is a solution to the BI-DSN instance $(G^*, \mathcal{D})$ with cost at most B^*. First we show the forward direction:

Lemma 6.17. *If the instance (G, H) of* Colored Subgraph Isomorphism *is a YES instance, then the instance $(G^*, \mathcal{D})$ of* BI-DSN *has a solution of cost at most B^*.*

Proof. Suppose that the PSI instance is a YES instance, i.e., there exists a function $\phi : V_H \to V_G$ such that

1. for every $i \in [\ell]$ we have $\phi(i) \in V_i$, and
2. for every edge $ij \in E_H$ we have $\phi(i)\phi(j) \in E_G$.

We now show that there is an edge set $E' \subseteq E(G^*)$ of weight B^* such that the network $\left(V(G^*), E'\right)$ is a solution BI-DSN instance $(G^*, \mathcal{D})$. The edge set E' consists of the following edges:

- For each $1 \leq i \leq \ell$, pick the edges $a_i \to \mathrm{VS}_{i,\mathbf{next}_{\beta_i}(i)}(0_{\phi(i)})$ and $\mathrm{VS}_{i,\alpha_i}(3_{\phi(i)}) \to b_i$.
- For each $1 \leq j \leq \ell$, pick the edges $c_j \to \mathrm{VS}_{\alpha_j,j}(0_{\phi(j)})$ and $\mathrm{VS}_{\mathbf{next}_{\beta_j}(j),j}(3_{\phi(j)}) \to d_j$.
- For each $1 \leq i, j \leq \ell$ such that $ij \in E^*_H$ pick the following edges (guaranteed to exist by Corollary 6.5).
  - The set of edges in $\mathrm{M}_{i,j}$ which is oriented rightwards, represents $\{\phi(i), \phi(j)\}$ and has weight M. Note that this is possible since $\mathrm{M}_{i,j}$ has a row corresponding to each edge of $E_{\{i,j\}}$, and $\phi(i)\phi(j) \in E_{\{i,j\}} \subseteq E_G$ by (1) and (2).

- For each $i \in [\ell], j \in N'_H(i)$ pick the following edges (guaranteed to exist by Corollary 6.5).
  - The set of edges in $\mathrm{VS}_{i,j}$ which is oriented rightwards, represents $\phi(i)$ and has weight M. Note that this is possible since $\mathrm{VS}_{i,j}$ has a row corresponding to each vertex of V_i, and $\phi(i) \in V(i)$ by (1).
- For each $j \in [\ell], i \in N'_H(j)$ pick the following edges (guaranteed to exist by Corollary 6.5).
  - The set of edges in $\mathrm{HS}_{i,j}$ which is oriented rightwards, represents $\phi(j)$ and has weight M. Note that this is possible since $\mathrm{HS}_{i,j}$ has a row corresponding to each vertex of V_j, and $\phi(j) \in V(j)$ by (1).
- For each $1 \leq i, j \leq \ell$ such that $ij \in E^*_H$ pick the four red edges connecting main gadgets and secondary gadgets given by
  - $\mathrm{VS}_{i,\mathbf{next}_j(i)}(3_{\phi(i)}) \to \mathrm{M}_{i,j}(0_{\phi(i),\phi(j)})$,
  - $\mathrm{HS}_{i,j}(3_{\phi(j)}) \to \mathrm{M}_{i,j}(0_{\phi(i),\phi(j)})$,
  - $\mathrm{M}_{i,j}(3_{\phi(i),\phi(j)}) \to \mathrm{VS}_{i,j}(0_{\phi(i)})$, and
  - $\mathrm{M}_{i,j}(3_{\phi(i),\phi(j)}) \to \mathrm{HS}_{\mathbf{next}_i(j),j}(0_{\phi(j)})$.

  Observe that these four red edges are guaranteed to exist since either $i \neq j$ which implies $\phi(i)\phi(j) \in E_{\{i,j\}} \subseteq E_G$ by (2) or otherwise $i = j$ which implies $\phi(i)\phi(i) \in E_{\{i,i\}}$ by (1).

It is easy to see that the cost of E' is $4\ell + (2k+2\ell) \cdot B + (2k+2\ell) \cdot B + (2k+\ell) \cdot (B+4) = B^*$. We now show that each demand pair of Type I is satisfied by the edge set E'.

- The pair $(\mathrm{M}_{\mathbf{prev}_i(j),j}(s_1), \mathrm{HS}_{i,j}(t_1))$ is satisfied by the path $\mathrm{M}_{\mathbf{prev}_i(j),j}(s_1) \to \mathrm{M}_{\mathbf{prev}_i(j),j}(1_{\phi(i),\phi(j)}) \to \mathrm{M}_{\mathbf{prev}_i(j),j}(2_{\phi(i),\phi(j)}) \to \mathrm{M}_{\mathbf{prev}_i(j),j}(3_{\phi(i),\phi(j)}) \to \mathrm{HS}_{i,j}(0_{\phi(j)}) \to \mathrm{HS}_{i,j}(1_{\phi(j)}) \to \mathrm{HS}_{i,j}(t_1)$ in N, since $i = \mathbf{next}_{\mathbf{prev}_i(j)}(j)$.
- The pair $(\mathrm{M}_{\mathbf{prev}_i(j),j}(s_2), \mathrm{HS}_{i,j}(t_2))$ is satisfied by the path $\mathrm{M}_{\mathbf{prev}_i(j),j}(s_2) \to \mathrm{M}_{\mathbf{prev}_i(j),j}(2_{\phi(i),\phi(j)}) \to \mathrm{M}_{\mathbf{prev}_i(j),j}(3_{\phi(i),\phi(j)}) \to \mathrm{HS}_{i,j}(0_{\phi(j)}) \to \mathrm{HS}_{i,j}(1_{\phi(j)}) \to \mathrm{HS}_{i,j}(2_{\phi(j)}) \to \mathrm{HS}_{i,j}(t_2)$ in N, since $i = \mathbf{next}_{\mathbf{prev}_i(j)}(j)$.
- The pair $(\mathrm{HS}_{i,j}(s_1), \mathrm{M}_{i,j}(t_1))$ is satisfied by the path $\mathrm{HS}_{i,j}(s_1) \to \mathrm{HS}_{i,j}(1_{\phi(j)}) \to \mathrm{HS}_{i,j}(2_{\phi(j)}) \to \mathrm{HS}_{i,j}(3_{\phi(j)}) \to \mathrm{M}_{i,j}(0_{\phi(i),\phi(j)}) \to \mathrm{M}_{i,j}(1_{\phi(i),\phi(j)}) \to \mathrm{M}_{i,j}(t_1)$ in N
- The pair $(\mathrm{HS}_{i,j}(s_2), \mathrm{M}_{i,j}(t_2))$ is satisfied by the path $\mathrm{HS}_{i,j}(s_2) \to \mathrm{HS}_{i,j}(2_{\phi(j)}) \to \mathrm{HS}_{i,j}(3_{\phi(j)}) \to \mathrm{M}_{i,j}(0_{\phi(i),\phi(j)}) \to \mathrm{M}_{i,j}(1_{\phi(i),\phi(j)}) \to \mathrm{M}_{i,j}(2_{\phi(i),\phi(j)}) \to \mathrm{M}_{i,j}(t_2)$ in N

The proof of satisfiability for the demand pairs of Type II is analogous. Hence, the network $\big(V(G^*), E'\big)$ is indeed a solution for the BI-DSN instance. □

Our next lemma shows the reverse direction of the correctness of the reduction: the existence of a solution of small cost for the BI-DSN instance implies a solution for the COLORED SUBGRAPH ISOMORPHISM instance.

Lemma 6.18. *If the instance $(G^*, \mathcal{D})$ of* BI-DSN *has a solution of cost at most B^*, then the instance (G, H) of* COLORED SUBGRAPH ISOMORPHISM *is a YES instance.*

Proof. The arguments here are almost identical to those from Lemma 6.7. Suppose that BI-DSN has a solution, say the network $\big(V(G^*), N\big)$, of cost at most $B^* = 4\ell + (2k+2\ell) \cdot B + (2k+2\ell) \cdot B + (2k+\ell) \cdot (B+4)$.

Claim 6.19. *For any $j \in [\ell], i \in N'_H(j)$ the edges in N which have both end-points in the horizontal secondary gadget $HS_{i,j}$ satisfy the in-out property. Hence, $HS_{i,j}$ uses up weight of at least B from the budget.*

Proof. Looking at the demand pairs in $\mathcal{D}$ of Type I, we observe that

- $\mathrm{HS}_{i,j}(s_1)$ is the source of some demand pair whose other end-point lies outside of $\mathrm{HS}_{i,j}$,
- $\mathrm{HS}_{i,j}(s_2)$ is the source of some demand pair whose other end-point lies outside of $\mathrm{HS}_{i,j}$,
- $\mathrm{HS}_{i,j}(t_1)$ is the target of some demand pair whose other end-point lies outside of $\mathrm{HS}_{i,j}$,
- $\mathrm{HS}_{i,j}(t_2)$ is the target of some demand pair whose other end-point lies outside of $\mathrm{HS}_{i,j}$.

Since the network $\big(V(G^*), N\big)$ is a solution of the BI-DSN instance, it follows that there is a path in N starting at $\mathrm{HS}_{i,j}(s_1)$ which must leave the gadget $\mathrm{HS}_{i,j}$, i.e., $\mathrm{HS}_{i,j}(s_1)$ can reach either a 0-vertex or a 3-vertex. The other three conditions of Definition 6.2 follow by similar reasoning. By Lemma 6.4, it follows that the edges in N which have both endpoints in $\mathrm{HS}_{i,j}$ use up weight of at least B from the budget. □

Analogous claims hold also for vertical secondary gadgets and main gadgets:

Claim 6.20. *For any $i \in [\ell], j \in N'_H(i)$ the edges in N which have both end-points in the vertical secondary gadget $VS_{i,j}$ satisfy the in-out property. Hence, $VS_{i,j}$ uses up weight of at least B from the budget.*

Claim 6.21. *For every i, j such that $ij \in E^*_H$ the edges in N which have both end-points the main gadget $M_{i,j}$ satisfy the in-out property. Hence, $M_{i,j}$ uses up weight of at least B from the budget.*

From Claim 6.19, Claim 6.20 and Claim 6.21 we know that each of the gadgets (horizontal secondary, vertical secondary and main) use up at least weight B in N. We now claim that the edge set N restricted to each vertical secondary gadget, horizontal secondary gadget and main gadget has weight exactly B. Suppose there is at least one gadget where the edges of N have weight more than B. By Lemma 6.4, the weight of edges of N in this gadget is at least $8M = B + M$. Since $M = k^4$ and $B^* = 4k + (2k + 2\ell) \cdot B + (2k + 2\ell) \cdot B + (2k + \ell) \cdot (B + 4)$, where $B = 7M$, the weight of N is at least

$$\begin{aligned}
&\Big((2k + 2\ell) \cdot B + (2k + 2\ell) \cdot B + (2k + \ell) \cdot B\Big) + M \\
&\qquad = \Big((2k + 2\ell) \cdot B + (2k + 2\ell) \cdot B + (2k + \ell) \cdot B\Big) + k^4 \\
&\qquad > \Big((2k + 2\ell) \cdot B + (2k + 2\ell) \cdot B + (2k + \ell) \cdot B\Big) + 24k \\
&\qquad \geq \Big((2k + \ell) \cdot B + (2k + \ell) \cdot B + (k + \ell) \cdot B\Big) + 4\ell + 4(2k + \ell) \\
&\qquad = B^*,
\end{aligned}$$

which is a contradiction $\big($since $k^4 > 24k$ for $k \geq 3$, and $k \geq \ell - 1$ since we can assume (6.16) that H is connected$\big)$.

Therefore, we have the following claim which follows from Lemma 6.4:

Claim 6.22. *The weight of N restricted to each gadget is exactly $B = 7M$. Moreover,*

- *for each $j \in [\ell], i \in N'_H(j)$, the horizontal secondary gadget $HS_{i,j}$ is represented by some $y_{i,j} \in V_j$ and is either right-oriented or left-oriented,*
- *for each $i \in [\ell], j \in N'_H(i)$, the vertical secondary gadget $VS_{i,j}$ is represented by some $x_{i,j} \in V_i$ and is either right-oriented or left-oriented, and*
- *for each i, j such that $ij \in E^*_H$, the main gadget $M_{i,j}$ is represented by some $(\lambda_{i,j}, \delta_{i,j}) \in E_{i,j}$ and is either right-oriented or left-oriented.*

We now show that for each $1 \leq i, j \leq \ell$ such that $ij \in E^*_H$, the edge set N must also contain some red edges which have exactly one end-point among vertices of $\mathrm{M}_{i,j}$.

Claim 6.23. *For each $1 \leq i, j \leq \ell$ such that $ij \in E^*_H$, the edge set N must also contain at least one red edge of each of the following four types:*

- *an edge with one end-point in the set of 3-vertices of $M_{i,j}$ and other end-point in the set of 0-vertices of $VS_{i,j}$*
- *an edge with one end-point in the set of 3-vertices of $M_{i,j}$ and other end-point in the set of 0-vertices of $HS_{\boldsymbol{next}_i(j),j}$*

- *an edge with one end-point in the set of* 0*-vertices of* $M_{i,j}$ *and other end-point in the set of* 3*-vertices of* $HS_{i,j}$
- *an edge with one end-point in the set of* 0*-vertices of* $M_{i,j}$ *and other end-point in the set of* 3*-vertices of* $VS_{i,\boldsymbol{next}_j(i)}$

Proof. We show that N must use at least one red edge which has one end-point in the set of 3-vertices of $\mathrm{M}_{i,j}$ and the other end-point in the set of 0-vertices of $\mathrm{VS}_{i,j}$. Analogous arguments hold for the other three secondary gadgets surrounding the main gadget $\mathrm{M}_{i,j}$ and hence we get the lower bound of four red edges as claimed (note that the vertex sets of the secondary gadgets are pairwise disjoint, and hence these edges are distinct).

We know by Claim 6.22 that $\mathrm{VS}_{i,j}$ is either right-oriented or left-oriented. Suppose $\mathrm{VS}_{i,j}$ is right-oriented. Also, by Lemma 6.4 and Claim 6.22, we know that the only base edges of $\mathrm{VS}_{i,j}$ picked in N are $\mathrm{VS}_{i,j}(0_{x_{i,j}}) \to \mathrm{VS}_{i,j}(1_{x_{i,j}}) \to \mathrm{VS}_{i,j}(2_{x_{i,j}}) \to \mathrm{VS}_{i,j}(3_{x_{i,j}})$ and the only connector edges of $\mathrm{VS}_{i,j}$ picked in E^* are $\mathrm{VS}_{i,j}(s_1) \to \mathrm{VS}_{i,j}(1_{x_{i,j}}), \mathrm{VS}_{i,j}(s_2) \to \mathrm{VS}_{i,j}(2_{x_{i,j}}), \mathrm{VS}_{i,j}(t_1) \leftarrow \mathrm{VS}_{i,j}(1_{x_{i,j}})$ and $\mathrm{VS}_{i,j}(t_2) \leftarrow \mathrm{VS}_{i,j}(2_{x_{i,j}})$. But there is a Type II demand pair whose target is $\mathrm{VS}_{i,j}(t_1)$: the path satisfying this demand pair has to enter $\mathrm{VS}_{i,j}$ through the vertex $\mathrm{VS}_{i,j}(0_{x_{i,j}})$. The only edges (which are not base or connector edges of $\mathrm{VS}_{i,j}$) incident on the 0-vertices of $\mathrm{VS}_{i,j}$ have their other end-point in the set of 3-vertices of $\mathrm{M}_{i,j}$. That is, N contains a red edge whose start vertex is a 3-vertex of $\mathrm{M}_{i,j}$ and end vertex is a 0-vertex of $\mathrm{VS}_{i,j}$. □

Claim 6.24. *For each* $1 \le i, j \le \ell$*, the edge set* N *must contain at least one orange edge of each of the following types:*

- *an outgoing edge from* a_i*,*
- *an incoming edge into* b_i*,*
- *an outgoing edge from* c_j*, and*
- *an incoming edge into* d_j*.*

Proof. Note that a_i is the source of two Type II demand pairs, viz. $(a_i, \mathrm{VS}_{i,\mathbf{next}_{\beta_i}(i)}(t_1))$ and $(a_i, \mathrm{VS}_{i,\mathbf{next}_{\beta_i}(i)}(t_2))$. Hence N must contain at least one outgoing edge from a_i. The other three statements follow similarly. □

We show now that we have no slack, i.e., the weight of N must be exactly B^*.

Claim 6.25. *The weight of* N *is exactly* B^**, and is inclusion-wise minimal.*

Proof. From Claim 6.22, we know that each gadget has a weight of B in N. Claim 6.23 says that each main gadget contributes at least 4 red edges which have exactly one end-point in this main gadget, and Claim 6.24 says that the orange edges (incident on border vertices) contribute at least 4ℓ to weight of N. Since all these edges are distinct, we have that the weight of N is at least $(2k + 2\ell) \cdot B + (2k + 2\ell) \cdot B + (2k + \ell) \cdot B + 4(\ell + 2k + \ell) = B^*$. Hence, the weight of N is exactly B^*, and it is minimal (under edge deletions) since no edges have zero weights. □

Consider a main gadget $\mathrm{M}_{i,j}$. It has four secondary gadgets surrounding it: $\mathrm{VS}_{i,j}$ below it, $\mathrm{VS}_{i,\mathbf{next}_j(i)}$ above it, $\mathrm{HS}_{i,j}$ to the left and $\mathrm{HS}_{\mathbf{next}_i(j),j}$ to the right. By Claim 6.22, these gadgets are represented by $x_{i,j}, x_{i,\mathbf{next}_j(i)}, y_{i,j}$, and $y_{\mathbf{next}_i(j),j}$ respectively. The main gadget $\mathrm{M}_{i,j}$ is represented by $(\lambda_{i,j}, \delta_{i,j})$.

Claim 6.26 (propagation)**.** *For every main gadget* $M_{i,j}$*, we have* $x_{i,j} = \lambda_{i,j} = x_{i,\boldsymbol{next}_j(i)}$ *and* $y_{i,j} = \delta_{i,j} = y_{\boldsymbol{next}_i(j),j}$*.*

Proof. Due to symmetry, it suffices to only argue that $x_{i,j} = \lambda_{i,j}$. Let us assume for the sake of contradiction that $x_{i,j} \neq \lambda_{i,j}$. We will now show that there is a vertex such that (1) there is exactly one edge adjacent to it in the edge set N and (2) it does not belong to any demand pair. Observe that removing its only adjacent edge from N does not effect the validity of the solution $\big(V(G^*), N\big)$ for the instance $(G^*, \mathcal{D})$ of bi-DSN. This contradicts Claim 6.25 which states that N is minimal.

From Claim 6.23 and Claim 6.25, it follows that N contains exactly one red edge, say e_1, which has one end-point in the set of 3-vertices of $\mathrm{M}_{i,j}$ and other end-point in set of 0-vertices of $\mathrm{VS}_{i,j}$. Also, N contains exactly one red edge, say e_2, which has one end-point in the set of 3-vertices of $\mathrm{M}_{i,j}$ and other end-point in set of 0-vertices of $\mathrm{HS}_{\mathbf{next}_i(j),j}$.

Observe that if e_1 does not have one endpoint at $\mathrm{VS}_{i,j}(0_{x_{i,j}})$, then the vertex $\mathrm{VS}_{i,j}(0_{x_{i,j}})$ is the desired vertex. Hence, suppose that one endpoint of the edge e_1 is $\mathrm{VS}_{i,j}(0_{x_{i,j}})$. The other endpoint of e_1 must be $\mathrm{M}_{i,j}(3_{x_{i,j},y})$ for some $y \in V_j$. Since $x_{i,j} \neq \lambda_{i,j}$, we have $\mathrm{M}_{i,j}(3_{x_{i,j},y}) \neq \mathrm{M}_{i,j}(3_{\lambda_{i,j},\delta_{i,j}})$. Suppose that one of the end-points of e_2 is $\mathrm{M}_{i,j}(3_{x',y'})$. Since $\mathrm{M}_{i,j}(3_{x_{i,j},y}) \neq \mathrm{M}_{i,j}(3_{\lambda_{i,j},\delta_{i,j}})$, at least one of the following must be true: $\mathrm{M}_{i,j}(3_{x_{i,j},y}) \neq \mathrm{M}_{i,j}(3_{x',y'})$ or $\mathrm{M}_{i,j}(3_{\lambda_{i,j},\delta_{i,j}}) \neq \mathrm{M}_{i,j}(3_{x',y'})$. If $\mathrm{M}_{i,j}(3_{x_{i,j},y}) \neq \mathrm{M}_{i,j}(3_{\lambda_{i,j},\delta_{i,j}})$, then $\mathrm{M}_{i,j}(3_{x_{i,j},y})$ is the desired vertex. Otherwise, if $\mathrm{M}_{i,j}(3_{\lambda_{i,j},\delta_{i,j}}) \neq \mathrm{M}_{i,j}(3_{x',y'})$, then $\mathrm{M}_{i,j}(3_{\lambda_{i,j},\delta_{i,j}})$ is the desired vertex. In all cases, we have found a vertex with desired properties; hence, we have arrived at a contradiction. □

By Claim 6.24 and Claim 6.25, for each $i \in [\ell]$ the edge set N has exactly one incoming edge into b_i. Define the function $\phi : V_H \to V_G$ given by $\phi(i) = z$ if the unique edge in N coming into b_i is from the vertex $\mathrm{VS}_{i,\alpha_i}(3_z)$. By construction of G^*, the vertical secondary gadget VS_{i,α_i} is a copy of $U_{|V_i|}$ and hence $\phi(i) \in V_i$. Similarly, by Claim 6.24 and Claim 6.25, for each $j \in [\ell]$ the edge set N has exactly one outgoing edge from c_j. Define the function $\psi : V_H \to V_G$ given by $\psi(j) = s$ if the unique edge in N coming out from c_j is into the vertex $\mathrm{HS}_{\alpha_j,j}(0_s)$. By construction of G^*, the horizontal secondary gadget $\mathrm{HS}_{\alpha_j,j}$ is a copy of $U_{|V_j|}$ and hence $\psi(j) \in V_j$.

By Claim 6.26, it follows that for each $1 \leq i, j \leq \ell$ such that $ij \in E_H^*$ we have $x_{i,j} = \lambda_{i,j} = x_{i,\mathbf{next}_j(i)}$ and $y_{i,j} = \delta_{i,j} = y_{\mathbf{next}_i(j),j}$ in addition to $(\lambda_{i,j}, \delta_{i,j}) \in E_{\{i,j\}}$ (by the definition of the main gadget).

Therefore, we have that $\phi(i) = x_{i,j}$ for each $j \in N'_H(i)$ and $\psi(j) = y_{i,j}$ for each $i \in N'_H(j)$. For each $i \in [\ell]$ the main gadget $\mathrm{M}_{i,i}$ is a copy of $U_{|E_{\{i,i\}}|}$. Hence, we have that $\phi(i) = x_{i,i} = y_{i,i} = \psi(i)$ for each $i \in [\ell]$. Now consider any $1 \leq i, j \leq \ell$ such that $ij \in E_H$. We know that $\mathrm{HS}_{i,j}, \mathrm{VS}_{i,j}$ are represented by $\phi(j), \phi(i)$ respectively. Since $\lambda_{i,j} = x_{i,j} = \phi(i), \delta_{i,j} = y_{i,j} = \phi(j)$ and $(\lambda_{i,j}, \delta_{i,j}) \in E_{i,j} \subseteq E_G$ it follows that $\phi(i)\phi(j) \in E_G$, i.e., the instance (G, H) of Colored Subgraph Isomorphism is a YES instance. This concludes the proof of Lemma 6.18. □

Finally we are ready to prove Theorem 1.7 which is restated below:

Theorem 1.7. *The* bi-DSN *problem is* W[1]*-hard parameterized by* k*. Moreover, under ETH there is no* $f(k) \cdot n^{o(k/\log k)}$ *time algorithm for* bi-DSN*, for any computable function* $f(k)$*.*

Proof. Each main gadget has $O(n^2)$ vertices and G^* has $2k + \ell = O(k)$ main gadgets, where $n = |V_G|$ and $k = |E_H|$. Each secondary gadget has $O(n)$ vertices and G^* has $2k + 2\ell = O(k)$ secondary gadgets. The number of border vertices in G^* is $4k$. Hence, the total number of vertices in the graph G^* is $O(n^2 k) = \mathrm{poly}(n, k)$. It is easy to see that G^* can be constructed in $\mathrm{poly}(n, k)$ time from a given instance (G, H) of Colored Subgraph Isomorphism. Combining the two directions from Lemma 6.17 and Lemma 6.18, we get a parameterized reduction from Colored Subgraph Isomorphism to an instance of bi-DSN with $|\mathcal{D}| = O(k)$ terminal pairs (Equation 5).

Hence, the W[1]-hardness of BI-DSN parameterized by the number k of demand pairs follows from the W[1]-hardness of COLORED SUBGRAPH ISOMORPHISM parameterized by $|E_H|$ (observe that the W[1]-hard problem MULTICOLORED CLIQUE [36, 69] is a special case of COLORED SUBGRAPH ISOMORPHISM when H is a clique). Marx [62, Corollary 6.3] showed that assuming ETH, the COLORED SUBGRAPH ISOMORPHISM problem cannot be solved in time $f(|E_H|) \cdot |V_G|^{o\left(|E_H|/\log|E_H|\right)}$ for any computable function f. Hence, it follows that under ETH, the BI-DSN cannot be solved in $f(k) \cdot n^{o(k/\log k)}$ time for any computable function f. □

Note that Theorem 1.7 also implies that there is no *efficient* parameterized approximation scheme, i.e. an algorithm computing a $(1+\varepsilon)$-approximate solution in time $f(k,\varepsilon) \cdot n^{O(1)}$ for some function f. This is because the hardness result holds for unweighted graphs in which the optimum solution has cost at most $B^* = 4\ell + (2k+2\ell)\cdot B + (2k+2\ell)\cdot B + (2k+\ell)\cdot(B+4) = O(k^5)$ if and only if the COLORED SUBGRAPH ISOMORPHISM instance is a YES instance. Consequently, if an efficient parameterized approximation scheme existed we could set ε to a value such that $(1+\varepsilon)\cdot B^* < B^* + 1$ with ε being a function of the parameter k independent of n. Every edge of our constructed graph G^* has weight at least 1, and hence an $(1+\epsilon)$-approximation is in fact forced to find a solution of cost at most B^*, i.e., it finds an optimum solution. Hence, we obtain the following corollary:

Corollary 6.27. *The* BI-DSN *problem has the following two lower bounds for efficient approximation schemes:*

- *Unless* FPT=W[1]*, there is no* $(1+\varepsilon)$*-approximation running in time* $f(\varepsilon,k)\cdot n^{O(1)}$ *for any function* f*.*
- *Under ETH, there is no* $(1+\varepsilon)$*-approximation running in time* $f(\varepsilon,k)\cdot n^{o(k/\log k)}$ *for any function* f*.*

6.4 NP-hardness and runtime lower bound for BI-SCSS

In this section we prove Theorem 1.11, which is restated below.

Theorem 1.11. *The* BI-SCSS *problem is NP-hard. Moreover, under ETH there is no* $2^{o(k)}\cdot n^{O(1)}$ *time algorithm for* BI-SCSS*.*

Proof. We reduce from the NP-hard HAMILTONIAN CYCLE problem. Given an undirected unweighted graph G on n vertices as an instance to HAMILTONIAN CYCLE, we construct a bidirected weighted complete graph H on the same vertex set as G as follows:

- If $\{u,v\}$ is an edge of G, then we set the weight of uv and vu in H to 1.
- If $\{u,v\}$ is not an edge of G, then we set the weight of uv and vu in H to 2.

Consider the BI-SCSS instance on H where every vertex is a terminal. We now show that G has a Hamiltonian cycle if and only if the BI-SCSS instance has a solution of cost n.

Suppose G has a Hamiltonian cycle. It corresponds to a directed cycle in H of cost n and is a feasible solution for the BI-SCSS instance. On the other hand, note that every BI-SCSS solution in H will have cost at least n, since every vertex has out-degree at least one in the solution. Hence, if there is a solution $N \subseteq H$ for the BI-SCSS instance of cost exactly n, then every vertex has out-degree exactly one in N, and each edge in N will have cost one. As N is strongly connected, this means N is a directed cycle. As this cycle consists of only edges of cost one, it follows that each edge in N is also an edge in G, i.e., the underlying undirected cycle $\overline{N}$ is a Hamiltonian cycle in G.

Finally, observe that we have shown above that BI-SCSS with $k = |V|$ terminals can solve the Hamiltonian cycle problem. It is known [19, Theorem 14.6] that under ETH the HAMILTONIAN CYCLE problem has no $2^{o(n)}\cdot n^{O(1)}$ algorithm. This immediately implies that BI-SCSS does not have an $2^{o(k)}\cdot n^{O(1)}$ algorithm under ETH. □

7 FPT inapproximability of SCSS and bi-DSN

The starting point of our inapproximability results are based on the recent parameterized inapproximability of Densest k-Subgraph from [9] (which in turn builds on a construction from [59]). To state the result precisely, let us first state the underlying assumption, the Gap Exponential Time Hypothesis (Gap-ETH). Note here that the version used here rules out not only deterministic but also randomized algorithms; this is needed for the inapproximability result of [9].

Hypothesis 7.1 ((Randomized) Gap-ETH [22, 60])**.** *There exists a constant $\delta > 0$ such that, given a* 3CNF *formula Φ on n variables, no (possibly randomized) $2^{o(n)}$-time algorithm can distinguish between the following two cases correctly with probability at least 2/3:*

- *Φ is satisfiable.*
- *Every assignment to the variables violates at least a δ-fraction of the clauses of Φ.*

Here we do not attempt to reason why Gap-ETH is a plausible assumption; for more detailed discussions on the topic, please refer to [9, 22]. For now, let us move on to state the inapproximability result from [9] that we need. Recall that, in the Densest k-Subgraph (DkS) problem [53], we are given an undirected graph $G = (V, E)$ and an integer k and we are asked to find a subset $S \subseteq V$ of size k that induces as many edges in G as possible. Chalermsook et al. [9] showed that, even when parameterized by k, the problem is hard to approximate to within a $k^{o(1)}$-factor, as stated more formally below.

Theorem 7.2 ([9, Lemma 5.21])**.** *Assuming randomized Gap-ETH, for any function $h(k) = o(1)$, there is no $f(k) \cdot n^{O(1)}$-time algorithm that, given a graph G on n vertices and an integer k, can distinguish between the following two cases:*

- *(YES) G contains at least one k-clique as a subgraph.*
- *(NO) Every k-subgraph of G contains less than $k^{-h(k)} \cdot \binom{k}{2}$ edges.*

Instead of working with DkS, it will be more convenient for us to work with a closely-related problem called Maximum Colored Subgraph Isomorphism, which can be defined as follows.

Maximum Colored Subgraph Isomorphism (MPSI)
Input: an instance Γ of MPSI consists of three components:
- an undirected graph $G = (V_G, E_G)$,
- a partition of vertex set V_G into disjoint subsets $V_1, \ldots, V_\ell$, and
- an undirected graph $H = (V_H = \{1, \ldots, \ell\}, E_H)$.

Goal: find an assignment $\phi : V_H \to V_G$ where $\phi(i) \in V_i$ for every $i \in [\ell]$ that maximizes the number of edges $ij \in E_H$ such that $\phi(i)\phi(j) \in E_G$.

It is worth noting that this problem is closely related to the so-called Label Cover problem in the hardness of approximation literature [66][8]. However, we choose the name Maximum Colored Subgraph Isomorphism as it is more compatible with the naming conventions earlier in Section 6; our new problem is simply an optimization version of Colored Subgraph Isomorphism defined in that section.

The graph H is sometimes referred to as the *supergraph* of Γ. Similarly, the vertices and edges of H are called *supernodes* and *superedges* of Γ. Moreover, the size of Γ is defined as $n = |V_G|$, the number of vertices of G. Additionally, for each assignment ϕ, we define its value

[8]The Label Cover problem may be viewed as a special case of MPSI with two additional constraints on the input: the graph G is bipartite, and every vertex on the left hand side of G has at most one edge to each partition V_i.

$\mathrm{val}(\phi)$ to be the fraction of superedges $ij \in E_H$ such that $\phi(i)\phi(j) \in E_G$; such superedges are said to be *covered* by ϕ. The objective of MPSI is now to find an assignment ϕ with maximum value. We denote the value of the optimal assignment by $\mathrm{val}(\Gamma)$, i.e., $\mathrm{val}(\Gamma) = \max_\phi \mathrm{val}(\phi)$.

For this problem, a hardness similar to that of Densest k-Subgraph can be shown:

Corollary 7.3. *Assuming randomized Gap-ETH, for any function $h(\ell) = o(1)$, there is no $f(\ell) \cdot n^{O(1)}$-time algorithm that, given an* MPSI *instance Γ of size n such that the supergraph H is a complete graph on ℓ supernodes, can distinguish between the following two cases:*

- *(YES)* $\mathrm{val}(\Gamma) = 1$.
- *(NO)* $\mathrm{val}(\Gamma) < \ell^{-h(\ell)}$.

The proof of Corollary 7.3 is rather simple, and follows the standard technique of using splitters. Nevertheless, for completeness, we give the full proof below. Before we proceed, we remark that our reduction is not the same as the Multicolored Clique hardness reduction from Clique [36, 69]. Recall that the reduction in [36, 69] simply makes each partition a copy of the original vertex set and add an edge between two new vertices iff there is an edge between the two corresponding vertices in the original graph. In this reduction, even in the NO case, we may take an edge (u, v) of the original graph and then select $\lceil k/2 \rceil$ copies of u together with $\lfloor k/2 \rfloor$ copies of v in the new graph. This means that the gap between the YES and the NO cases in [36, 69] can be at most two. Since we require a super constant gap in Corollary 7.3, we need a different reduction.

Definition 7.4. (**splitters**) Let $n \geq r \geq k$. An (n, k, r)-splitter is a family Λ of functions $[n] \to [r]$ such that for every subset $S \subseteq [n]$ of size k there is a function $\lambda \in \Lambda$ such that λ is injective on S.

The following constructions of special families of splitters are due to Alon et al. [2] and Naor et al. [67].

Theorem 7.5 ([2, 67])**.** *There exists a $2^{O(k)} \cdot n^{O(1)}$-time algorithm that takes in $n, k \in \mathbb{N}$ such that $n \geq k$ and outputs an (n, k, k)-splitter family of functions $\Lambda_{n,k}$ such that $|\Lambda_{n,k}| = 2^{O(k)} \cdot \log n$.*

Proof of Corollary 7.3. Suppose for the sake of contradiction that there exists an algorithm $\mathbb{B}$ that can solve the distinguishing problem stated in Corollary 7.3 in $f(\ell) \cdot n^{O(1)}$ time for some function f. We will use this to construct another algorithm $\mathbb{B}'$ that can solve the distinguishing problem stated in Theorem 7.2 in time $f'(k) \cdot n^{O(1)}$ for some function f', which will thereby violate Gap-ETH.

The algorithm $\mathbb{B}'$, on input (G, k), proceeds as follows. We assume w.l.o.g. that $V = [n]$. First, $\mathbb{B}'$ runs the algorithm from Theorem 7.5 on (n, k) to produce an (n, k, k)-splitter family of functions $\Lambda_{n,k}$. For each $\lambda \in \Lambda_{n,k}$, it creates a MPSI instance $\Gamma^\lambda = (G^\lambda, H^\lambda, V_1^\lambda \cup \cdots \cup V_k^\lambda)$ where

- the graph G^λ is simply the input graph G,
- for each $i \in [k]$, we set $V_i^\lambda = \lambda^{-1}(\{i\})$, and,
- the supergraph H^λ is simply the complete graph on $[k]$, i.e., $H^\lambda = ([k], \binom{[k]}{2})$.

Then, it runs the given algorithm $\mathbb{B}$ on Γ^λ. If $\mathbb{B}$ returns YES for some $\lambda \in \Lambda$, then $\mathbb{B}'$ returns YES. Otherwise, $\mathbb{B}'$ outputs NO.

It is obvious that the running time of $\mathbb{B}'$ is at most $O(2^{O(k)} f(k) \cdot n^{O(1)})$. Moreover, if G contains a k-clique, say $(v_1, \ldots, v_k)$, then by the properties of splitters we are guaranteed that there exists $\lambda \in \Lambda_{n,k}$ such that $\lambda(\{v_1, \ldots, v_k\}) = [k]$. Hence, the assignment $i \mapsto v_i$ covers all superedges in E_{H^λ}, implying that $\mathbb{B}$ indeed outputs YES on such Γ^λ. On the other hand, if every k-subgraph of G contains less than $k^{-h(k)} \cdot \binom{k}{2}$, then, for any $\lambda \in \Lambda_{n,k}$ and any assignment

ϕ of Γ^λ, $(\phi(1), \ldots, \phi(k))$ induces less than $k^{-h(k)} \cdot \binom{k}{2}$ edges in G. This also upper bounds the number of superedges covered by ϕ, which implies that Γ^λ is a NO instance of Corollary 7.3. Thus, in this case, $\mathbb{B}$ outputs NO on all Γ^λ's. In other words, $\mathbb{B}'$ can correctly distinguish the two cases in Theorem 7.2 in $O(2^{O(k)} f(k) \cdot n^{O(1)})$ time. This concludes our proof of Corollary 7.3. □

With the parameterized hardness of approximating MPSI ready, we can now prove our hardness results for SCSS and BI-DSN, starting with the former.

7.1 Strongly Connected Steiner Subgraph

Our proof of the parameterized inapproximability of SCSS is based on a reduction from MAXIMUM COLORED SUBGRAPH ISOMORPHISM whose properties are described below.

Lemma 7.6. *For every constant $\gamma > 0$, there exists a polynomial time reduction that, given an instance $\Gamma = (G, H, V_1 \cup \cdots \cup V_\ell)$ of* MPSI *where the supergraph H is a complete graph*[9]*, produces an instance $(G', \mathcal{T}')$ of* SCSS*, such that*

- *(completeness) if $\mathrm{val}(\Gamma) = 1$, then there exists a network $N' \subseteq G'$ of cost $2(1 + \gamma^{1/5})$ that is a solution of the instance $(G', \mathcal{T}')$ of* SCSS*,*
- *(soundness) if $\mathrm{val}(\Gamma) < \gamma$, then every network $N \subseteq G'$ that is a solution of the instance $(G', \mathcal{T}')$ of* SCSS *has cost more than $2(2 - 2\gamma^{1/5})$, and*
- *(parameter dependency) the number of terminals $|\mathcal{T}'|$ is ℓ^2.*

Proof. Assume without loss of generality that there is no edge in G between two vertices in the same set of the partition $V_1, \ldots, V_\ell$. We use the reduction of Guo et al. [42] with only a slight modification in that we use different edge weights. Our graph remains unchanged from the Guo et al. [42] reduction; here we copy the graph definition verbatim from [42]. We refer the reader to [42, Figure 3.1] for an illustration of the reduction. The vertex set V' of G' is $B \cup C \cup C' \cup D \cup D' \cup F$ where B, C, C', D, D', F are defined as follows:

- $B = \{b_i \mid i \in [\ell]\}$,
- $C = \{c_v \mid v \in V_G\}$,
- $C' = \{c'_v \mid v \in V_G\}$,
- $D = \{d_{u,v}, d_{v,u} \mid uv \in E_G\}$,
- $D' = \{d'_{u,v}, d'_{v,u} \mid uv \in E_G\}$, and,
- $F = \{f_{i,j} \mid 1 \leq i \neq j \leq \ell\}$.

We view the partition $V_G = V_1 \cup \cdots \cup V_\ell$ as a function $\lambda : V_G \to [\ell]$. The edge set E' of G' is $\mathcal{A} \cup \mathcal{A}' \cup \mathcal{B} \cup \mathcal{D} \cup \mathcal{D}' \cup \mathcal{H} \cup \mathcal{Z} \cup \mathcal{Z}'$ where

- $\mathcal{A} = \{\alpha_v = (b_{\lambda(v)}, c_v) \mid v \in V_G\}$,
- $\mathcal{A}' = \{\alpha'_v = (c'_v, b_{\lambda(v)}) \mid v \in V_G\}$,
- $\mathcal{B} = \{\beta_v = (c_v, c'_v) \mid v \in V_G\}$,
- $\mathcal{D} = \{\delta_{u,v} = (c'_u, d_{u,v}), \delta_{v,u} = (c'_v, d_{v,u}) \mid uv \in E_G\}$,
- $\mathcal{D}' = \{\delta'_{u,v} = (d'_{u,v}, c_v), \delta'_{v,u} = (d'_{v,u}, c_u) \mid uv \in E_G\}$,
- $\mathcal{H} = \{\epsilon_{u,v} = (d_{u,v}, d'_{u,v}), \epsilon_{v,u} = (d_{v,u}, d'_{v,u}) \mid uv \in E_G\}$,
- $\mathcal{Z} = \{\zeta_{u,v} = (f_{\lambda(u),\lambda(v)}, d_{u,v}), \zeta_{v,u} = (f_{\lambda(v),\lambda(u)}, d_{v,u}) \mid uv \in E_G\}$, and,
- $\mathcal{Z}' = \{\zeta'_{u,v} = (d'_{u,v}, f_{\lambda(u),\lambda(v)}), \zeta'_{v,u} = (d'_{v,u}, f_{\lambda(v),\lambda(u)}) \mid uv \in E_G\}$.

As for the weights, we give weight $2\gamma^{1/5}/\ell$ to β_v for every $v \in V_G$, and weight $1/\binom{\ell}{2}$ to $\epsilon_{u,v}$ and $\epsilon_{v,u}$ for every $uv \in E_G$; the rest of the edges have weight zero. As noted earlier, this is different from the weights assigned by Guo et al. [42]; they simply assigned the same weight to every

[9] Reductions in Lemma 7.6 and Lemma 8.2 can be trivially modified to work under a weaker assumption that H is regular (but not necessarily complete). However, we choose to only state the reductions when H is complete since this suffices for our purposes and the reductions are simpler to describe in this case.

edge. Finally, the terminals is defined as $\mathcal{T}' := B \cup F$. Observe that the number of terminals is $|\mathcal{T}'| = |B| + |F| = \ell + 2\binom{\ell}{2} = \ell^2$. We next move on to prove the completeness and soundness properties of the reduction.

(Completeness) The solution in the completeness case is exactly the same as the solution selected in [42]; we will repeat their argument here.

If $\mathrm{val}(\Gamma) = 1$, then there exists $(v_1, \ldots, v_\ell) \in V_1 \times \cdots \times V_\ell$ that induces an ℓ-clique. Consider the network $N' = \big(V(G'), E'\big)$ where $E' = \big\{\alpha_{v_i}, \alpha'_{v_i}, \beta_{v_i} \mid i \in [\ell]\big\} \cup \big\{\delta_{v_i,v_j}, \delta'_{v_i,v_j}, \epsilon_{v_i,v_j}, \zeta_{v_i,v_j}, \zeta'_{v_i,v_j} \mid 1 \leq i, j \leq \ell, i \neq j\big\}$. The total weight the network N' is $\ell \cdot \big(2\gamma^{1/5}/\ell\big) + 2\binom{\ell}{2} \cdot \Big(1/\binom{\ell}{2}\Big) = 2(1+\gamma^{1/5})$ as desired.

To see that the network N' is indeed a solution for the instance $(G', \mathcal{T}')$ of SCSS, observe that it suffices to show that, for every $1 \leq i \neq j \leq \ell$, $f_{i,j}$ is reachable from b_i and b_i is reachable from $f_{j,i}$. The former holds due to the path in N given by the following edges (in order) $\alpha_{v_i}, \beta_{v_i}, \delta_{v_i,v_j}, \epsilon_{v_i,v_j}, \zeta'_{v_i,v_j}$ whereas the latter holds due to the path in N given by the following edges (in order) $\zeta_{v_j,v_i}, \epsilon_{v_j,v_i}, \delta'_{v_j,v_i}, \beta_{v_i}, \alpha'_{v_i}$.

(Soundness) Our soundness proof will require a more subtle analysis than that of Guo et al. [42]. Again, we will prove by contrapositive. Suppose that there exists a network $N = \big(V(G'), E^*\big)$ of cost $\rho \leq 2(2 - 2\gamma^{1/5})$ which is a solution for the instance $(G', \mathcal{T}')$ of SCSS. For each $i \in [\ell]$, let $S_i \subseteq V_G$ denote the set of all vertices $v \in V_i$ such that β_{v_i} is included in E^*. Moreover, let $S = S_1 \cup \cdots \cup S_\ell$. Observe that, since each edge in $\mathcal{B}$ has weight $2\gamma^{1/5}/\ell$, we have

$$|S| \leq \frac{\rho}{(2\gamma^{1/5}/\ell)} \leq \frac{4}{(2\gamma^{1/5}/\ell)} \leq 2\gamma^{-1/5}\ell.$$

For every $1 \leq i \neq j \leq \ell$, let $\mathcal{H}_{i,j}$ denote the set of all $\epsilon_{u,v} \in E^*$ such that $u \in V_i$ and $v \in V_j$. First, we claim that, for every $1 \leq i \neq j \leq \ell$, $\mathcal{H}_{i,j} \neq \emptyset$. To see that this holds, consider the set $T_{i,j} = \{f_{i,j}\} \cup \{d'_{u,v} \mid u \in V_i, v \in V_j, uv \in E\}$. The only edges from outside $T_{i,j}$ coming into this set are those in $\mathcal{H}_{i,j}$. Since $\{f_{i,j}, b_i\} \subseteq \mathcal{T}'$, the vertex $f_{i,j}$ has to be reachable from b_i in N. However, since $b_i \notin T_{i,j}$, we can conclude that at least one edge in $\mathcal{H}_{i,j}$ must be selected in E^*.

Next, recall that each edge of the form $\epsilon_{u,v}$ has weight $1/\binom{\ell}{2}$. Since N has cost ρ, we have

$$\sum_{1 \leq i \neq j \leq \ell} |\mathcal{H}_{i,j}| \leq \binom{\ell}{2} \cdot \rho, \text{and thus}$$

$$\sum_{1 \leq i \neq j \leq \ell} (|\mathcal{H}_{i,j}| - 1) = \sum_{1 \leq i \neq j \leq \ell} |\mathcal{H}_{i,j}| - \ell(\ell-1) \leq \binom{\ell}{2} \cdot (\rho - 2)\,.$$

From $\mathcal{H}_{i,j} \neq \emptyset$ for every $1 \leq i \neq j \leq \ell$, the above inequality implies that for at least $\binom{\ell}{2} \cdot (4 - \rho) \geq 4\gamma^{1/5}\binom{\ell}{2}$ pairs of (i,j)'s we have $|\mathcal{H}_{i,j}| = 1$ $\big(\text{since } \rho \leq 2(2 - 2\gamma^{1/5})\big)$. Let $\mathcal{P}_{\text{unique}}$ be the set of all such pairs of (i,j)'s.

We will argue that a random assignment defined from picking one vertex from each S_i uniformly independently at random covers many superedges in expectation. To do this, we need to first show that, for many (i,j)'s, there exist $u \in S_i$ and $v \in S_j$ such that $uv \in E_G$. In fact, we can show this for every $(i,j) \in \mathcal{P}_{\text{unique}}$ as stated below.

Claim 7.7. *For every $(i,j) \in \mathcal{P}_{unique}$, there exists $u \in S_i$ and $v \in S_j$ such that $uv \in E_G$.*

Proof. Since $(i,j) \in \mathcal{P}_{\text{unique}}$, $\mathcal{H}_{i,j}$ contains only one element. Let this element be ϵ_{u_i,u_j}. We will prove that $u_i \in S_i$ and $u_j \in S_j$; note that this implies the claimed statement since $u_iu_j \in E_G$ by definition of ϵ_{u_i,u_j}.

To see that $u_i \in S_i$, consider the subset $C_{i,j} = \{f_{i,j}\} \cup \{d'_{u,v} \mid u \in V_i, v \in V_j, uv \in E\} \cup \{d_{u_i,u_j}\} \cup \{c'_{u_i}\}$. There are only two types of edges coming into $C_{i,j}$: (1) β_{u_i} and (2) $\epsilon_{u,v}$

where $u \in V_i, v \in V_j$ and $(u,v) \neq (u_i,u_j)$. Since $\mathcal{H}_{i,j} = \{\epsilon_{u_i,u_j}\}$, the edges of the latter types are not selected in E^*. Moreover, since $f_{i,j}$ is reachable from u_i, there must be at least one edge coming into $C_{i,j}$. As a result, β_{u_i} must be selected, which means that $u_i \in S_i$.

An analogous argument can be applied to u_j. Specifically, consider the subset $C'_{i,j} = \{f_{i,j}\} \cup \{d_{u,v} \mid u \in V_i, v \in V_j, uv \in E\} \cup \{d'_{u_i,u_j}\} \cup \{c_{u_j}\}$. There are only two types of edges coming out of C'_{ij}: (1) β_{u_j} and (2) $\epsilon_{u,v}$ where $u \in V_i, v \in V_j$ and $(u,v) \neq (u_i,u_j)$. Since $\mathcal{H}_{i,j} = \{\epsilon_{u_i,u_j}\}$, the edges of the latter types are not selected in E^*. Moreover, since u_j is reachable from $f_{i,j}$, there must be at least one edge coming out of $C_{i,j}$. As a result, β_{u_j} must be selected, which means that $u_j \in S_j$. ⌟

Now, let $\phi : V_H \to V_G$ be a random assignment where each $\phi(i)$ is chosen independently uniformly at random from S_i. By Claim 7.7, for every $(i,j) \in \mathcal{P}_{\text{unique}}$, there exists $u \in S_i$ and $v \in S_j$ such that $uv \in E_G$. This means that, for such (i,j), the probability that the superedge $ij \in E_H$ is covered is at least the probability that $\phi(i) = u$ and $\phi(j) = v$, which is equal to $\frac{1}{|S_i||S_j|}$. We now want a lower bound on the expected number of superdges covered by ϕ. For this, we use the following inequality which follows from a special case of Hölder's inequality for 3 variables[10]

$$\begin{aligned}
\Big(\sum_{(i,j)\in\mathcal{P}_{\text{unique}}} \frac{1}{|S_i||S_j|}\Big) \cdot \Big(\sum_{(i,j)\in\mathcal{P}_{\text{unique}}} |S_i|\Big) \cdot \Big(\sum_{(i,j)\in\mathcal{P}_{\text{unique}}} |S_j|\Big) & \\
&\geq \Big(\sum_{(i,j)\in\mathcal{P}_{\text{unique}}} \Big(\frac{1}{|S_i||S_j|}\Big)^{1/3} \cdot |S_i|^{1/3} \cdot |S_j|^{1/3}\Big)^3 \\
&= \Big(\sum_{(i,j)\in\mathcal{P}_{\text{unique}}} 1^{1/3}\Big)^3 \\
&= |\mathcal{P}_{\text{unique}}|^3 \qquad (1)
\end{aligned}$$

Hence, we have that the expected number of superedges covered by ϕ is at least

$$\begin{aligned}
\frac{1}{2} \cdot \sum_{(i,j)\in\mathcal{P}_{\text{unique}}} \frac{1}{|S_i||S_j|} &\geq \frac{1}{2} \cdot \frac{|\mathcal{P}_{\text{unique}}|^3}{\left(\sum_{(i,j)\in\mathcal{P}_{\text{unique}}} |S_i|\right)\left(\sum_{(i,j)\in\mathcal{P}_{\text{unique}}} |S_j|\right)} && \text{(from Equation (1))} \\
&\geq \frac{1}{2} \cdot \frac{|\mathcal{P}_{\text{unique}}|^3}{((\ell-1)|S|)^2} && \text{(since } S = \textstyle\bigcup_{i=1}^{\ell} S_i \text{, and } (i,j) \in \mathcal{P}_{\text{unique}} \Rightarrow i \neq j\text{)} \\
&\geq \frac{1}{2} \cdot \frac{\left(4\gamma^{1/5}\binom{\ell}{2}\right)^3}{(\ell-1)^2 \cdot \left(2\gamma^{-1/5}\ell\right)^2} \\
& && \text{(since } |\mathcal{P}_{\text{unique}}| \geq 4\gamma^{1/5}\tbinom{\ell}{2} \text{ and } |S| \leq 2\gamma^{-1/5}\ell\text{)} \\
&= \frac{1}{2} \cdot \frac{64\gamma^{3/5} \cdot \binom{\ell}{2}^3}{(\ell-1)^2 \cdot 4\gamma^{-2/5} \cdot \ell^2} \\
&\geq 2\gamma \cdot \binom{\ell}{2},
\end{aligned}$$

where Note that the factor $1/2$ comes from the fact that we may double count each edge for both $(i,j),(j,i)$. Hence, there exists an assignment of Γ with value at least 2γ, which implies that $\text{val}(\Gamma) \geq 2\gamma \geq \gamma$. □

We can now easily prove Theorem 1.9 by combining Lemma 7.6 and Corollary 7.3.

[10] $(\sum_{r=1}^n a_r^3)(\sum_{r=1}^n b_r^3)(\sum_{r=1}^n c_r^3) \geq (\sum_{r=1}^n a_r b_r c_r)^3$

Proof of Theorem 1.9. We again prove by contrapositive. Suppose that, for some constant $\varepsilon > 0$ and for some function $f(k)$ independent of n, there exists an $f(k) \cdot n^{O(1)}$-time $(2-\varepsilon)$-approximation algorithm for SCSS. Let us call this algorithm $\mathbb{A}$.

It is easy to see that there exists a sufficiently small $\gamma^* = \gamma^*(\varepsilon)$ such that $\frac{2-2\gamma^{*1/5}}{1+\gamma^{*1/5}} \geq (2-\varepsilon)$. We create an algorithm $\mathbb{B}$ that can distinguish between the two cases of Corollary 7.3 with $h(\ell) = \log(1/\gamma^*)/\log \ell$. Our new algorithm $\mathbb{B}$ works as follows. Given an instance $(G, H, V_1 \cup \cdots \cup V_\ell)$ of MPSI where H is a complete graph, $\mathbb{B}$ uses the reduction from Lemma 7.6 to create an SCSS instance on the graph G' with $k = \ell^2$ terminals. $\mathbb{B}$ then runs $\mathbb{A}$ on this instance; if $\mathbb{A}$ returns a solution N of cost at most $2(2 - 2\gamma^{*1/5})$, then $\mathbb{B}$ returns YES. Otherwise, $\mathbb{B}$ returns NO.

To see that algorithm $\mathbb{B}$ can indeed distinguish between the YES and NO cases, first observe that, in the YES case, Lemma 7.6 guarantees that the optimal solution has cost at most $2(1 + \gamma^{*1/5})$. Since $\mathbb{A}$ is a $(2-\varepsilon)$-approximation algorithm, it returns a solution of cost at most $2(1 + \gamma^{*1/5}) \cdot (2 - \varepsilon) \leq 2(2 - 2\gamma^{*1/5})$ where the inequality comes from our choice of γ^*; this means that $\mathbb{B}$ outputs YES. On the other hand, if $(G, H, V_1 \cup \cdots \cup V_\ell)$ is a NO instance, then the soundness property of Lemma 7.6 guarantees that the optimal solution in G' has cost more than $2(2 - 2\gamma^{*1/5})$, which implies that $\mathbb{B}$ outputs NO.

Finally, observe that the running time of $\mathbb{B}$ is $f(\ell^2) \cdot n^{O(1)}$ and that $h(\ell) = o(1)$. Hence, from Corollary 7.3, randomized Gap-ETH breaks. □

7.2 Directed Steiner Network on Bidirected Graphs

We will next prove our inapproximability result for bi-DSN. For this result, we will need a slightly more specific hardness of approximation for Maximum Colored Subgraph Isomorphism where every supernode has bounded degree. This bounded degree version of MPSI is defined below.

t-Bounded Degree Maximum Colored Subgraph Isomorphism $(\text{MPSI}(t))$
Input: an instance Γ of MPSI(t) consists of three components:
- an undirected graph $G = (V_G, E_G)$,
- a partition of vertex set V_G into disjoint subsets $V_1, \ldots, V_\ell$, and
- an undirected graph $H = (V_H = \{1, \ldots, \ell\}, E_H)$ such that each vertex of H has degree at most t.

Goal: find an assignment $\phi : V_H \to V_G$ where $\phi(i) \in V_i$ for every $i \in [\ell]$ that maximizes the number of edges $ij \in E_H$ such that $\phi(i)\phi(j) \in E_G$.

Lokshtanov et al. [58] gave the following reduction from (unbounded degree) MPSI to the bounded degree version of the problem. We remark here that their reduction uses standard technique of sparsification via expanders, and similar reductions have been presented before in literature (see e.g. [21]).

Lemma 7.8 ([58])**.** *For every $\varepsilon > 0$, there exists $\varepsilon' > 0$ and a polynomial time reduction that, given an instance $\Gamma = (G, H, V_1 \cup \cdots \cup V_\ell)$ of* MPSI*, produces an instance $\Gamma' = (G', H', V_1 \cup \cdots \cup V_{\ell'})$ of* MPSI*(4) such that*
- *(YES) if* $\text{val}(\Gamma) = 1$*, then* $\text{val}(\Gamma') = 1$*,*
- *(NO) if* $\text{val}(\Gamma) < 1 - \varepsilon$*, then* $\text{val}(\Gamma') < 1 - \varepsilon'$*, and*
- *(parameter dependency)* $\ell' = O(\ell^2)$*.*

Combined this with the parameterized inapproximability of Corollary 7.3, we can immediately conclude that the bounded degree version of MPSI is also hard to approximate, even for parameterized algorithms:

Corollary 7.9. *Assuming randomized Gap-ETH, for some $\varepsilon > 0$, there is no $f(\ell) \cdot n^{O(1)}$-time algorithm that, given a* MPSI*(4) instance $\Gamma = (G, H, V_1 \cup \cdots \cup V_\ell)$ of size n, can distinguish between the following two cases:*

- *(YES)* $\mathrm{val}(\Gamma) = 1$.
- *(NO)* $\mathrm{val}(\Gamma) < 1 - \varepsilon$.

We are now ready to state the main lemma of this subsection, which provides a reduction from bounded degree MPSI to bi-DSN:

Lemma 7.10. *For every constant $\varepsilon > 0$ and any $d \in \mathbb{N}$, there exists a polynomial time reduction that, given an instance $\Gamma = (G, H, V_1 \cup \cdots \cup V_\ell)$ of* MPSI*(d), produces an instance $(G', \mathcal{D}')$ of* bi-DSN *and $B^* \in \mathbb{N}$ such that*

- *(completeness) if $\mathrm{val}(\Gamma) = 1$, then there exists a network $N \subseteq G'$ of cost B^* that satisfies all demands,*
- *(soundness) if $\mathrm{val}(\Gamma) < 1 - \varepsilon$, then every network $N \subseteq G'$ that satisfies all demands has cost more than $(1 + \frac{\varepsilon}{11840d})B^*$, and*
- *(parameter dependency) The number of demand pairs $|\mathcal{D}'|$ is $O(\ell)$.*

Before we proceed to prove the above lemma, let us note that Theorem 1.5 follows immediately from Corollary 7.9 and Lemma 7.10.

The construction for Lemma 7.10 is exactly the same as that in Section 6.3 with only one exception: each gadget will now be a copy of the uniqueness gadget from Section 6.1 with $M = 13$ (instead of $M = k^4$ used before). Again, it is clear that the number of demand pairs is $O(k + \ell) = O(\ell)$ where k is the number of superedges, i.e., $k = |E_H|$.

Let $B = 7M = 91$ and $B^* = 4\ell + (2k + 2\ell) \cdot B + (2k + 2\ell) \cdot B + (2k + \ell) \cdot (B + 4)$. Note that we can simplify the value of B^* as follows:

$$B^* = 4\ell + (6k + 5\ell) \cdot B + 4(2k + \ell) = 8(k + \ell) + 91(6k + 5\ell) = 554k + 463\ell \tag{7}$$

It is not hard to see that, in the completeness case, the solution used in Section 6.3 still works, and that it has cost exactly B^* as desired. Hence, we are only left to show the soundness of the reduction.

We proceed to prove the soundness. Again, we will prove our soundness by contrapositive. Suppose that there exists a network $\big(V(G'), N\big)$ of cost $\rho < (1 + \beta)B^*$ where $\beta = \frac{\varepsilon}{11840d}$ that satisfies all the demand pairs. We will also assume without loss of generality that the edge set N is *inclusion-wise minimal*, i.e., that if we remove any edge from N, then at least one demand pair must be unsatisfied.

Since our underlying graph and the demand pairs are exactly the same as those from Section 6.3, the restriction of N into each gadget must again satisfy the in-out property (similar to Claim 6.19, Claim 6.20, and Claim 6.21) as stated below:

Lemma 7.11. *For any $j \in [\ell], i \in N'_H(j)$ the edges of N which have both end-points in the horizontal secondary gadget $HS_{i,j}$ satisfies the in-out property. Hence, $HS_{i,j}$ uses up weight of at least B from the budget.*

Lemma 7.12. *For any $i \in [\ell], j \in N'_H(i)$ the edges of N which have both end-points in the vertical secondary gadget $VS_{i,j}$ satisfies the in-out property. Hence, $VS_{i,j}$ uses up weight of at least B from the budget.*

Lemma 7.13. *For every i, j such that $ij \in E^*_H$ the edges of N which have both end-points the main gadget $M_{i,j}$ satisfies the in-out property. Hence, $M_{i,j}$ uses up weight of at least B from the budget.*

We say that a gadget is *tight* if N restricted to the gadget has cost exactly B. Recall that the first step of the proof of the reverse direction[11] of Theorem 1.7 was to observe that every gadget must be tight; this was true because the value M over there was set so large that even an excess of B was already more than the total cost of all red edges. However, this is not true in our modified construction anymore as we choose $M = 13$. Fortunately for us, we will still be able to show that all but a small fraction of the gadgets are tight.

To prove such a bound, first recall that Claim 6.23 (used in the proof of Theorem 1.7) exactly shows that if a main gadget and all its four surrounding secondary gadgets are tight, then the edge set N must contain at least four red edges with exactly one-end point in the main gadget. We restate this formally as follows (proof is omitted since it is exactly the same as that of Claim 6.23)

Lemma 7.14. *For each $1 \leq i, j \leq \ell$ such that $ij \in E^*_H$, if the main gadget $M_{i,j}$ and the four secondary gadgets surrounding it $\left(VS_{i,j}, VS_{i,\boldsymbol{next}_j(i)}, HS_{i,j}\right.$ and $\left.HS_{\boldsymbol{next}_i(j),j}\right)$ are tight, then the edge set N must contain at least one red edge of each of the following types:*

(a) an edge with one end-point in the set of 3-vertices of $M_{i,j}$ and the other end-point in the set of 0-vertices of $VS_{i,j}$,

(b) an edge with one end-point in the set of 0-vertices of $M_{i,j}$ and the other end-point in the set of 3-vertices of $VS_{i,\boldsymbol{next}_j(i)}$,

(c) an edge with one end-point in the set of 0-vertices of $M_{i,j}$ and the other end-point in the set of 3-vertices of $HS_{i,j}$, and

(d) an edge with one end-point in the set of 3-vertices of $M_{i,j}$ and the other end-point in the set of 0-vertices of $HS_{\boldsymbol{next}_i(j),j}$.

Next we restate Claim 6.24, which we can use here since it only uses the fact that the edge set N is such that $\left(V(G'), N\right)$ is a solution for the instance $(G', \mathcal{D}')$ of BI-DSN:

Claim 6.24. *For each $1 \leq i, j \leq \ell$, the edge set N must contain at least one orange edge of each of the following types:*

- *an outgoing edge from a_i,*
- *an incoming edge into b_i,*
- *an outgoing edge from c_j, and*
- *an incoming edge into d_j.*

We are now ready to prove a bound on the number of non-tight gadgets. The key idea here is that, while having a non-tight gadget may help "save" the number of required red edges from Claim 6.23, this saving is still smaller than the excess cost of M. Hence, if there are too many non-tight gadgets, then the cost of N must be much more than the minimum possible cost of B^*, which would contradict our assumption that the cost of N is at most $(1 + \beta)B^*$.

In addition to the bound on the number of non-tight gadgets, we will be able to give an upper bound on the number of main gadgets with at least five red edges touching them; again, this is just because these edges adds to the minimum possible cost B^*.

Lemma 7.15. *There are at most $\beta \cdot B^*$ non-tight gadgets. Moreover, there are at most $\beta \cdot B^*$ main gadgets $M_{i,j}$ such that there are at least five red edges with at least one endpoint in $M_{i,j}$.*

Proof. Let X be the number of non-tight gadgets and Y be the number of main gadget $\mathrm{M}_{i,j}$ such that there are at least five red edges with at least one endpoint in $\mathrm{M}_{i,j}$.

We can lower bound the cost of the solution N as follows.

[11] If the instance $(G^*, \mathcal{D})$ of BI-DSN has a solution of weight $\leq B^*$ then the instance (G, H) of COLORED SUBGRAPH ISOMORPHISM has a solution

- From Claim 6.24, at least 4ℓ orange edges must be selected.
- Since there is a total of $6k + 5\ell$ gadgets (including both main and secondary gadgets), at least $(6k + 5\ell - X)$ of these gadgets are tight. These tight gadgets use up weight of $(6k + 5\ell - X)B$ from the budget. Further, Lemma 6.4, together with Lemma 7.11, Lemma 7.12 and Lemma 7.13, implies that each of the X non-tight gadgets uses up weight at least $8M = (B + M)$. Hence, in total, the weight of edges of N whose both endpoints are from the same gadget is at least $(6k + 5\ell)B + XM$.
- Let us divide the main gadgets $\mathrm{M}_{i,j}$ into three groups based on the number of red edges touching them: (1) there are at most three such edges, (2) there are at least five such edges and (3) there are exactly four such edges.
  From Lemma 7.14, each gadget of type (1) must either be non-tight or be adjacent to at least one non-tight secondary gadgets. Since there are only X non-tight gadgets and each secondary gadget is adjacent to at most two main gadgets, the number of main gadgets of type (1) is at most $X + 2X = 3X$. Recall also that we assume that the number of main gadgets of type (2) is Y. As a result, the number of red edges is at least $5Y + 4(2k + \ell - 3X - Y) = Y - 12X + 4(2k + \ell)$.

We can conclude that in total the cost of N must be at least

$$\begin{aligned}
&4\ell + \Big((6k + 5\ell)B + XM\Big) + \Big(Y - 12X + 4(2k + \ell)\Big) \\
&\quad = 4\ell + \Big((6k + 5\ell)B + 13X\Big) + \Big(Y - 12X + 4(2k + \ell)\Big) && \text{(since } M = 13\text{)} \\
&\quad = \Big(4\ell + (6k + 5\ell)B + 4(2k + \ell)\Big) + Y + X \\
&\quad = \Big(4\ell + (2k + 2\ell)B + (2k + 2\ell)B + (2k + \ell)(B + 4)\Big) + Y + X \\
&\quad = B^* + Y + X
\end{aligned}$$

where the last equality follows because $B^* = 4\ell + (2k + 2\ell)B + (2k + 2\ell)B + (2k + \ell)(B + 4)$. Since we assume that the total cost of N is at most $(1 + \beta)B^*$, we have $X, Y \leq \beta \cdot B^*$ as desired. □

We will next use the above bound to help us find a solution $\phi : V_H \to V_G$ to the MPSI(d) instance Γ. Unlike in the proof of Theorem 1.7 where the network N canonically gives $\phi(i)$ for every $i \in [\ell]$, this will only be true for "good" i which is defined below.

Definition 7.16. A main gadget $\mathrm{M}_{i,j}$ is *good* if the gadget itself and all its surrounding secondary gadgets ($\mathrm{VS}_{i,j}$, $\mathrm{VS}_{i,\mathbf{next}_j(i)}$, $\mathrm{HS}_{i,j}$, and $\mathrm{HS}_{\mathbf{next}_i(j),j}$) are tight and there are exactly four red edges with one endpoint in $\mathrm{M}_{i,j}$. We call a main gadget $\mathrm{M}_{i,j}$ *bad* if it is not good.

Furthermore, $i \in [\ell]$ is said to be *good* if $\mathrm{M}_{i,j}$ and $\mathrm{M}_{j,i}$ are good for every $ij \in E_H^*$. Similarly, we say that $i \in [\ell]$ is *bad* if it is not good.

We now set up some notation regarding representation of each tight gadget. Note that, while in Section 6.3 every gadget is tight and hence the notation there applied for all gadgets, the following notation is only well-defined for tight gadgets in our proof:

- For each $j \in [\ell]$ and each $i \in N'_H(j)$, if the horizontal secondary gadget $\mathrm{HS}_{i,j}$ is tight then $\mathrm{HS}_{i,j}$ is represented (Definition 6.3) by some $y_{i,j} \in V_j$,
- For each $i \in [\ell]$ and each $j \in N'_H(i)$, if the vertical secondary gadget $\mathrm{VS}_{i,j}$ is tight then $\mathrm{VS}_{i,j}$ represented (Definition 6.3) by some $x_{i,j} \in V_i$,
- For each $ij \in E_H^*$, if the main gadget $\mathrm{M}_{i,j}$ is tight then $\mathrm{M}_{i,j}$ is represented (Definition 6.3) by some $(\lambda_{i,j}, \delta_{i,j}) \in E_{i,j}$.

Consider the assignment $\phi : V_H \to V_G$ defined as follows: for each $i \in [\ell]$

$$\phi(i) = \begin{cases} \lambda_{i,i} & \text{if } i \text{ is good} \\ \text{any arbitrarily chosen vertex from } V_i & \text{otherwise} \end{cases}$$

The remaining argument consists of two parts. First, we will show that ϕ covers every superedge $ij \in E_H$ such that both i, j are good. Then, we will argue that only a small fraction of $i \in [\ell]$ is bad. Combining these two parts completes our proof.

To show that ϕ satisfies all superedges whose endpoints are both good, we first argue, similar to Claim 6.26, that good gadgets allow us to propagate equality of representations, as stated formally below.

Lemma 7.17. *For every good main gadget $M_{i,j}$, we have $x_{i,j} = \lambda_{i,j} = x_{i,\boldsymbol{next}_j(i)}$ and $y_{i,j} = \delta_{i,j} = y_{\boldsymbol{next}_j(i),j}$.*

Proof. Due to symmetry, it suffices to only argue that $x_{i,j} = \lambda_{i,j}$. Let us assume for the sake of contradiction that $x_{i,j} \neq \lambda_{i,j}$. We will argue that there is a vertex such that (1) there is exactly one edge adjacent to it from the network N and (2) it does not belong to any demand pair. Observe that removing its only adjacent edge from N does not affect the validity of the solution. This contradicts our assumption that N is minimal.

From Lemma 7.14 and from our assumption that there are exactly four red edges with one endpoint in $\mathrm{M}_{i,j}$, there must be exactly one red edge from each of the following types:

(a) an edge with one end-point in the set of 3-vertices of $\mathrm{M}_{i,j}$ and the other end-point in the set of 0-vertices of $\mathrm{VS}_{i,j}$,
(b) an edge with one end-point in the set of 0-vertices of $\mathrm{M}_{i,j}$ and the other end-point in the set of 3-vertices of $\mathrm{VS}_{i,\mathbf{next}_j(i)}$,
(c) an edge with one end-point in the set of 0-vertices of $\mathrm{M}_{i,j}$ and the other end-point in the set of 3-vertices of $\mathrm{HS}_{i,j}$, and
(d) an edge with one end-point in the set of 3-vertices of $\mathrm{M}_{i,j}$ and the other end-point in the set of 0-vertices of $\mathrm{HS}_{\mathbf{next}_i(j),j}$.

Observe that if the edge of type (a) does not have one endpoint at $\mathrm{VS}_{i,j}(0_{x_{i,j}})$, then the vertex $\mathrm{VS}_{i,j}(0_{x_{i,j}})$ is the desired vertex.

Now, suppose that one endpoint of the edge of type (a) is $\mathrm{VS}_{i,j}(0_{x_{i,j}})$. The other endpoint must be $\mathrm{M}_{i,j}(3_{x_{i,j},y})$ for some $y \in V_j$. Since $x_{i,j} \neq \lambda_{i,j}$, we have $\mathrm{M}_{i,j}(3_{x_{i,j},y}) \neq \mathrm{M}_{i,j}(3_{\lambda_{i,j},\delta_{i,j}})$. Consider the edge of type (d); suppose that one of its endpoint is $\mathrm{M}_{i,j}(3_{x',y'})$. Since $\mathrm{M}_{i,j}(3_{x_{i,j},y}) \neq \mathrm{M}_{i,j}(3_{\lambda_{i,j},\delta_{i,j}})$, at least one of the following must be true: $\mathrm{M}_{i,j}(3_{x_{i,j},y}) \neq \mathrm{M}_{i,j}(3_{x',y'})$ or $\mathrm{M}_{i,j}(3_{\lambda_{i,j},\delta_{i,j}}) \neq \mathrm{M}_{i,j}(3_{x',y'})$.

If $\mathrm{M}_{i,j}(3_{x_{i,j},y}) \neq \mathrm{M}_{i,j}(3_{\lambda_{i,j},\delta_{i,j}})$, then $\mathrm{M}_{i,j}(3_{x_{i,j},y})$ is the desired vertex. Otherwise, if $\mathrm{M}_{i,j}(3_{\lambda_{i,j},\delta_{i,j}}) \neq \mathrm{M}_{i,j}(3_{x',y'})$, then $\mathrm{M}_{i,j}(3_{\lambda_{i,j},\delta_{i,j}})$ is the desired vertex.

In all cases, we have found a vertex with desired properties, and hence we have arrived at a contradiction. □

Our main claim now follows almost immediately from Lemma 7.17.

Lemma 7.18. *The mapping ϕ covers every $ij \in E_H$ such that both i, j are good.*

Proof. Consider any such superedge $ij \in E_H$. Let $j_1 < j_2 < \cdots < j_p$ be all elements of $N'_H(i)$ and $i_1 < i_2 < \cdots < i_q$ be all elements of $N'_H(j)$, Lemma 7.17 implies that

$$x_{i,j_1} = \lambda_{i,j_1} = x_{i,j_2} = \cdots = \lambda_{i,j_p},$$

and

$$y_{i_1,j} = \delta_{i_1,j} = y_{i_2,j} = \cdots = \delta_{i_q,j}.$$

Since $j \in N'_H(i)$ and $i \in N'_H(j)$, the above inequalities imply that $\lambda_{i,j} = \lambda_{i,i}$ and $\delta_{i,j} = \delta_{j,j}$. Furthermore, observe that $(\lambda_{j,j}, \delta_{j,j}) \in E_{j,j}$, meaning that $\delta_{j,j} = \lambda_{j,j}$.

Recall that we set $\phi(i) = \lambda_{i,i}$ and $\phi(j) = \lambda_{j,j}$. This means that $(\phi(i), \phi(j)) = (\lambda_{i,j}, \delta_{i,j})$ which must be in $E_{i,j}$. In other words, ij is covered by ϕ. □

For the second part, let us first argue an upper bound on the number of bad main gadgets. Observe that each bad main gadget $\mathrm{M}_{i,j}$ must satisfy at least one of the three following conditions: (1) $\mathrm{M}_{i,j}$ is not tight, (2) one of its surrounding secondary gadgets is not tight, or (3) there are at least five red edges with one endpoint in $\mathrm{M}_{i,j}$. Lemma 7.15 implies that there are at most $\beta \cdot B^*$, $2\beta \cdot B^*$, and $\beta \cdot B^*$ main gadgets that satisfy (1), (2), and (3) respectively (recall that each secondary gadget has edges to at most two main gadgets). Hence, in total, there are at most $4\beta \cdot B^*$ bad main gadgets. Since, for each bad $i \in [\ell]$, there must exist some $j \in N'_H(i)$ such that $\mathrm{M}_{i,j}$ or $\mathrm{M}_{j,i}$ is a bad gadget, there can be at most $8\beta \cdot B^*$ bad $i \in [\ell]$.

Due to our bounded degree assumption on H, there can be at most $8d\beta \cdot B^*$ superedges $ij \in E_H$ such that at least one of i, j is bad. As a result, ϕ satisfies all but $8\beta d \cdot B^*$ superedges. As a result, we have

$$\begin{aligned}
\mathrm{val}(\Gamma) &\geq 1 - \frac{8\beta d \cdot B^*}{k} \\
&= 1 - \frac{8\beta d \cdot (554k + 463\ell)}{k} && \text{(since } B^* = 554k + 463\ell \text{ from Equation 7)} \\
&\geq 1 - 8\beta d(554 + 926) && \text{(since } k \geq \ell/2\text{)} \\
&= 1 - \varepsilon && \text{(since } \beta = \tfrac{\varepsilon}{11840d}\text{)}
\end{aligned}$$

where the second inequality comes from the fact that we can assume without loss of generality that the supergraph H does not contain any isolated vertex. This concludes the proof of Lemma 7.10.

8 A Reduction from MPSI to DSN

In a previous version [16] of this manuscript, we had provided the following $k^{o(1)}$-factor inapproximability result for DSN:

Theorem 8.1. *Under Gap-ETH, for any function $g(k) = o(1)$ and any $f(k)$ independent of n, there is no $f(k) \cdot n^{O(1)}$ time algorithm that computes an $k^{g(k)}$-approximation for* DSN.

Theorem 8.1 has since been subsumed by [23] which shows an improved hardness of $k^{1/4-o(1)}$-approximation for DSN under the same assumption of Gap-ETH. At the heart of our proof of Theorem 8.1 is the following lemma which provides a gap-preserving FPT reduction from Maximum Colored Subgraph Isomorphism to DSN.

Lemma 8.2. *There exists a polynomial time reduction that, given an instance $\Gamma = (G, H, V_1 \cup \cdots \cup V_\ell)$ of MPSI where the supergraph H is a complete graph, produces an instance of* DSN *with a graph G' and k demand pairs, such that*

- *(completeness) if $\mathrm{val}(\Gamma) = 1$, there is a network $N \subseteq G'$ of cost 1 that satisfies all demands,*
- *(soundness) for any $\gamma > 0$ (possibly depending on k), if $\mathrm{val}(\Gamma) < \gamma$, then every network $N \subseteq G'$ that satisfies all demands has cost more than $1/\sqrt{4\gamma}$, and*

- *(parameter dependency)* $k = \ell^2 - \ell$.

The proof of Theorem 8.1 follows immediately from Lemma 8.2 and Corollary 7.3:

Proof of Theorem 8.1. We prove by using the contrapositive. Suppose that, for some function $g(k) = o(1)$ and for some function $f(k)$ independent of n, there exists an $f(k) \cdot n^{o(1)}$ time $k^{g(k)}$-approximation algorithm for DSN. Let us call this algorithm $\mathbb{A}$.

We now design an algorithm $\mathbb{B}$ that can distinguish between the two cases of Corollary 7.3 with $h(\ell) = 4g(\ell^2-\ell)+\frac{4}{\log_2 \ell}$. The algorithm $\mathbb{B}$ works as follows: given an instance $(G, \ell, V_1 \cup V_2 \cup \ldots \cup V_\ell)$ of Maximum Colored Subgraph Isomorphism where the supergraph H is the complete graph on ℓ nodes, $\mathbb{B}$ uses the reduction from Lemma 8.2 to create a DSN instance on the graph G' with $k = \ell^2 - \ell$ demands. $\mathbb{B}$ then runs $\mathbb{A}$ on this instance; if $\mathbb{A}$ returns a solution N of cost at most $k^{g(k)} = (\ell^2 - \ell)^{g(\ell^2-\ell)}$, then $\mathbb{B}$ returns YES. Otherwise, $\mathbb{B}$ returns NO.

To see that algorithm $\mathbb{B}$ can indeed distinguish between the YES and NO cases, first observe that,in the YES case, Lemma 8.2 guarantees that the optimal solution is of cost at most 1. Since $\mathbb{A}$ is an $k^{g(k)}$-approximation algorithm, it returns a solution of cost at most $k^{g(k)} = (\ell^2 - \ell)^{g(\ell^2-\ell)}$, meaning that $\mathbb{B}$ outputs YES. On the other hand, if $(G, \ell, V_1 \cup V_2 \cup \ldots \cup V_\ell)$ is a NO instance, then the soundness property of Lemma 8.2 guarantees that the optimal solution in G' has cost more than $\frac{1}{\sqrt{4\ell^{-h(\ell)}}}$. In this case, $\mathbb{B}$ also outputs NO since we have

$$\begin{aligned}
\frac{1}{\sqrt{4\ell^{-h(\ell)}}} &= \frac{\sqrt{\ell^{h(\ell)}}}{2} \\
&= \frac{1}{2} \cdot \ell^{2g(\ell^2-\ell)+\frac{2}{\log_2 \ell}} && \left(\text{since } h(\ell) = 4g(\ell^2 - \ell) + \tfrac{4}{\log_2 \ell}\right) \\
&= \frac{1}{2} \cdot (\ell^2)^{g(\ell^2-\ell)} \cdot \ell^{\frac{2}{\log_2 \ell}} \\
&> \frac{1}{2} \cdot (\ell^2 - \ell)^{g(\ell^2-\ell)} \cdot 4 && \left(\text{since } \ell^2 > (\ell^2 - \ell) \text{ for each } \ell \geq 1 \text{ and } \ell^{\frac{2}{\log_2 \ell}} = 4\right) \\
&> (\ell^2 - \ell)^{g(\ell^2-\ell)}
\end{aligned}$$

Finally, observe that the running time of $\mathbb{B}$ is bounded by the running time of $\mathbb{A}$ plus the $n^{O(1)}$ time needed for the reduction of Lemma 8.2. Since $k = \ell^2 - \ell$ and the running time of $\mathbb{A}$ is $f(k) \cdot n^{O(1)}$, it follows that the running time of B can be expressed as $f'(\ell) \cdot n^{O(1)}$ for some function f'. Moreover, since $g(k) = o(1)$ it also follows that $h(\ell) = o(1)$. Hence, from Corollary 7.3, randomized Gap-ETH breaks. This concludes the proof of Theorem 8.1. □

In [23], an $\ell^{1-o(1)}$ factor inapproximability result for Maximum Colored Subgraph Isomorphism is proved, which is an improvement over the $\ell^{o(1)}$ factor hardness in Corollary 7.3. The authors of [23] then use this improved hardness together with our reduction in Lemma 8.2 to arrive at their $k^{1/4-o(1)}$ factor hardness for DSN. Since Lemma 8.2 is used even in [23] but does not appear in the published version of [23], we have kept its proof in our paper.

Proof of Lemma 8.2. The reduction is similar to that of Dodis and Khanna [24]. In particular, given $\Gamma = (G, H, V_1 \cup \cdots \cup V_\ell)$ where H is the complete graph, the DSN instance is generated as follows.

- The vertex set V' is $(V_G \times [2]) \cup \{s_1, \ldots, s_\ell\} \cup \{t_1, \ldots, t_\ell\}$ (i.e. two copies of V together with ℓ new vertices designated as sources and ℓ new vertices designated as sinks).
- There are three types of edges in E'. First, for every $i \in [\ell]$, there is an edge from s_i to each vertex in $V_i \times \{1\}$. Moreover, for every $i \in [\ell]$, there is an edge from each vertex in $V_i \times \{2\}$ to t_i. Finally, there is an edge from $(u, 1)$ to $(v, 2)$ and from $(v, 1)$ to $(u, 2)$

for every edge uv in the original graph G. In other words, $E' = \{(s_i, (v, 1)) \mid i \in [k], v \in V_i\} \cup \{((v, 2), t_i) \mid i \in [k], v \in V_i\} \cup \{((u, 1), (v, 2)), ((v, 1), (u, 2)) \mid uv \in E_G\}$.

- The edges of the first two types have weight $1/(2\ell)$, whereas the edges of the last type have weight zero.
- Finally, the demands are simply (s_i, t_j) for every $i, j \in [\ell]$ such that $i \neq j$.

Clearly, the number of demand pairs k is $\ell^2 - \ell$ as desired. We now move on to show the completeness and soundness properties of the reduction.

(Completeness) If $\mathrm{val}(\Gamma) = 1$, then there exists $(v_1, \ldots, v_\ell) \in V_1 \times \cdots V_\ell$ that induces a clique. Thus, we can pick edges in the set $\{(s_i, (v_i, 1)) \mid i \in [\ell]\} \cup \{((v_i, 2), t_i) \mid i \in [\ell]\} \cup \{((v_i, 1), (v_j, 2)) \mid i, j \in [\ell], i \neq j\}$. Clearly, the cost of this network is exactly one and it satisfies all the demand pairs.

(Soundness) We will prove this by contrapositive. Suppose that there exists a network $N \subseteq G'$ of cost $\rho \leq 1/\sqrt{4\gamma}$. For each $i \in [\ell]$, let $S_i \subseteq V_G$ denote the set of all vertices v such that at least one of $(s_i, (v, 1))$ or $((v, 2), t_i)$ is included in N. Observe that, from how our graph G' is constructed, for every $i \neq j \in [k]$, the (s_i, t_j) demand implies that there exist $u \in S_i$ and $v \in S_j$ such that $uv \in E_G$. Observe also that, since N has cost ρ, $|S| \leq 2\ell \cdot \rho \leq \ell/\sqrt{\gamma}$.

Let $\phi : V_H \to V_G$ be a random assignment where each $\phi(i)$ is chosen independently uniformly at random from S_i. For every $i \neq j \in [\ell]$, since there exist $u \in S_i$ and $v \in S_j$ such that $uv \in E_G$, the probability that the superedge $ij \in E_H$ is covered is at least the probability that $\phi(i) = u$ and $\phi(j) = v$, which is equal to $\frac{1}{|S_i||S_j|}$. We now want a lower bound on the expected number of superedges covered by ϕ. For this, we use the following inequality which follows from a special case of Hölder's inequality for 3 variables[12]

$$\begin{aligned}\Big(\sum_{1\leq i\neq j\leq \ell} \frac{1}{|S_i||S_j|}\Big) \cdot \Big(\sum_{1\leq i\neq j\leq \ell} |S_i|\Big) \cdot \Big(\sum_{1\leq i\neq j\leq \ell} |S_j|\Big) &\geq \Big(\sum_{1\leq i\neq j\leq \ell} \Big(\frac{1}{|S_i||S_j|}\Big)^{1/3} \cdot |S_i|^{1/3} \cdot |S_j|^{1/3}\Big)^3 \\ &= \Big(\sum_{1\leq i\neq j\leq \ell} 1^{1/3}\Big)^3 \\ &= (\ell(\ell-1))^3 \qquad (2)\end{aligned}$$

Hence, we have that the expected number of superedges covered by ϕ is at least

$$\begin{aligned}\sum_{ij\in E_H} \frac{1}{|S_i||S_j|} &= \frac{1}{2} \sum_{1\leq i\neq j\leq \ell} \frac{1}{|S_i||S_j|} \\ &\geq \frac{1}{2} \cdot \frac{(\ell(\ell-1))^3}{\left(\sum_{1\leq i\neq j\leq \ell} |S_i|\right)\left(\sum_{1\leq i\neq j\leq \ell} |S_j|\right)} \quad \text{(From Equation (2))} \\ &= \binom{\ell}{2} \cdot \frac{\ell^2}{|S|^2} \quad \Big(\text{Since } \sum_{1\leq i\neq j\leq \ell} |S_i| \leq (\ell-1) \cdot |S|\Big) \\ &\geq \binom{\ell}{2}\gamma,\end{aligned}$$

where the last inequality follows from $|S| \leq \ell/\sqrt{\gamma}$. Hence, there exists an assignment of Γ with value at least γ, which implies that $\mathrm{val}(\Gamma) \geq \gamma$. This concludes the proof of Lemma 8.2. □

[12] $(\sum_{r=1}^n a_r^3)(\sum_{r=1}^n b_r^3)(\sum_{r=1}^n c_r^3) \geq (\sum_{r=1}^n a_r b_r c_r)^3$

9 Open Questions

While our work has advanced our understanding of the computational complexity of SCSS and DSN, there are still several interesting open questions left. We list some of them below:

- Can we get better approximation algorithms for BI-DSN (without any restriction on the optimum) than simply getting twice the best ratio known for the undirected STEINER FOREST problem? This is an interesting question for both the parameterized and polynomial time setting.
- We showed that for BI-DSN there is both a parameterized 2-approximation algorithm and a polynomial-sized $(2+\varepsilon)$-approximate kernel for any $\varepsilon > 0$. However the latter is just a simple consequence of the PSAKS for BI-DSN$_{\text{PLANAR}}$. Is there a polynomial-sized c-approximate kernel with $c \leq 2$ for BI-DSN? Note that this relates to the previous question as well.
- We proved that the parameterized 2-approximation algorithm for SCSS is best possible, since no $(2-\varepsilon)$-approximation can be computed in $f(k) \cdot n^{O(1)}$ time for any function f, under Gap-ETH. This implies that there is a 2-approximate kernel (of large size), while no $(2-\varepsilon)$-approximate kernel exists under the same assumption. However, can we obtain a *polynomial-sized* 2-approximate kernel for SCSS? Or maybe just a polynomial-sized c-approximate kernel for some constant $c \geq 2$?
- Can we prove any runtime lower bound under some reasonable complexity assumption (e.g., ETH or Gap-ETH) to compute a 2-approximation for SCSS using the number of terminals as a parameter? In other words, could there be a significantly faster 2-approximation algorithm than the one given in [15]?
- We gave a $2^{k^2+O(k)} \cdot n^{O(1)}$ time FPT algorithm for BI-SCSS and a lower bound of $2^{o(k)} \cdot n^{O(1)}$. Can we obtain a single-exponential FPT algorithm for BI-SCSS?
- What is the status of BI-DSN on planar input graphs parameterized by k: FPT or W[1]-hard? Our hardness reduction in Theorem 1.3 produces graphs that are not planar even though their optima are.
- Can the parameterized approximation scheme for BI-DSN$_{\text{PLANAR}}$ be generalized to minor-closed classes of graphs? In particular, the KPR Theorem used to prove Theorem 4.1 is applicable to such classes. What prevents us to generalize here are the vertex degree transformation of Section 2.2, since applying these to some graph excluding a fixed minor can result in a graph containing this minor.
- Theorem 4.1 inherently introduces a double exponential term in $O(1/\varepsilon)$ to the kernel size for BI-DSN$_{\text{PLANAR}}$, and a triple exponential term to the runtime of the approximation scheme for BI-DSN$_{\text{PLANAR}}$. As argued in Section 4 it is known that the bound in Theorem 4.1 cannot be improved. Is there a different technique that yields a parameterized approximation scheme and/or a PSAKS for BI-DSN$_{\text{PLANAR}}$, which has better dependence on $1/\varepsilon$? Or alternatively, is there some reasonable complexity assumption that can exclude such an improvement?

References

[1] Ajit Agrawal, Philip Klein, and R Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem on networks. *SIAM Journal on Computing*, 24(3):440–456, 1995.

[2] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the ACM*, 42(4): 844–856, 1995.

[3] Benny Applebaum. Exponentially-Hard Gap-CSP and Local PRG via Local Hardcore Functions. In *FOCS 2017*, pages 836–847. doi: 10.1109/FOCS.2017.82.

[4] MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Dániel Marx. Approximation schemes for Steiner forest on planar graphs and graphs of bounded treewidth. *Journal of the ACM*, 58(5):21, 2011.

[5] 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.

[6] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets Möbius: fast subset convolution. In *STOC*, pages 67–74, 2007.

[7] Al Borchers and Ding-Zhu Du. The k-Steiner Ratio in Graphs. *SIAM Journal on Computing*, 26(3):857–869, 1997.

[8] Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. Steiner tree approximation via iterative randomized rounding. *Journal of the ACM*, 60(1):6, 2013.

[9] Parinya Chalermsook, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, and Luca Trevisan. From gap-exponential time hypothesis to fixed parameter tractable inapproximability: Clique, dominating set, and more. *SIAM J. Comput.*, 49(4):772–810, 2020.

[10] Moses Charikar, Chandra Chekuri, To-Yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. Approximation algorithms for directed Steiner problems. *J. Algorithms*, 33 (1):73–91, 1999.

[11] 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*, 7(2):18, 2011.

[12] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci.*, 72(8):1346–1367, 2006.

[13] W-T Chen and N-F Huang. The strongly connecting problem on multihop packet radio networks. *IEEE Transactions on Communications*, 37(3):293–295, 1989.

[14] Rajesh Chitnis and Andreas Emil Feldmann. FPT inapproximability of directed cut and connectivity problems. In *IPEC*, pages 8:1–8:20, 2019.

[15] Rajesh Chitnis, MohammadTaghi Hajiaghayi, and Guy Kortsarz. Fixed-parameter and approximation algorithms: A new look. In *IPEC*, pages 110–122, 2013.

[16] Rajesh Chitnis, Andreas Emil Feldmann, and Pasin Manurangsi. Parameterized Approximation Algorithms for Bidirected Steiner Network Problems. In *ESA*, pages 20:1–20:16, 2018.

[17] Rajesh Hemant Chitnis, Andreas Emil Feldmann, Mohammad Taghi Hajiaghayi, and Dániel Marx. Tight bounds for planar strongly connected Steiner subgraph with fixed number of terminals (and extensions). *SIAM J. Comput.*, 49(2):318–364, 2020.

[18] Janka Chlebiková and Miroslav Chlebík. The Steiner tree problem on graphs: Inapproximability results. *Theoretical Computer Science*, 406(3):207–214, 2008.

[19] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. ISBN 978-3-319-21275-3.

[20] Reinhard Diestel. *Graph Theory, 5th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2017. ISBN 978-3-662-53621-6.

[21] Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007.

[22] Irit Dinur. Mildly exponential reduction from gap 3SAT to polynomial-gap label-cover. *ECCC*, 23:128, 2016.

[23] Irit Dinur and Pasin Manurangsi. ETH-Hardness of Approximating 2-CSPs and Directed Steiner Network. In *ITCS*, pages 36:1–36:20, 2018. doi: 10.4230/LIPIcs.ITCS.2018.36.

[24] Yevgeniy Dodis and Sanjeev Khanna. Design networks with bounded pairwise distance. In *STOC 1999*, pages 750–759.

[25] Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Kernelization Lower Bounds Through Colors and IDs. *ACM Trans. Algorithms*, 11(2):13:1–13:20, 2014.

[26] S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1(3):195–207, 1971.

[27] Ding-Zhu Du, Yanjun Zhang, and Qing Feng. On better heuristic for Euclidean Steiner minimum trees. In *FOCS 1991*, pages 431–439.

[28] Jack Edmonds. Optimum Branchings. *Journal of Research of the National Bureau of Standards*, B71:233–240, 1967.

[29] Eduard Eiben, Dusan Knop, Fahad Panolan, and Ondrej Suchý. Complexity of the steiner network problem with respect to the number of terminals. In *STACS*, pages 25:1–25:17, 2019.

[30] David Eisenstat, Philip Klein, and Claire Mathieu. An efficient polynomial-time approximation scheme for Steiner forest in planar graphs. In *SODA 2012*, pages 626–638.

[31] Jittat Fakcharoenphol and Kunal Talwar. An improved decomposition theorem for graphs excluding a fixed minor. In *Approximation, Randomization, and Combinatorial Optimization.. Algorithms and Techniques*, pages 36–46. 2003.

[32] Jon Feldman and Matthias Ruhl. The directed Steiner network problem is tractable for a constant number of terminals. *SIAM J. Comput.*, 36(2):543–561, 2006.

[33] Moran Feldman, Guy Kortsarz, and Zeev Nutov. Improved approximation algorithms for directed steiner forest. *J. Comput. Syst. Sci.*, 78(1):279–292, 2012.

[34] Andreas Emil Feldmann and Dániel Marx. The complexity landscape of fixed-parameter directed Steiner network problems. In *ICALP*, pages 27:1–27:14, 2016.

[35] Andreas Emil Feldmann, Jochen Könemann, Neil Olver, and Laura Sanità. On the equivalence of the bidirected and hypergraphic relaxations for Steiner tree. *Mathematical programming*, 160(1-2):379–406, 2016.

[36] Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theor. Comput. Sci.*, 410(1):53–61, 2009.

[37] Fedor V. Fomin, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. Subexponential Parameterized Algorithms for Planar and Apex-Minor-Free Graphs via Low Treewidth Pattern Covering. In *FOCS*, pages 515–524, 2016.

[38] Greg N Frederickson and Joseph JaJa. Approximation algorithms for several graph augmentation problems. *SIAM Journal on Computing*, 10(2):270–283, 1981.

[39] Bernhard Fuchs, Walter Kern, D Molle, Stefan Richter, Peter Rossmanith, and Xinhui Wang. Dynamic programming for minimum steiner trees. *Theory of Computing Systems*, 41(3):493–500, 2007.

[40] E. N. Gilbert and H. O. Pollak. Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 16(1):1–29, 1968.

[41] Fabrizio Grandoni, Bundit Laekhanukit, and Shi Li. $O(\log^2 k \,/\, \log\log k)$-approximation algorithm for directed steiner tree: a tight quasi-polynomial-time algorithm. In *STOC*, pages 253–264, 2019.

[42] Jiong Guo, Rolf Niedermeier, and Ondrej Suchý. Parameterized complexity of arc-weighted directed Steiner problems. *SIAM J. Discrete Math.*, 25(2):583–599, 2011.

[43] Eran Halperin and Robert Krauthgamer. Polylogarithmic inapproximability. In *STOC*, pages 585–594, 2003.

[44] Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

[45] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

[46] Giuseppe F Italiano, Yahav Nussbaum, Piotr Sankowski, and Christian Wulff-Nilsen. Improved algorithms for min cut and max flow in undirected planar graphs. In *STOC 2011*, pages 313–322.

[47] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Plenum, 1972.

[48] Marek Karpinski and Alexander Zelikovsky. New approximation algorithms for the Steiner tree problem. *Journal of Combinatorial Optimization*, 1(1):47–65, 1997.

[49] Hervé Kerivin and A Ridha Mahjoub. Design of survivable networks: A survey. *Networks*, 46(1):1–21, 2005.

[50] Philip Klein, Serge A. Plotkin, and Satish Rao. Excluded Minors, Network Decomposition, and Multicommodity Flow. In *STOC 1993*, pages 682–690.

[51] Philip N. Klein and Dániel Marx. Solving Planar k -Terminal Cut in $O(n^{c\sqrt{k}})$ Time. In *ICALP*, pages 569–580, 2012.

[52] Philip N. Klein and Dániel Marx. A subexponential parameterized algorithm for Subset TSP on planar graphs. In *SODA*, pages 1812–1830, 2014.

[53] Guy Kortsarz and David Peleg. On choosing a dense subgraph (extended abstract). In *FOCS 1993*, pages 692–701.

[54] Nhat X Lam, Trac N Nguyen, Min Kyung An, and Dung T Huynh. Dual power assignment optimization and fault tolerance in WSNs. *Journal of Combinatorial Optimization*, 30(1): 120–138, 2015.

[55] James Lee. A simpler proof of the KPR theorem, 2012. URL https://tcsmath.wordpress.com/2012/01/11/a-simpler-proof-of-the-kpr-theorem/. accessed: 23.6.2020.

[56] Daniel Lokshtanov, Saket Saurabh, and Magnus Wahlström. Subexponential Parameterized Odd Cycle Transversal on Planar Graphs. In *FSTTCS*, pages 424–434, 2012.

[57] Daniel Lokshtanov, Fahad Panolan, MS Ramanujan, and Saket Saurabh. Lossy Kernelization. In *STOC*, pages 224–237, 2017.

[58] Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Parameterized complexity and approximability of directed odd cycle transversal. In *SODA*, pages 2181–2200, 2020.

[59] Pasin Manurangsi. Almost-polynomial ratio ETH-hardness of approximating densest k-subgraph. In *STOC*, pages 954–961, 2017.

[60] Pasin Manurangsi and Prasad Raghavendra. A Birthday Repetition Theorem and Complexity of Approximating Dense CSPs. In *ICALP*, pages 78:1–78:15, 2017.

[61] Dániel Marx. On the Optimality of Planar and Geometric Approximation Schemes. In *FOCS*, pages 338–348, 2007.

[62] Dániel Marx. Can You Beat Treewidth? *Theory of Computing*, 6(1):85–112, 2010.

[63] Dániel Marx. A Tight Lower Bound for Planar Multiway Cut with Fixed Number of Terminals. In *ICALP*, pages 677–688, 2012.

[64] Dániel Marx and Michal Pilipczuk. Optimal Parameterized Algorithms for Planar Facility Location Problems Using Voronoi Diagrams. In *ESA*, pages 865–877, 2015.

[65] Dániel Marx, Marcin Pilipczuk, and Michal Pilipczuk. On subexponential parameterized algorithms for Steiner tree and directed subset TSP on planar graphs. In *FOCS*, pages 474–484, 2018.

[66] Dana Moshkovitz. The Projection Games Conjecture and the NP-hardness of $\ln n$-approximating Set-Cover. *Theory Comput.*, 11:221–235, 2015.

[67] Moni Naor, Leonard J. Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. In *FOCS*, pages 182–191, 1995.

[68] Jesper Nederlof. Fast polynomial-space algorithms using inclusion-exclusion. *Algorithmica*, 65(4):868–884, 2013.

[69] Krzysztof Pietrzak. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *J. Comput. Syst. Sci.*, 67(4): 757–771, 2003.

[70] Marcin Pilipczuk, Michal Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. Subexponential-Time Parameterized Algorithm for Steiner Tree on Planar Graphs. In *STACS*, pages 353–364, 2013.

[71] Marcin Pilipczuk, Michał Pilipczuk, Piotr Sankowski, and Erik Jan Van Leeuwen. Network sparsification for Steiner problems on planar and bounded-genus graphs. *ACM Transactions on Algorithms (TALG)*, 14(4):1–73, 2018.

[72] Hans Jürgen Prömel and Angelika Steger. A new approximation algorithm for the Steiner tree problem with performance ratio 5/3. *Journal of Algorithms*, 36:89–101, 2000.

[73] Ram Ramanathan and Regina Rosales-Hain. Topology control of multihop wireless networks using transmit power adjustment. In *INFOCOM*, volume 2, pages 404–413, 2000.

[74] Gabriel Robins and Alexander Zelikovsky. Tighter bounds for graph Steiner tree approximation. *SIAM Journal on Discrete Mathematics*, 19(1):122–134, 2005.

[75] Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2001. ISBN 978-3-662-04565-7.

[76] Adrian Vetta. Approximating the minimum strongly connected subgraph via a matching lower bound. In *SODA*, volume 7, pages 417–426, 2001.

[77] Jens Vygen. Faster algorithm for optimum steiner trees. *Information Processing Letters*, 111(21-22):1075–1079, 2011.

[78] Chen Wang, Myung-Ah Park, James Willson, Yongxi Cheng, Andras Farago, and Weili Wu. On approximate optimal dual power assignment for biconnectivity and edge-biconnectivity. *Theoretical Computer Science*, 396(1-3):180–190, 2008.

[79] Richard T Wong. A dual ascent approach for Steiner tree problems on a directed graph. *Mathematical programming*, 28(3):271–287, 1984.

[80] Alexander Zelikovsky. An 11/6-approximation algorithm for the network Steiner problem. *Algorithmica*, 9:463–470, 1993.