An ETH-Tight Algorithm for Bidirected Steiner Connectivity

Daniel Lokshtanov[1], Pranabendu Misra[2], Fahad Panolan[3(✉)], Saket Saurabh[4,5], and Meirav Zehavi[6]

[1] University of California, Santa Barbara, USA
[2] Chennai Mathematical Institute, Chennai, India
[3] Indian Institute of Technology, Hyderbad, India
fahad@cse.iith.ac.in
[4] The Institute of Mathematical Sciences, HBNI, Chennai, India
[5] University of Bergen, Bergen, Norway
[6] Ben-Gurion University, Beersheba, Israel

Abstract. In the Strongly Connected Steiner Subgraph problem, we are given an n-vertex digraph D, a weight function $w\colon A(D) \mapsto \mathbb{R}^+$ on the arc set of D, and a set of k terminals $Q \subseteq V(D)$, and our objective is to find a strongly connected subgraph of D containing Q with minimum total weight. The problem is known to be W[1]-hard on general digraphs. However on bi-directed graphs (digraphs where, if uv is an arc then so is vu) with symmetric weight function $w\colon A(D) \mapsto \mathbb{R}^+$ (i.e., $w(uv) = w(vu)$ for any $uv \in A(D)$), Chitnis, Feldmann and Manurangsi [TALG 2021] showed that the problem is fixed parameter tractable (FPT) with running time $2^{\mathcal{O}(k^2)}n^{\mathcal{O}(1)}$, where n is the input length. They also show that, unless the Exponential Time Hypothesis (ETH) fails, there is no algorithm for the problem on bi-directed graphs with running time $2^{o(k)}n^{\mathcal{O}(1)}$. They left the existence of a single-exponential in k time algorithm as an open problem. We resolve this question, by designing an algorithm for the problem running in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ that is asymptotically tight under ETH, thereby closing the gap between the upper and lower-bounds for this problem.

Chitnis, Feldmann and Manurangsi [TALG 2021] showed that an optimum solution to this problem can always be described by a collection of trees, that are mapped to the input graph via homomorphisms, and glued together at the terminal vertices. This structural result allows us to design an algorithm via the combination of a Dreyfus-Wagner style dynamic programming algorithm and the notion of representative sets over linear matroids.

Keywords: FPT · Graph Connectivity · Representative Family · Matroids

Supported by NSF award CCF-2008838, Swarnajayanti Fellowship grant DST/SJF/MSA-01/2017-18, and ERC grants LOPRE (grant agreement No. 819416) and PARAPATH.

P. Morin and S. Suri (Eds.): WADS 2023, LNCS 14079, pp. 588–604, 2023.
https://doi.org/10.1007/978-3-031-38906-1_39

1 Introduction

A central family of problems in network design addresses the objective of connecting a given set of terminals in a given network in a particular way, as to ensure that the terminals can transmit information from one to the other in that particular way. For instance, such problems naturally arise when modelling radio or ad-hoc wireless networks. Here, a very general setting is when the input consists of an arc-weighted digraph D as well as a collection $\mathcal{Q}$ of pairs of *terminals* (designated vertices) in D, while the objective is to find a subgraph of D with minimum cost (in terms of arc weights) such that for each terminal pair $(s,t) \in \mathcal{Q}$, there is a directed path from s to t in the subgraph. Unfortunately, this problem, called Directed Steiner Network (DSN) (also known as Directed Steiner Forest, although the sought solution may not be a forest), is particularly hard. On the one hand, it is W[1]-hard with respect to $k = |\mathcal{Q}|$ [27], which means that it is unlikely to be *fixed-parameter tractable (FPT)*—that is, solvable in time $f(k) \cdot n^{\mathcal{O}(1)}$ for some computable function f of k. In fact, unless Exponential Time Hypothesis (ETH) fails, it is not even solvable in time $f(k) \cdot n^{o(k)}$ for any computable function f of k [10]. On the other hand, DSN cannot be approximated within factor $O(2^{\log^{1-\epsilon} n})$ in polynomial time unless NP $\subseteq$ DTIME$(2^{\text{polylog}(n)})$ [16]. Furthermore, even combining both FPT and approximation also does not seem to be of much help. Indeed, Dinur and Manurangsi [15] have shown that the approximation of DSN within factor $k^{\frac{1}{4}-o(1)}$ is already not FPT unless Gap-ETH fails. Recently, Manurangsi et al. [31] proved that there is no FPT approximation algorithm for DSN with a factor of $o(k^{\frac{1}{2}})$ assuming Strongish Planted Clique Hypothesis. On the positive side, DSN is solvable in time $n^{\mathcal{O}(k)}$ [19], and can be approximated within factors $\mathcal{O}(n^{\frac{2}{3}+\epsilon})$ and $\mathcal{O}(k^{\frac{1}{2}+\epsilon})$ in polynomial time [2,6,20]. When the arc costs are uniform, DSN can be approximated within a factor $\mathcal{O}(n^{\frac{3}{5}+\epsilon})$ [11].

When dealing with undirected graphs, the problem is *significantly* easier at both parameterized complexity and approximation fronts. Here, we are given an undirected graph G and a set of terminals Q, and the objective is to find a subtree of G with minimum number of edges containing Q. This problem is called Steiner Tree (ST), and a more general version called Steiner Forest is also well studied. Steiner Tree is one of the 21 NP-hard problems in the seminal paper of Karp [28]. Already in 1971, it was shown to be FPT with respect to $k = |Q|$—specifically, it was shown to be solvable in time $3^k \cdot n^{\mathcal{O}(1)}$ [18]. The running time was improved to $(2+\epsilon)^k \cdot n^{f(\frac{1}{\epsilon})}$ and later to $2^k \cdot n^{\mathcal{O}(1)}$ for some computable function f of $\frac{1}{\epsilon}$ and $\epsilon > 0$ [4,23]. The algorithm in [4] works for the weighted version of the problem with edge weights from $\{1, 2, \ldots, M\}$, and the running time will have an additional multiplicative factor of M. On the other hand, unless ETH fails, Steiner Tree cannot be solved in time $2^{o(n)}$ and hence neither in time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$. (In fact, unless the so called Set Cover Conjecture (SeCoCo) fails, it is not even solvable in time $(2-\epsilon)^k \cdot n^{\mathcal{O}(1)}$ [13].) From the approximation perspective, ST can be approximated within a constant factor in polynomial-time: for over several decades, there is an ongoing quest to find

the best possible approximation factor (see, e.g., [5,25,33,37]); to the best of our knowledge, the current best approximation factor is of $\ln(4)+\epsilon < 1.39$ [5,25]. On the negative side, unless P=NP, ST is inapproximable within factor $\frac{96}{95}$ [3,12]. It is known that unless PH collapses, ST does not admit a polynomial kernel [17].

Between these two extremes, lies the Strongly Connected Steiner Subgraph (SCSS) problem. Here, given an arc weighted digraph D and a collection of terminals Q, the objective is to find a strongly connected subgraph of D with minimum number of arcs containing Q. Notice that the SCSS problem is precisely the special case of DSN where $\mathcal{Q} = \{(u, v) : u, v \in Q\}$. Further, the flavor of SCSS is precisely that of ST in the sense that every terminal can transmit information to every other terminal in the sought subnetwork, which is a realistic demand in network design. Unlike DSN, SCSS is known to be approximable within factor 2 in $3^k \cdot n^{\mathcal{O}(1)}$ time [9]. Unfortunately, like DSN, SCSS is W[1]-hard, and in fact it cannot even be approximated within factor $(2 - \epsilon)$ in time $f(k) \cdot n^{\mathcal{O}(1)}$ unless Gap-ETH fails [8]. We remark that the special case of SCSS where $Q = V(D)$, known as Minimum Strongly Connected Spanning Subgraph, can be approximated in polynomial time within factor 2 [22] and when all the edge weights are equal to 1, it has a factor $\frac{3}{2}$ approximation in polynomial time [35]. Also, it can be solved in single-exponential time in n (specifically, $2^{4\omega n} n^{\mathcal{O}(1)}$ where ω is the matrix multiplication exponent) [21].

Knowing that SCSS is unlikely to be FPT, how can we make its flavor more similar to ST so that it will be tractable (FPT) yet will still deal naturally with digraphs? Recently, Chitnis, Feldmann and Manurangsi [8] initiated a study where they restricted DSN to *bi-directed graphs* called Bi-DSN. Formally, a digraph D is bi-directed if for any pair of vertices $\{u, v\}$ in D, either both arcs uv and vu belong to D, or none of them does. Moreover, if $uv, vu \in A(D)$, then their weights are also equal. As noted in [8], bi-directed graphs model some realistic settings [7,30,34,36]—for instance, when nodes have the same transmitter model (e.g., in some wireless networks), thus if a node u can transmit information to a node v, the node v can transmit information to the node u as well. Critically, unlike undirected graphs which also model this property, bi-directed graphs capture the property that if we want to transmit information in both directions, then we need to pay *twice*, once for each direction. Thus, similarly to general digraphs, orientation plays a role. In particular, it can completely change the set of optimal solutions; for example, see Fig. 1, taken from [8]. Chitnis et al. proved that Bi-DSN is W[1]-hard, and it admits a polynomial time 4-approximation algorithm and a 2-approximation algorithm running in time $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$. Formally, Bi-Strongly Connected Steiner Subgraph (Bi-SCSS) is defined as follows.

Bi-SCSS **Parameter:** $k = |Q|$
Input: A bi-directed graph D, a set of terminals $Q \subseteq V(D)$ and a symmetric weight function $w : A(D) \to \mathbb{R}^+$ such that $w(uv) = w(vu)$ for any $uv \in A(D)$.
Output: A strongly connected subgraph of D containing Q of minimum total weight.

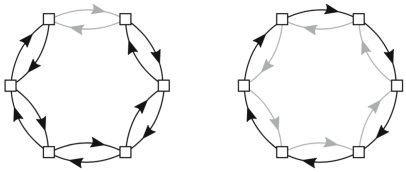


Fig. 1. An example in [8] of a Bi-SCSS instance where all vertices are terminals. Left: Black edges show a solution which takes an undirected optimum twice. Right: The actual optimum solution is shown in black.

In [8], it was proved that Bi-SCSS is solvable in time $2^{\mathcal{O}(k^2)} \cdot n^{\mathcal{O}(1)}$, as well as that it is NP-hard and that unless ETH fails, it cannot be solved in time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$. It was noted in [8] that bidirected inputs are the first example where SCSS remains NP-hard but turns out to be FPT parameterized by k. The results of [8] leave a gap between the known upper and conditional lower bounds and they posed the following open question.

> Can we obtain a single-exponential FPT algorithm for Bi-SCSS?

In this paper, we answer the above question in the affirmative.

Theorem 1. *The* Bi-SCSS *problem is solvable in time* $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$.

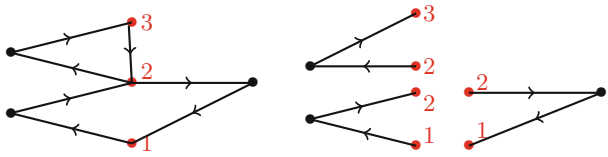
In particular, due to the aforementioned lower bound, *our algorithm is optimal under the ETH.* Our algorithm builds on a structural result which states that a minimal solution can always be described as a sum of trees mapped by homomorphisms into the graph and glued together at the terminal vertices. This result follows from [8]. The structural result allows us to find an optimal solution by making use of representative sets combined with a Dreyfus-Wagner style dynamic programming algorithm. We note the technique of representative sets has been used to improve the running times of FPT and exact algorithms of various problems and design efficient kernelization algorithms [21,26,29]. We describe our algorithms for the unweighted version of the problem, which easily extends to the symmetric weighted version via weighted representative sets. A short technical overview of our result is given below.

Our Methods. We start with the definition of homomorphism on digraphs. A *homomorphism* from a digraph H to a digraph D is a function ϕ from $V(H)$ to $V(D)$ such that for any $uv \in A(H)$, $\phi(u)\phi(v) \in A(D)$. Our first step is a structural analysis of solutions to an instance of Bi-SCSS. This result shows that optimal solutions can be decomposed into a sum of trees mapped by a homomorphism from a strongly connected digraph and glued together at the terminal vertices. Towards that we make the following observation. Any solution D' to an instance of Bi-SCSS (D, Q) can be viewed as the image of a homomorphism

from a strongly connected graph H that contains all of Q. Observe that, if there is a homomorphism ϕ from a strongly connected graph H to D then the image of ϕ forms a strongly connected subgraph D' of D with $|A(D')| \leq |A(H)|$. Let us note that H need not be a subgraph of D. Therefore, one can view a minimum solution to an instance (D, Q) of Bi-SCSS as a homomorphism ϕ from a strongly connected graph H with minimum number of edges such that $Q \subseteq \phi(V(H))$. Formally, consider an instance (D, Q) of Bi-SCSS, and let H be a strongly connected graph and ϕ be a homomorphism from H to D such that $Q \subseteq \phi(V(H))$. Then $\phi(H)$ is a solution to (D, Q) corresponding to the pair (H, ϕ).

We consider both D and H as k-boundaried graphs. In a *k-boundaried digraph* D, a certain subset of k vertices, denoted by $\beta(G)$, are tagged as *boundary vertices* and are labeled with $1, \ldots, k$. For an instance (D, Q) of Bi-SCSS, the boundary vertices in D are exactly the vertices in Q (i.e., $Q = \beta(D)$). Then we say that (H, ϕ), where H is a strongly connected k-boundaried digraph and ϕ is a homomorphism that preserves labels (i.e., the vertex with label i in H will be mapped to the vertex in D with label i), is a *solution* to the k-boundaried bi-directed graph D. The *elements* of a k-boundaried digraph H are defined as follows. Let C be a connected component in $\overline{H} - \beta(H)$, where $\overline{H}$ is the underlying undirected graph of H. Here, $\overline{H}$ is a simple graph that has an edge between u and v if and only if $uv \in A(H)$ or $vu \in A(H)$. Then the subgraph of H induced on the closed neighborhood of $V(C)$, excluding the arcs incident only on $\beta(H)$, i.e., $H[N[V(C)]] - A(\beta(H))$, is called an element of H. One can show that there is an optimal solution (H, ϕ) such that for every element J of H, $\overline{J}$ is a tree. This result follows directly from [8, Lemma 5.2]. In other words, we can view H as a collection of trees that are glued together at boundary vertices union the set of edges within the boundary (see Fig. 2 for an illustration).

Having established the above structural result, we proceed to an algorithm for finding an optimum solution to the given instance. We design a Dreyfus-Wagner style dynamic programming (DP) algorithm via *representative families* over graphic matroids. Our algorithm builds upon a well known characterization of strongly connected digraph: a digraph is strongly connected if and only if for any vertex u there is a spanning *in-tree* and a spanning *out-tree* rooted at u. This characterization can be captured using a collection of linear matroids [24]. Then, we present a dynamic programming algorithm, that gradually builds an optimum solution to the given instance. Here, we crucially use the fact that this solution can be decomposed into a collection of trees, and these trees can be constructed via a Dreyfus-Wagner style dynamic programming [18]. To ensure that the algorithm runs in $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ time, we need an additional tool. At each step of the dynamic programming algorithm, we must prune the collection of partial solutions using the notion of representative sets [21] over linear matroids. Otherwise, the set of partial solution cannot be bounded by a function of k alone.



(a) Graph H (b) The elements of H.

Fig. 2. A strongly connected graph H with boundary vertices colored red is drawn in the left side. Its elements are drawn in the right side. (Color figure online)

2 Preliminaries

We refer the reader to [14] for standard graph theoretic notions that are not explicitly stated here. For a positive integer $t \in \mathbb{N}$, $[t]$ denotes the set $\{1, \ldots, t\}$. For a function $f\colon A \to B$ and $X \subseteq A$, $f(X) = \{f(x) : x \in X\}$. We use the term graph to denote an undirected graph without self-loops, but with parallel edges. However, the digraphs considered in this paper are without self-loops and parallel arcs. For a graph G, we use $V(G)$ and $E(G)$ to denote its vertex set and edge set, respectively. Let G be a graph, $u \in V(G)$, and $U \subseteq V(G)$. We use $d_G(u)$ to denote the number of edges in G that are incident to u. We use $N[U]$ to denote the closed neighborhood of U and $G[U]$ to denote the subgraph of G induced on U. We use $E(U)$ to denote the set $\{uv \in E(G) : u, v \in U\}$.

For a digraph D, we use $V(D)$ and $A(D)$ to denote its vertex set and arc set, respectively. For a digraph D, we use $\overline{D}$ to denote the underlying undirected graph of D. Moreover, if there exist two vertices $u, v \in V(D)$ such that $uv, vu \in A(D)$, then there will be two edges between u and v in $\overline{D}$. For a (di)graph D and an arc/edge subset F of D, $D[F]$ denotes the graph induced on F, i.e., $D[F]$ has vertex set $V(D)$ and arc/edge set F. A connected component of a digraph D is a maximal subgraph C of D such that $\overline{C}$ is a connected graph.

A *k-boundaried (di)graph* is a (di)graph G with a set $W \subseteq V(G)$ of cardinality at most k such that each vertex $w \in W$ has a unique label $\ell_G(w) \in \{1, \ldots, k\}$. The vertex subset W is referred to as the *boundary* of G. For a k-boundaried (di)graph G, the function $\beta(G)$ returns the boundary of G. For two k-boundaried (di)graphs G_1 and G_2, $G_1 \oplus G_2$ denotes the k-boundaried graph obtained by "gluing together the same labelled boundary vertices". That is, the gluing operation takes the disjoint union of G_1 and G_2 and identifies the vertices of $\beta(G_1)$ and $\beta(G_2)$ with the same label. If there are vertices $u_1, v_1 \in \beta(G_1)$ and $u_2, v_2 \in \beta(G_2)$ such that $\ell_{G_1}(u_1) = \ell_{G_2}(u_2)$ and $\ell_{G_1}(v_1) = \ell_{G_2}(v_2)$, then G has vertices u and v formed by unifying u_1 and u_2, and v_1 and v_2, respectively. Moreover, $uv \in E(G_1 \oplus G_2)$ if and only if $u_1v_1 \in E(G_1)$ or $u_2v_2 \in E(G_2)$. Also, u and v are boundary vertices in $G_1 \oplus G_2$ such that $\ell_{G_1 \oplus G_2}(u) = \ell_{G_1}(u_1)$ and $\ell_{G_1 \oplus G_2}(v) = \ell_{G_1}(v_1)$.

A *homomorphism* from a digraph H to a digraph D is a function ϕ from $V(H)$ to $V(D)$ such that for any $uv \in A(H)$, $\phi(u)\phi(v) \in A(D)$. Also, if D and

H are boundaried graphs we say that ϕ is a *label preserving homomorphism* if ϕ also satisfies the following: For all $b \in \beta(H)$, $\phi(b) \in \beta(D)$ and $\ell_H(b) = \ell_D(\phi(b))$.

A digraph T is an *arborescence* or an *out-tree* if there is a vertex u in T, called the root, such that, for any vertex v in T, there is exactly one directed path in T from u to v. We also call T a u-arborescence and T is an out-tree rooted at u. A digraph T is an *in-tree* if the digraph obtained from T by reversing all its arcs is an out-tree. For a digraph D, we say that a subgraph T of D is an in-branching (or an out-branching) of D if T is an in-tree (or an out-tree) and $V(D) = V(T)$. The following observation is an alternate characterization of in-trees and out-trees.

Observation 1. *Let T be a digraph and $u \in V(T)$. The digraph T is an in-tree rooted at u if and only if $\overline{T}$ is a tree, the out-degree of u is 0, and for each $v \in V(T) \setminus \{u\}$, there is exactly one arc vw in T for some $w \in V(T)$. Similarly, T is an out-tree if and only if $\overline{T}$ is a tree, in-degree of u is 0, and for each $v \in V(T) \setminus \{u\}$, there is exactly one arc wv in T for some $w \in V(T)$.*

3 Algorithm

Recall that, the input to Bi-SCSS is a pair (D, Q) where D is bi-directed graph D, and $Q \subseteq V(D)$ is the set of terminal vertices. Also recall that we present an algorithm for the unweighted version of the problem. First we state that a certain class of solutions for this problem admit a decomposition into trees. For notational convenience, instead of using a pair (D, Q) to denote an instance of Bi-SCSS, we consider it as a k-boundaried bi-directed graph D with boundary $\beta(D)$, where the boundary vertices $\beta(D)$ play the role of terminals. Recall that we have a labeling $\ell_D : \beta(D) \to [k]$ where $k = |\beta(D)|$. Let us begin with the following lemma that allows us to view a solution to an instance of Bi-SCSS as a homomorphism between graphs. Similar methods, that is, viewing solutions as homomorphisms were also considered before, for example [32].

Lemma 1. *Let D be an instance of* Bi-SCSS*, and H be a strongly connected digraph with boundary $\beta(H)$ such that $|\beta(H)| = |\beta(D)|$. If there is a label preserving homomorphism ϕ from H to D, then there is a strongly connected subgraph D' of D containing $\beta(D)$ such that $|A(D')| \leq |A(H)|$.*

Proof. We define the graph D' as follows. The vertex set of D' is $V(D') = \phi(V(H)) \subseteq V(D)$. For any arc $uv \in A(H)$, $\phi(u)\phi(v)$ belongs to $A(D')$. This completes the construction of D'.

Since H is strongly connected, between any two vertices x and y in H, there is a path P_1 from x to y, and there is a path from y to x in H. Each of these paths corresponds to a walk in D'. This implies that there is a path from $\phi(x)$ to $\phi(y)$, and there is a path from $\phi(y)$ to $\phi(x)$ in D'. Thus, D' is a strongly connected graph. Moreover, $\beta(D) \subseteq V(D')$ and $|A(D')| \leq |A(H)|$. □

Observe that, for a solution D' to the instance D, there is a trivial label preserving homomorphism from $H = D'$ to D. Thus, because of Lemma 1,

we can view the solutions of a Bi-SCSS instance D as homomorphisms from strongly connected graphs H to D that induce a one-to-one map from $\beta(H)$ to $\beta(D)$ that also preserves the labels. That is, a solution to an instance G of Bi-SCSS is a pair (H, ϕ) where H is a strongly connected k-boundaried digraph with $|\beta(H)| = |\beta(D)|$, and ϕ is a label preserving homomorphism from H to D.

Next we define a decomposition of a solution into its *elements*, and then prove that the underlying undirected graphs of elements of an optimum solution, are trees (see Fig. 2).

Definition 1 (Elements of a solution). *Let H be a k-boundaried graph and $Q = \beta(H)$. Let $C_1, \ldots, C_r$ be the connected components of $\overline{H} - Q$. Then, the k-boundaried graphs $H[N[V(C_1)]] - E(Q), \ldots, H[N[V(C_r)]] - E(Q)$ are called the* elements *of H. Here, for each element J, we have $\beta(J) = Q \cap V(J)$ and $\ell_J(b) = \ell_H(b)$ for all $b \in \beta(J)$.*

The following lemma (Lemma 2) directly follows from [8, Lemma 5.2]. As [8, Lemma 5.2] holds in the case of symmetric weight functions, Lemma 2 can be extended to the case of symmetric weight functions as well.

Lemma 2. *Let D be an instance of* Bi-SCSS*. Then, there exist a strongly connected subgraph D' of D containing all the boundary vertices with the least number of edges and a solution (H, ϕ) to D such that*

- *$D' = (\phi(V(H)), \{\phi(u)\phi(v) : uv \in A(H)\})$, and*
- *for every element J of H, $\overline{J}$ is a tree.*

We say that (H, ϕ) is a *homomorphism minimal* solution to D, if for every element J of H, $\overline{J}$ is a tree. Lemma 2 implies that there is an optimum homomorphism minimal solution.

Our algorithm is a Dreyfus-Wagner [18] style dynamic programming algorithm combined with representative families [21] on disjoint union of graphic matroids to prune the sizes of DP table entries. Let D be an instance of Bi-SCSS, where D is a k-boundaried graph. Let (H, ϕ) be an optimal homomorphism minimal solution to the instance D. Lemma 2 implies that for every element J of H, $\overline{J}$ is a tree. The tree structure of the elements of H allows us to do a DP algorithm in a manner that is designed for Steiner Tree by Dreyfus and Wagner [18]. But, both the sizes of elements and the number of potential choices for elements that need to be considered as partial solutions are not bounded by a function of k. To overcome this hurdle, we use the notion of representative families on graphic matroid to prune the list of partial solutions in our dynamic programming table.

The following proposition will allow us to see solutions as a union of in-branchings and out-branchings. This was used in previous results (for example [10]). The underlying tree structures in in-branchings and out-branchings are useful for applying representative families on graphic matroids.

Proposition 1. ([1]). *Let H be a digraph and $r \in V(H)$. Then, H is strongly connected if and only if there is an in-branching rooted at r and an out-branching rooted at r in H.*

Observation 2 (⋆).[1] *Let D be an instance of* Bi-SCSS *and (H, ϕ) be a homomorphism minimal solution to D. Then, for any element J of H and $u \in V(J)$ with $d_{\overline{J}}(u) \leq 1$, $u \in \beta(H)$.*

From now on, throughout the section we fix a k-boundaried digraph D as the input of Bi-SCSS *and $n = |V(D)|$.* Since we have to explicitly keep track of parts of in-branching and out-branching in a "partial solution" while doing DP, we refine the notion of solution from a pair to a quadruple.

Definition 2. *A* homomorphism minimal solution *to D is a quadruple $(H, \phi, T_{in}, T_{out})$, where H is a k-boundaried digraph, $|\beta(H)| = |\beta(D)|$, and $A(T_{in}) \cup A(T_{out}) = A(H)$ such that the following holds. Let $y \in \beta(H)$ be the vertex in H such that $\ell_H(y) = 1$.*

(i) The function ϕ is a label preserving homomorphism from H to D.
(ii) T_{in} is an in-branching of H rooted at y and T_{out} is an out-branching of H rooted at y.
(iii) For any element J of H and leaf u in J, $\overline{J}$ is a tree and $u \in \beta(H)$.

Observation 3. *Let $(H, \phi, T_{in}, T_{out})$ be a homomorphism minimal solution to D and J be an element of H. For any $v \in V(J) \setminus \beta(J)$, all the arcs in $A(H)$ that are incident to v, are also present in J.*

Observation 3 follows from the definition of elements. By Observation 3, for a homomorphism minimal solution $(H, \phi, T_{in}, T_{out})$, each element is a subgraph J of H such that $\overline{J}$ is a tree and the arcs in $A(H)$ that are incident to a non-boundary vertex $v \in V(J) \setminus \beta(J)$ are also present in the graph J. Thus, to have complete information about the elements we define an element tuple as follows. For a homomorphism minimal solution $(H, \phi, T_{in}, T_{out})$ and an element J of H, the *element quadruple* associated with J is a quadruple $(J, \psi, A_{in}, A_{out})$, where $\psi = \phi|_{V(J)}$, $A_{in} = A(J) \cap A(T_{in})$ and $A_{out} = A(J) \cap A(T_{out})$. We remark that the underlying undirected graphs of $J[A_{in}]$ and $J[A_{out}]$, are connected. The proof of the following observation follows from Observation 3.

Observation 4. *Let $(H, \phi, T_{in}, T_{out})$ be a homomorphism minimal solution to D and J be an element of H. Let $(J, \psi, A_{in}, A_{out})$ be the element quadruple associated with J. Let $u \in V(J)$ and A_u be the set of arcs in J that are incident with u. Let $F \subseteq A_u$ and let J' be the connected component of $J - F$ containing u. Then, for any $w \in V(J') \setminus (\beta(J) \cup \{u\})$, all the arcs of A_{in} and A_{out} that are incident with w, are also present in J'.*

In the DP algorithm, the first step is to compute a "representative family" of element quadruples and in the second step we compose element quadruples and the edges between terminals in an iterative manner using representative families again, to form a solution.

[1] The proofs of the results marked with ⋆ are deferred to the full version of the paper.

3.1 Step 1: Representative Family of Element Quadruples

To compute a representative family of element quadruples using a DP algorithm, we first define *element partial solutions*.

Definition 3. *An elemental partial solution is a 6-tuple* $(R, \phi, A_{in}, A_{out}, h, z)$*, where* R *is a* k*-boundaried digraph,* $A_{in} \subseteq A(R)$*,* $A_{out} \subseteq A(R)$*,* $A(R) = A_{in} \cup A_{out}$*,* $h \in V(R)$*, and* $z \in V(D)$ *such that the following holds.*

(1) $\overline{R}$ *is a tree.*
(2) If there is a vertex $x \in V(R)$ *with label* $k+1$*, then* $x = h$*.*
(3) ϕ *is a label preserving homomorphism from* R *to* D *such that* $\phi(h) = z$[2]*.*
(4) The set of vertices $\{x \in V(R) : \ell_R(x) \in [k]\}$ *forms an independent set in* $\overline{R}$*.*
(5) For any $u \in V(R) \setminus \beta(R)$*, there is exactly one arc of the form* uw *in* A_{in} *and there is exactly one arc of the form* $w'u$ *in* A_{out}*.*

Here, notice that we have used one more label for R. This is similar to the idea of having one more terminal vertex in the Dreyfus-Wagner algorithm. That is, this extra boundary vertex is used to join two partial solutions. Recall that, in the Dreyfus-Wagner algorithm, we compute a minimum spanning tree using a dynamic programming algorithm. Here, we use a similar kind of dynamic programming algorithm, but instead of computing one solution, we compute a set of elemental partial solutions such that a solution to D can be composed of the elemental partial solutions from the set we computed and arcs between boundary vertices.

We define the *size* of an elemental partial solution $(R, \phi, A_{in}, A_{out}, h, z)$ to be $|A(R)|$. Suppose we have computed the set $\mathcal{S}$ of all possible elemental partial solutions of size at most $2n-2$, then a solution can be constructed by composing the element quadruples associated with it and the arcs between the terminal vertices. But the cardinality of $\mathcal{S}$ is not bounded by a function of k. Thus, instead of computing $\mathcal{S}$, we compute only a "representative of $\mathcal{S}$". In fact, in our DP algorithm we compute subsets of elemental partial solutions in the increasing order of their size that preserve some candidates which can be extended to an optimum solution. In short, we use ideas similar to the Dreyfus-Wagner algorithm to construct partial solutions of larger size from partial solutions of smaller size and we prune the partial solutions using representative sets on matroids to reduce its cardinality.

Definition 4. *Let* $Q = (R, \phi, A_{in}, A_{out}, h, z)$ *be a tuple satisfying the conditions in Definition 3. Let* $Z = (L, \psi, B_{in} \subseteq A(L), B_{out} \subseteq A(L), h^\star \in V(L), z)$ *be a tuple where* L *is a* $(k+1)$*-boundaried digraph,* $h^\star \in \beta(L)$*, such that*

(a) if there is a vertex x *such that* $\ell_L(x) = k+1$*, then* $x = h^\star$ *(i.e.,* $\ell_L^{-1}(k+1) \subseteq \{h^\star\}$*), and*
(b) ψ *is a label preserving homomorphism from* L *to* D *such that* $\psi(h^\star) = z$*.*

[2] Here, we slightly abuse the notation and consider z to be a labelled vertex in D with label $k+1$ if $\ell_R(h) = k+1$.

Let y be the vertex in $R \oplus L$ labelled with 1. We say that Q and Z ***can be merged****, if the following holds.*

(*i*) *$(V(R \oplus L), A_{in} \cup B_{in})$ and $(V(R \oplus L), A_{out} \cup B_{out})$ are an in-branching and an out-branching rooted at y of $R \oplus L$, respectively, and*
(*ii*) *there is a vertex with label j in $R \oplus L$ for all $j \in \{1, \ldots, k\}$.*

That is, $(R \oplus L, \psi^\star)$ forms a solution to D, where $\psi^\star(x) = \phi(x)$ for all $x \in V(R)$, and $\psi^\star(x) = \psi(x)$ for all $x \in V(L)$.

It is easy to prove that the function $\psi^\star$ in the above definition is well defined. Intuitively, in Definition 4, Q is a partial solution and it can be extended to a solution using the tuple Z which is yet to be computed by our DP algorithm. Now we define the notion of representative set of elemental partial solutions and later we will prove that a *small* set which represents any given set of elemental partial solutions can be computed *efficiently*.

Definition 5. *Let $\mathcal{F}$ be a set of elemental partial solutions. We say that a subset $\mathcal{F}' \subseteq \mathcal{F}$* represents *$\mathcal{F}$ if the following holds. Suppose there is a tuple $Z = (L, \psi, B_{in} \subseteq A(L), B_{out} \subseteq A(L), h^\star \in V(L), z \in V(D))$ and there is a tuple $Q \in \mathcal{F}$, such that Q and Z can be merged, then there is a tuple $Q' \in \mathcal{F}'$ such that Q' and Z can be merged and the size of Q' is at most the size of Q.*

Recall that $\mathcal{S}$ is the set of all possible elemental partial solutions. Next, we explain how to compute a subfamily $\mathcal{S}'$ that represents $\mathcal{S}$, of cardinality $2^{\mathcal{O}(k)}n$ using a Dreyfus-Wagner style dynamic programming algorithm along with representative families on the disjoint sum of two graphic matroids. In what follows we explain how to compute $\mathcal{S}'$ assuming Lemma 3 which can be proved using the notion of representative families on linear matroids and the proof is omitted here.

Lemma 3. *There is an algorithm that given an instance D (a k-boundaried graph) of* Bi-SCSS*, and a set of elemental partial solutions $\mathcal{F}$ of D, runs in time $\mathcal{O}(|\mathcal{F}| \cdot n)$ and outputs a subfamily $\mathcal{F}'$ of $\mathcal{F}$ of cardinality at most $20^{k+1}n$ that represents $\mathcal{F}$.*

Now, we are ready to explain a DP algorithm to compute a subfamily $\mathcal{S}'$ of $\mathcal{S}$, of cardinality $2^{\mathcal{O}(k)}n$, that represents $\mathcal{S}$. Let $\mathcal{S}_i$ be the set of all the elemental partial solutions of size at most i. At stage i, our algorithm will compute a subfamily $\mathcal{S}'_i$ of $\mathcal{S}_i$ that represents $\mathcal{S}_i$ in time $2^{\mathcal{O}(k)}n^3$ using already computed sets of partial solutions $\mathcal{S}'_1, \ldots, \mathcal{S}'_{i-1}$. Then, at the end it is straightforward to prove that the family $\widehat{\mathcal{S}} = \mathcal{S}'_1 \cup \ldots \cup \mathcal{S}'_{2n-2}$ is a representative for $\mathcal{S}$. Then, in the final step using similar pruning technique we compute an optimum solution again by a DP algorithm using $\widehat{\mathcal{S}}$.

Now, we get back to the computation of $\mathcal{S}'_i$ that represents $\mathcal{S}_i$, for all $i \in \{1, \ldots, 2n-2\}$. At the first step, we enumerate $\mathcal{S}_1$ (its cardinality will be bounded by n^2) and use Lemma 3 to compute a subfamily $\mathcal{S}'_1$ of size at most $20^{k+1}n$.

Computation of $\mathcal{S}'_i$ for $i \in \{2, \ldots, 2n-2\}$: Inductively, assume that we have computed $\mathcal{S}'_j$ of cardinality at most $20^{k+1}n$, for all $j \in \{1, \ldots, i-1\}$. First we set $\mathcal{S}^\star_i = \emptyset$. Then, we add the following tuples to $\mathcal{S}^\star_i$.

(i) For all $j, r \in [i-1]$ such that $i = j + r$, and for every pair of tuples $(R_1, \phi_1, A'_{in}, A'_{out}, h_1, z) \in \mathcal{S}'_j$ and $(R_2, \phi_2, A''_{in}, A''_{out}, h_2, z) \in \mathcal{S}'_r$ such that $\ell_{R_1}(h_1) = \ell_{R_2}(h_2)$, we add a tuple $Q = (R = R_1 \oplus R_2, \phi, A'_{in} \cup A''_{in}, A'_{out} \cup A''_{out}, h, z)$ to $\mathcal{S}^\star_i$, if Q is an elemental partial solution, where h is the vertex in R such that $\ell_R(h) = \ell_{R_1}(h_1) = \ell_{R_2}(h_2)$, and ϕ are defined as follows. For any $x \in V(R_q)$, $\phi(x) = \phi_q(x)$ for all $q \in \{1, 2\}$. Since $\phi_1|_{\beta(R_1)\cap\beta(R_2)}$ is same as $\phi_2|_{\beta(R_1)\cap\beta(R_2)}$, the function ϕ is well defined. Here we compute elemental partial solutions from already computed partial solutions by gluing at the vertices labelled $k+1$.

(ii) For each $(R', \phi', A'_{in}, A'_{out}, h', z') \in \mathcal{S}'_{i-1}$ and *arc* $z'z \in A(D)$ let $Q_1 = (R, \phi, A_{in} \cup \{h'h\}, A_{out}, h, z)$, $Q_2 = (R, \phi, A_{in}, A_{out} \cup \{h'h\}, h, z)$, and $Q_3 = (R, \phi, A_{in} \cup \{h'h\}, A_{out} \cup \{h'h\}, h, z)$, where R is the graph $(V(R') \cup \{h\}, A(R') \cup \{h'h\})$, $\beta(R) = (\beta(R') \cup \{h\}) \setminus \{h'\}$, and ϕ is defined as follows: for any $y \in V(R')$, $\phi(y) = \phi'(y)$, and $\phi(h) = z$. Here, $\ell_R(h) = k+1$ and $\ell_R(x) = \ell_{R'}(x)$ for all $x \in \beta(R) \setminus \{h\}$. Then, for any $j \in \{1, 2, 3\}$, we add Q_j to $\mathcal{S}^\star_i$ if Q_j is an elemental partial solution. Here we compute partial solutions from already computed partial solutions by gluing an arc where the head of the arc is a "new vertex".

(iii) We have three more cases which are omitted here, as they are identical to Case (ii). In those cases, we compute partial solutions from already computed partial solutions by gluing an arc where, (a) the tail of the arc is a "new vertex", (b) head of the arc is a "new vertex" and both the endpoints of the arc are boundary vertices, (c) tail of the arc is a "new vertex" and both the endpoints of the arc are boundary vertices, respectively.

Clearly, the cardinality of $\mathcal{S}^\star_i$ is bounded by $2^{\mathcal{O}(k)}n^2$, because $|\mathcal{S}'_j| \leq 20^{k+1}n$, for all $j \in \{1, \ldots, i-1\}$. Now, we use Lemma 3 and compute a representative $\mathcal{S}'_i$ of $\mathcal{S}^\star_i$. The construction of $\mathcal{S}'_i$ takes time $2^{\mathcal{O}(k)}n^3$ and the cardinality of $\mathcal{S}'_i$ is at most $20^{k+1}n$. Thus, the total running time to compute $\mathcal{S}'_1, \ldots, \mathcal{S}'_{2n-2}$ is $2^{\mathcal{O}(k)}n^4$.

Now, let $\widehat{\mathcal{S}} = \mathcal{S}'_1 \cup \ldots \cup \mathcal{S}'_{2n-2}$. Next, we prove that $\mathcal{S}'_i$ is a representative of $\mathcal{S}_i$, and $\widehat{\mathcal{S}}$ represents $\mathcal{S}$.

Lemma 4 ($\star$). *For all $i \in \{1, \ldots, 2n-2\}$, $\mathcal{S}'_i$ represents $\mathcal{S}_i$.*

Lemma 5. *$\widehat{\mathcal{S}}$ represents $\mathcal{S}$ and $|\widehat{\mathcal{S}}|$ is upper bounded by $\mathcal{O}(20^k n^2)$.*

Proof. Let $Q \in \mathcal{S}$ and Z be a tuple such that Q and Z can be merged. Let the size of Q be i. Then, we have that there exists $Q' \in \mathcal{S}'_i \subseteq \widehat{\mathcal{S}}$ such that Q' and Z can be merged. Because of Lemma 3, we have that $|\widehat{\mathcal{S}}| \leq \mathcal{O}(20^k n^2)$. □

3.2 Step 2: Composition of Element Quadruples

The next step is to compose the element quadruples to form a solution. In this step as well we use the notion of representative families and do a dynamic programming.

Definition 6. *A* partial solution *is a quadruple $(Y, \phi, A_{in}, A_{out})$, where Y is a k-boundaried digraph, $A_{in} \subseteq A(Y)$, $A_{out} \subseteq A(Y)$, and $A(Y) = A_{in} \cup A_{out}$ such that the following holds.*

(1) ϕ is a label preserving homomorphism from Y to D.
(2) For any $u \in V(Y) \setminus \beta(Y)$, there is exactly one arc of the form uw in A_{in} and there is exactly one arc of the form $w'u$ in A_{out}.

The *size* of a partial solution $(Y, \phi, A_{in}, A_{out})$ is $|A(Y)|$. For any two partial solutions $Q_1 = (Y_1, \phi_1, F_1, F_1')$ and $Q_2 = (Y_2, \phi_2, F_2, F_2')$, we define $Q_1 \circ Q_2$ to be the quadruple $(Y = Y_1 \oplus Y_2, \phi, F_1 \cup F_2, F_1' \cup F_2')$, where $\phi(x) = \phi_1(x)$ for all $x \in V(Y_1)$ and $\phi(x) = \phi_2(x)$ otherwise. Similar to the case of elemental partial solution, we define the notion of a representative for partial solutions and we prove that small representative families can be computed efficiently.

Definition 7. *Let $\mathcal{F}$ be a set of partial solutions. We say that a subset $\mathcal{F}' \subseteq \mathcal{F}$* represents *$\mathcal{F}$ if the following holds. Suppose there is a partial solution $Q \in \mathcal{F}$ and there is another partial solution Z (not necessarily in $\mathcal{F}$), such that $Q \circ Z$ is a solution to G, then there exists $Q' \in \mathcal{F}'$ such that $Q' \circ Z$ is a solution to D and the size of Q' is at most the size of Q.*

Lemma 6 ($\star$). *There is an algorithm that given an instance D of* Bi-SCSS*, where D is a k-boundaried bi-directed graph and a set of partial solutions $\mathcal{F}$ of D, runs in time $2^{\mathcal{O}(k)}|\mathcal{F}| \cdot n$ and outputs a subfamily $\mathcal{F}'$ of $\mathcal{F}$ of cardinality at most 20^k that represents $\mathcal{F}$.*

For any solution $(H, \phi, T_{in}, T_{out})$ of D, we know that the elements of H do not contain arcs between the boundary vertices. Let $\mathcal{B}$ be the set of all partial solutions $(Y, \phi, A_{in}, A_{out})$ of size 1 with $\beta(Y) = V(Y)$ and $|V(Y)| = 2$. Let $\widehat{\mathcal{R}}$ be the set of partial solutions extracted from $\widehat{\mathcal{S}}$ which is constructed in the previous subsection. That is,

$$\widehat{\mathcal{R}} = \{Q = (Y, \phi, A_{in}, A_{out}) : Q \text{ is a partial solution and } \exists h, z \text{ s.t. } (Y, \phi, A_{in}, A_{out}, h, z) \in \widehat{\mathcal{S}}\}.$$

Now compute sets $\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_{2n-2}$ in this order as follows. We set $\mathcal{P}_0 = \{(\emptyset, \emptyset, \emptyset, \emptyset)\}$. $\mathcal{P}_1 = \widehat{\mathcal{R}}$. For any $i \in [2n-2]$, first we set $\mathcal{Q}_i = \{Q \circ R : Q \in \mathcal{P}_{i-1} \text{ and } R \in \widehat{\mathcal{R}}\}$. Then, let $\mathcal{P}_i$ be the representative of $\mathcal{Q}_i$, computed using Lemma 6. That is, $\mathcal{P}_i$ will be a representative subfamily of partial solutions which are composed of i partial solutions from $\widehat{\mathcal{R}}$ (that corresponds to elemental partial solutions). Now, for each $i \in \{0, 1, 2n-1\}$ and $j \in \{0, \ldots, k^2\}$, we construct a set $\mathcal{P}_{i,j}$ as follows in the increasing order of j. We set $\mathcal{P}_{i,0} = \mathcal{P}_i$ for all $i \in \{0, \ldots, 2n-2\}$. Now, for any $j \in [k^2]$, let $\mathcal{Q}_{i,j} = \{P \circ B : P \in \mathcal{P}_{i,j-1} \text{ and } B \in \mathcal{B}\}$. Then, by using Lemma 6, we compute $\mathcal{P}_{i,j}$ which is a representative of $\mathcal{Q}_{i,j}$. Now, among all the tuples in $\bigcup_{i,j} \mathcal{Q}_{i,j}$ that are solutions of D, we output the solution with minimum size. This completes the algorithm.

Now, we prove the correctness of our algorithm. For each optimum solution $(H, \phi, T_{in}, T_{out})$, H is a union of elements and arcs between boundary vertices

of H. That is, $(H, \phi, T_{in}, T_{out})$ is composed of tuples from $\mathcal{S}$ and $\mathcal{B}$. That is, if H is composed of i elements and j arcs between boundary vertices, then there exist $P_1, \ldots, P_i \in \mathcal{R}$ (where $\mathcal{R}$ is defined below) and $B_1, \ldots B_j \in \mathcal{B}$ such that

$$(H, \phi, T_{in}, T_{out}) = ((((((P_1 \circ P_2) \circ \ldots) \quad \circ P_i) \circ B_1) \circ B_2) \circ \ldots) \circ B_j, \text{ where}$$
$$\mathcal{R} = \{Q = (Y, \phi, A_{in}, A_{out}) : Q \text{ is a partial solution and}$$
$$\exists h, z \text{ s.t. } (Y, \phi, A_{in}, A_{out}, h, z) \in \mathcal{S}\}$$

For each $i \in \{0, \ldots, 2n-2\}$ and $j \in \{0, \ldots, k^2\}$, we define $\mathcal{Z}_{i,j}$ as follows. We define $\mathcal{Z}_{0,0} = \{(\emptyset, \emptyset, \emptyset, \emptyset)\}$. For any $i \in [2n-2]$, $\mathcal{Z}_{i,0} = \{Z \circ P : Z \in \mathcal{Z}_{i-1,0} \text{ and } P \in \mathcal{R}\}$. Now, for any $j \in [k^2]$, we define $\mathcal{Z}_{i,j} = \{Z \circ P : Z \in \mathcal{Z}_{i,j-1} \text{ and } P \in \mathcal{B}\}$. That is, for an optimum solution $Z = (H, \phi, T_{in}, T_{out})$, if H is composed of i elements and j arcs between boundary vertices, then $Z \in \mathcal{Z}_{i,j}$. The correctness of our algorithm follows from the lemma below.

Lemma 7. *For any $i \in \{0, \ldots, 2n-2\}$ and $j \in \{0, \ldots, k^2\}$, $\mathcal{P}_{i,j}$ is a representative for $\mathcal{Z}_{i,j}$.*

Proof (Proof sketch). First using induction on i, we prove that $\mathcal{P}_{i,0}$ is a representative for $\mathcal{Z}_{i,0}$. The base case is when $i = 0$ and it is true because $\mathcal{P}_{0,0} = \mathcal{Z}_{0,0}$. Now, consider the induction step when $i > 0$. Suppose there is a partial solution $Z \in \mathcal{Z}_{i,0}$ and another partial solution F such that $Z \circ F$ is a solution to D. Then, there exists $Z_1 \in \mathcal{Z}_{i-1,0}$ and $Z_2 \in \mathcal{R}$ such that $Z = Z_1 \circ Z_2$. Let $F_1 = Z_2 \circ F$. Notice that $Z \circ F = (Z_1 \circ Z_2) \circ F = Z_1 \circ (Z_2 \circ F) = Z_1 \circ F_1$. Then, by induction hypothesis, there exists $\widehat{Z}_1 \in \mathcal{P}_{i-1,0}$ such that $\widehat{Z}_1 \circ F_1$ is a solution to D. Notice that $Z_2 \circ (\widehat{Z}_1 \circ F) = \widehat{Z}_1 \circ F_1$. Since $\widehat{\mathcal{S}}$ is a representative of $\mathcal{S}$ and from the construction of $\widehat{\mathcal{R}}$, there exists $\widehat{Z}_2 \in \widehat{\mathcal{R}}$ such that $\widehat{Z}_2 \circ (\widehat{Z}_1 \circ F)$ is a solution D. Notice that $\widehat{Z}_2 \circ (\widehat{Z}_1 \circ F)$ is equal to $(\widehat{Z}_1 \circ \widehat{Z}_2) \circ F$, where $\widehat{Z}_1 \in \mathcal{P}_{i-1,0}$ and $\widehat{Z}_2 \in \widehat{\mathcal{R}}$. By the definition of $\mathcal{Q}_{i,0}$, we get that $\widehat{Z}_1 \circ \widehat{Z}_2 \in \mathcal{Q}_{i,0}$. Then, since $\mathcal{P}_{i,0}$ is a representative of $\mathcal{Q}_{i,0}$, there is a partial solution $\widehat{Z} \in \mathcal{P}_{i,0}$ such that $\widehat{Z} \circ F$ is a solution to D and size of $\widehat{Z}$ is at most the size of $\widehat{Z}_1 \circ \widehat{Z}_2$ which is at most the size of Z. Thus, we have proved that for any $i \in \{0, \ldots, 2n-2\}$, $\mathcal{P}_{i,0}$ is a representative for $\mathcal{Z}_{i,0}$.

Using similar arguments, one can prove by induction on j that for any $j \in \{0, \ldots, k^2\}$, $\mathcal{P}_{i,j}$ is a representative for $\mathcal{Q}_{i,j}$. □

Running Time Analysis. We have already mentioned that the computation of $\mathcal{S}'_1, \ldots, \mathcal{S}'_{2n-2}$ takes time $2^{\mathcal{O}(k)}n^4$ and by Lemma 5 the cardinality of $\widehat{\mathcal{S}}$ is at most $\mathcal{O}(20^k n^2)$. Hence the cardinality of $\widehat{\mathcal{R}}$ is upper bounded by $\mathcal{O}(20^k n^2)$. By Lemma 6, the computation of $\mathcal{P}_0, \ldots, \mathcal{P}_{2n-2}$, takes time $2^{\mathcal{O}(k)}n^2$, and $|\mathcal{P}_i| \leq 20^k$, for all $i \in \{0, \ldots, 2n-2\}$. Because of Lemma 6 the computation of $\mathcal{P}_{i,j}$ for all $i \in \{0, \ldots, 2n-2\}$ and $j \in \{0, \ldots, k^2\}$ together takes time $2^{\mathcal{O}(k)}n^2$ and the cardinality of each $\mathcal{Q}_{i,j}$ is upper bounded by 20^k. Thus, the total running time of the algorithm is $2^{\mathcal{O}(k)}n^4$.

References

1. Bang-Jensen, J., Gutin, G.Z.: Digraphs: Theory, Algorithms and Applications, 2nd edn. Springer, Berlin (2008). Incorporated
2. Berman, P., Bhattacharyya, A., Makarychev, K., Raskhodnikova, S., Yaroslavtsev, G.: Approximation algorithms for spanner problems and directed Steiner forest. Inf. Comput. **222**, 93–107 (2013)
3. Bern, M.W., Plassmann, P.E.: The Steiner problem with edge lengths 1 and 2. Inf. Process. Lett. **32**(4), 171–176 (1989)
4. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Fourier meets möbius: fast subset convolution. In: Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, 11–13 June 2007, pp. 67–74 (2007)
5. Byrka, J., Grandoni, F., Rothvoß, T., Sanità, L.: Steiner tree approximation via iterative randomized rounding. J. ACM **60**(1), 6:1–6:33 (2013)
6. Chekuri, C., Even, G., Gupta, A., Segev, D.: Set connectivity problems in undirected graphs and the directed Steiner network problem. ACM Trans. Algorithms **7**(2), 18:1–18:17 (2011)
7. Chen, W., Huang, N.: The strongly connecting problem on multihop packet radio networks. IEEE Trans. Commun. **37**(3), 293–295 (1989)
8. Chitnis, R., Feldmann, A.E., Manurangsi, P.: Parameterized approximation algorithms for bidirected steiner network problems. ACM Trans. Algorithms **17**(2), 12:1–12:68 (2021). https://doi.org/10.1145/3447584
9. Chitnis, R.H., Hajiaghayi, M., Kortsarz, G.: Fixed-parameter and approximation algorithms: a new look. In: Parameterized and Exact Computation - 8th International Symposium, IPEC 2013, Sophia Antipolis, France, 4–6 September 2013, Revised Selected Papers, pp. 110–122 (2013)
10. Chitnis, R.H., Hajiaghayi, M., Marx, D.: Tight bounds for planar strongly connected Steiner subgraph with fixed number of terminals (and extensions). In: Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, 5–7 January 2014, pp. 1782–1801 (2014)
11. Chlamtáč, E., Dinitz, M., Kortsarz, G., Laekhanukit, B.: Approximating spanners and directed Steiner forest: upper and lower bounds. ACM Trans. Algorithms (TALG) **16**(3), 1–31(2020). https://doi.org/10.1145/3381451
12. Chlebík, M., Chlebíková, J.: The Steiner tree problem on graphs: inapproximability results. Theor. Comput. Sci. **406**(3), 207–214 (2008)
13. Cygan, M., et al.: On problems as hard as CNF-SAT. ACM Trans. Algorithms **12**(3), 41:1–41:24 (2016)
14. Diestel, R.: Graph Theory, 4th Ed. Graduate texts in mathematics, vol. 173. Springer, Cham (2012)
15. Dinur, I., Manurangsi, P.: ETH-hardness of approximating 2-CSPs and directed Steiner network. In: 9th Innovations in Theoretical Computer Science Conference, ITCS 2018, 11–14 January 2018, Cambridge, MA, USA, pp. 36:1–36:20 (2018)
16. Dodis, Y., Khanna, S.: Design networks with bounded pairwise distance. In: Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, 1–4 May 1999, Atlanta, Georgia, USA, pp. 750–759 (1999)
17. Dom, M., Lokshtanov, D., Saurabh, S.: Kernelization lower bounds through colors and IDs. ACM Trans. Algorithms **11**(2), 1–20 (2014). https://doi.org/10.1145/2650261

18. Dreyfus, S.E., Wagner, R.A.: The Steiner problem in graphs. Networks **1**(3), 195–207 (1971)
19. Feldman, J., Ruhl, M.: The directed Steiner network problem is tractable for a constant number of terminals. SIAM J. Comput. **36**(2), 543–561 (2006)
20. Feldman, M., Kortsarz, G., Nutov, Z.: Improved approximation algorithms for directed Steiner forest. J. Comput. Syst. Sci. **78**(1), 279–292 (2012)
21. Fomin, F.V., Lokshtanov, D., Panolan, F., Saurabh, S.: Efficient computation of representative families with applications in parameterized and exact algorithms. J. ACM **63**(4), 29:1–29:60 (2016)
22. Frederickson, G.N., JáJá, J.: Approximation algorithms for several graph augmentation problems. SIAM J. Comput. **10**(2), 270–283 (1981)
23. Fuchs, B., Kern, W., Mölle, D., Richter, S., Rossmanith, P., Wang, X.: Dynamic programming for minimum Steiner trees. Theory Comput. Syst. **41**(3), 493–500 (2007)
24. Gabow, H.N.: A matroid approach to finding edge connectivity and packing arborescences. J. Comput. Syst. Sci. **50**(2), 259–273 (1995). https://doi.org/10.1006/jcss.1995.1022
25. Goemans, M.X., Olver, N., Rothvoß, T., Zenklusen, R.: Matroids and integrality gaps for hypergraphic Steiner tree relaxations. In: Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, 19–22 May 2012, pp. 1161–1176 (2012)
26. Goyal, P., Misra, P., Panolan, F., Philip, G., Saurabh, S.: Finding even subgraphs even faster. J. Comput. Syst. Sci. **97**, 1–13 (2018). https://doi.org/10.1016/j.jcss.2018.03.001
27. Guo, J., Niedermeier, R., Suchý, O.: Parameterized complexity of arc-weighted directed Steiner problems. SIAM J. Discrete Math. **25**(2), 583–599 (2011)
28. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (eds.) Complexity of Computer Computations. The IBM Research Symposia Series, pp. 85–103. Springer, Boston (1972). https://doi.org/10.1007/978-1-4684-2001-2_9
29. Kratsch, S., Wahlström, M.: Representative sets and irrelevant vertices: new tools for kernelization. J. ACM **67**(3), 16:1–16:50 (2020). https://doi.org/10.1145/3390887
30. Lam, N.X., Nguyen, T.N., An, M.K., Huynh, D.T.: Dual power assignment optimization and fault tolerance in WSNS. J. Comb. Optim. **30**(1), 120–138 (2015)
31. Manurangsi, P., Rubinstein, A., Schramm, T.: The strongish planted clique hypothesis and its consequences. In: Lee, J.R. (ed.) 12th Innovations in Theoretical Computer Science Conference (ITCS 2021). Leibniz International Proceedings in Informatics (LIPIcs), vol. 185, pp. 10:1–10:21. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2021). https://doi.org/10.4230/LIPIcs.ITCS.2021.10
32. Nederlof, J.: Fast polynomial-space algorithms using inclusion-exclusion. Algorithmica **65**(4), 868–884 (2013). https://doi.org/10.1007/s00453-012-9630-x
33. Prömel, H.J., Steger, A.: A new approximation algorithm for the Steiner tree problem with performance ratio 5/3. J. Algorithms **36**(1), 89–101 (2000)
34. Ramanathan, R., Hain, R.: Topology control of multihop wireless networks using transmit power adjustment. In: Proceedings IEEE INFOCOM 2000, The Conference on Computer Communications, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Reaching the Promised Land of Communications, Tel Aviv, Israel, 26–30 March 2000, pp. 404–413 (2000)

35. Vetta, A.: Approximating the minimum strongly connected subgraph via a matching lower bound. In: Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, 7–9 January 2001, Washington, DC, USA, pp. 417–426 (2001)
36. Wang, C., Park, M.A., Willson, J., Cheng, Y., Farago, A., Wu, W.: On approximate optimal dual power assignment for biconnectivity and edge-biconnectivity. Theor. Comput. Sci. **396**(1–3), 180–190 (2008)
37. Zelikovsky, A.: An 11/6-approximation algorithm for the network Steiner problem. Algorithmica **9**(5), 463–470 (1993)