# A Tight Algorithm for Strongly Connected Steiner Subgraph On Two Terminals With Demands

Rajesh Hemant Chitnis[1⋆], Hossein Esfandiari[1⋆], MohammadTaghi Hajiaghayi[1⋆], Rohit Khandekar[2⋆⋆], Guy Kortsarz[3⋆⋆⋆], and Saeed Seddighin[1⋆]

[1] Department of Computer Science, University of Maryland at College Park, USA,
[2] KCG Holdings Inc., USA.
[3] Department of Computer Science, Rutgers University-Camden, USA.

**Abstract.** Given an edge-weighted directed graph $G = (V, E)$ on $n$ vertices and a set $T = \{t_1, t_2, \ldots t_p\}$ of $p$ terminals, the objective of the Strongly Connected Steiner Subgraph (SCSS) problem is to find an edge set $H \subseteq E$ of minimum weight such that $G[H]$ contains a $t_i \to t_j$ path for each $1 \leq i \neq j \leq p$. The problem is NP-hard, but Feldman and Ruhl [FOCS '99; SICOMP '06] gave a novel $n^{O(p)}$ algorithm for the $p$-SCSS problem.

In this paper, we investigate the computational complexity of a variant of 2-SCSS where we have demands for the number of paths between each terminal pair. Formally, the 2-SCSS-$(k_1, k_2)$ problem is defined as follows: given an edge-weighted directed graph $G = (V, E)$ with weight function $\omega : E \to \mathbb{R}_{\geq 0}$, two terminal vertices $s, t$, and integers $k_1, k_2$ ; the objective is to find a set of $k_1$ paths $F_1, F_2, \ldots, F_{k_1}$ from $s \rightsquigarrow t$ and $k_2$ paths $B_1, B_2, \ldots, B_{k_2}$ from $t \rightsquigarrow s$ such that $\sum_{e \in E} \omega(e) \cdot \phi(e)$ is minimized where $\phi(e) = \max\left\{|\{i : 1 \leq i \leq k_1, e \in F_i\}| \ ; \ |\{j : 1 \leq j \leq k_2, e \in B_j\}|\right\}$. For each $k \geq 1$, we show the following:

- The 2-SCSS-$(k, 1)$ problem can be solved in $n^{O(k)}$ time.
- A matching lower bound for our algorithm: the 2-SCSS-$(k, 1)$ problem does not have an $f(k) \cdot n^{o(k)}$ algorithm for any computable function $f$, unless the Exponential Time Hypothesis (ETH) fails.

Our algorithm for 2-SCSS-$(k, 1)$ relies on a structural result regarding the optimal solution followed by using the idea of a "token game" similar to that of Feldman and Ruhl [FOCS '99; SICOMP '06]. We show with an example that the structural result does not hold for the 2-SCSS-$(k_1, k_2)$ problem if $\min\{k_1, k_2\} \geq 2$. Therefore 2-SCSS-$(k, 1)$ is the most general problem one can attempt to solve with our technique. To obtain the lower bound matching the algorithm, we reduce from a special variant of the Grid Tiling problem introduced by Marx [FOCS '07; ICALP '12].

⋆ Supported in part by NSF CAREER award 1053605, NSF grant CCF-1161626, ONR YIP award N000141110662, DARPA/AFOSR grant FA9550-12-1-0423. Email: {rchitnis, hossein, hajiagha, }@cs.umd.edu

⋆⋆ Email: rkhandekar@gmail.com

⋆⋆⋆ Supported by NSF grant 1218620. Email: guyk@camden.rutgers.edu

The 2-SCSS-$(k_1, k_2)$ problem is as a special case of the Directed Capacitated Survivable Network Design (Dir-Cap-SNDP) problem introduced by Goemans et al. [9] in which we are given an directed multigraph with costs and capacities on the edges, and the question is to find a minimum cost subset of edges that satisfies all pairwise minimum-cut requirements.

## 1 Introduction

The Steiner Tree (ST) problem is one of the earliest and most fundamental problems in combinatorial optimization: given an undirected edge-weighted graph $G = (V, E)$ with edge weights $c : E \to \mathbb{R}^+$ and a set $T \subseteq V$ of terminals, the objective is to find a tree $S$ of minimum cost $c(S) := \sum_{e \in S} c(e)$ which spans all the terminals. The Steiner Tree problem is believed to have been first formally defined by Gauss in a letter in 1836. The first combinatorial formulation of the ST problem is attributed to Hakimi [12] in 1971. In the directed version of the ST problem, called Directed Steiner Tree (DST), we are also given a root vertex $r$ and the objective is to find a minimum size arborescence in the directed graph which connects the root $r$ to each terminal from $T$. An easy reduction from Set Cover shows that the DST problem is also NP-complete. The best known approximation ratio for DST in polynomial time is $|T|^\epsilon$ for any $\epsilon > 0$ due to Zelikovsky [23]. Halperin and Krauthgamer [13] showed an $\Omega(\log^{2-\epsilon} n)$ inapproximability for any $\epsilon > 0$ for the DST problem.

Steiner-type of problems arise in the design of networks. Since many networks are symmetric, the directed versions of Steiner type of problems were mostly of theoretical interest. However in recent years, it has been observed [20] that the connection cost in various networks such as satellite or radio networks are not symmetric. Therefore, directed graphs form the most suitable model for such networks. In addition, Ramanathan [20] also used the DST problem to find low-cost multicast trees, which have applications in point-to-multipoint communication in high bandwidth networks. A generalization of the DST problem is the Strongly Connected Steiner Subgraph (SCSS) problem. In the $p$-SCSS problem, given a directed graph $G = (V, E)$ and a set $T = \{t_1, t_2, \ldots, t_p\}$ of $p$ terminals the objective is to find a set $S \subseteq V$ such that $G[S]$ contains a $t_i \to t_j$ path for each $1 \leq i \neq j \leq p$. The best known approximation ratio in polynomial time for SCSS is $p^\epsilon$ for any $\epsilon > 0$ [3]. A result of Halperin and Krauthgamer [13] implies SCSS has no $\Omega(\log^{2-\epsilon} n)$-approximation for any $\epsilon > 0$, unless NP has quasi-polynomial Las Vegas algorithms.

The Directed Steiner Forest and Directed Steiner Network problems are two other standard generalizations of the DST problem. We mention more about their history in Appendix A.

**Token Game of Feldman and Ruhl:** The paper of Feldman and Ruhl [8] gives novel $n^{O(p)}$ algorithms for both the $p$-SCSS problem and the $p$-DSF problem[4]. They first consider the special case of 2-SCSS to illustrate their approach.

[4] Refer to Appendix B for a brief overview of previous work on exact algorithms for directed Steiner problems

We briefly mention their approach here: In the 2-SCSS problem, the input is a directed graph and two terminal vertices $s, t$ and we want to find a minimum subgraph containing $s \rightsquigarrow t$ and $t \rightsquigarrow s$ paths. They consider a "token game" as follows: there is one forward token $f$ and one backward token $b$. The forward token moves along the direction of the edges and the backward token moves in the reverse direction as that of the edges. Both tokens are initially located at $s$. Feldman and Ruhl define a set of "valid moves" for each of the tokens and an associated cost for each move. The main technical part of their paper is to show that for every move sequence (which takes both tokens from $s$ to $t$) of total cost $C$ there is a corresponding solution of 2-SCSS of size $C + 1$, and if the optimum solution for the 2-SCSS instance has cost $C + 1$ then there is a move sequence (which takes both tokens from $s$ to $t$) of total cost $C$.

**The 2-SCSS-$(k_1, k_2)$ Problem:** We define the following generalization of the 2-SCSS problem:

**2-SCSS-$(k_1, k_2)$**
*Input* : An edge-weighted digraph $G = (V, E)$ with weight function $\omega : E \to \mathbb{R}_{\geq 0}$, two terminal vertices $s, t$, and integers $k_1, k_2$
*Question*: Find a set of $k_1$ paths $F_1, F_2, \ldots, F_{k_1}$ from $s \rightsquigarrow t$ and $k_2$ paths $B_1, B_2, \ldots, B_{k_2}$ from $t \rightsquigarrow s$ such that $\sum_{e \in E} \omega(e) \cdot \phi(e)$ is minimized where $\phi(e) = \max\Big\{|\{i : 1 \leq i \leq k_1, e \in F_i\}| \ ; \ |\{j : 1 \leq j \leq k_2, e \in B_j\}|\Big\}$.

Observe that 2-SCSS-$(1, 1)$ is the same as the 2-SCSS problem. The definition of the 2-SCSS-$(k_1, k_2)$ problem allows us to potentially choose the same edge multiple times, but we have to pay for each time we use it in a path between a given terminal pair. This can be thought of as **"buying disjointness"** by adding parallel edges. In large real-world networks, it might be more feasible to modify the network by adding some parallel edges to create disjoint paths than finding disjoint paths in the existing network. Teixira et al. [21, 22] model path diversity in Interner Service Provider (ISP) networks and the Sprint network by disjoint paths between two hosts. There have been several patents [10, 19] attempting to design multiple paths between the components of Google Data Centers.

The 2-SCSS-$(k_1, k_2)$ problem is as a special case of the Directed Survivable Network Design (Dir-Cap-SNDP) problem [9] in which we are given an directed multigraph with costs and capacities on the edges, and the question is to find a minimum cost subset of edges that satisfies all pairwise minimum-cut requirements. In the 2-SCSS-$(k_1, k_2)$ problem, we do not require disjoint paths. As observed in Chakrabarty et al. [2] and Goemans et al. [9], the Dir-Cap-SNDP problem becomes much easier to approximate if we allow taking multiple copies of each edge.

### 1.1 Our Results and Techniques:

In this paper, we consider the 2-SCSS-$(k, 1)$ problem parameterized by $k$, which is the sum of all the demands. To the best of our knowledge, this is the first study

of parameterized complexity of any directed connectivity problem with respect to this parameter. In fact, we are unaware of any non-trivial exact algorithms for a version of the SCSS problem with demands between the terminal pairs. Our main algorithmic result is the following:

**Theorem 1.** *The 2-SCSS-$(k, 1)$ problem can be solved in $n^{O(k)}$ time.*

Our algorithm proceeds as follows: In Section 2.1 we first show that there is an optimal solution for the 2-SCSS-$(k, 1)$ problem which satisfies a structural property which we call as *reverse-compatibility*. Then in Section 2.2 we introduce a "Token Game" (similar to that of Feldman and Ruhl [FOCS '99; SICOMP '06]) and show that it can be solved in $n^{O(k)}$. Finally in Section 2.3, using the existence of an optimal solution satisfying reverse-compatibility, we give a reduction from the 2-SCSS-$(k, 1)$ problem to the Token Game which gives an $n^{O(k)}$ algorithm for the 2-SCSS-$(k, 1)$ problem. This algorithm also generalizes the result of Feldman and Ruhl [8] for 2-SCSS, since 2-SCSS is equivalent to 2-SCSS-$(1, 1)$. In Appendix F, we show with an example (see Figure 4) that the structural result does not hold for the 2-SCSS-$(k_1, k_2)$ problem if $\min\{k_1, k_2\} \geq 2$. Therefore 2-SCSS-$(k, 1)$ is the most general problem one can attempt to solve with our technique.

Theorem 1 does not rule out the possibility that the 2-SCSS-$(k, 1)$ problem is not solvable in polynomial time. Our main hardness result rules out this possibility by showing that our algorithm is *tight* and that the exponent of $O(k)$ is best possible.

**Theorem 2.** *The 2-SCSS-$(k, 1)$ problem is W[1]-hard parameterized by $k$. Moroever, under the ETH the 2-SCSS-$(k, 1)$ problem cannot be solved in $f(k) \cdot n^{o(k)}$ time for any function $f$, where $n$ is the number of vertices in the graph.*

We reduce from the Grid Tiling problem formulated by the pioneering work on Marx in FOCS '07 (see [15]):

$k \times k$ **Grid Tiling**
*Input* : Integers $k, n$, and $k^2$ non-empty sets $S_{i,j} \subseteq [n] \times [n]$ where $i, j \in [k]$
*Question*: For each $1 \leq i, j \leq k$ does there exist a value $s_{i,j} \in S_{i,j}$ such that

- If $s_{i,j} = (x, y)$ and $s_{i,j+1} = (x', y')$ then $x = x'$.
- If $s_{i,j} = (x, y)$ and $s_{i+1,j} = (x', y')$ then $y = y'$.

The Grid Tiling problem has turned to be a convenient starting point for parameterized reductions for planar problems, and has been used recently in various W[1]-hardness proofs on planar graphs [16, 5, 17]. Under the ETH, Chen et al. [4] showed that $k$-Clique[5] does not admit an algorithm running in time $f(k) \cdot n^{o(k)}$ for any function $f$. Marx [15] gave a reduction from $k$-Clique to $k \times k$ Grid Tiling. In Section 3, we give a reduction from $k \times k$ Grid Tiling to 2-SCSS-$(k, 1)$. Since the parameter blowup is linear, the $f(k) \cdot n^{o(k)}$ lower bound for Grid Tiling from [15] transfers to 2-SCSS-$(k, 1)$.

[5] The $k$-Clique problem asks whether there is a clique of size $\geq k$?

We would like to point out two *distinctive features* of our reduction. Firstly, the graph that we construct in the reduction is not planar, and to the best of our knowledge this is the first use of the Grid Tiling problem to show W[1]-hardness for a problem on non-planar graphs. Secondly, the reduction in [15] from $k$-Clique to $k \times k$ Grid Tiling actually shows the hardness of a special case of the Grid Tiling problem where the sets are constructed as follows: given a graph $G = (V, E)$ for the $k$-Clique problem with $V = \{v_1, v_2, \ldots, v_n\}$ we set $S_{i,i} = \{(j, j) : 1 \leq j \leq [n]\}$ for each $i \in [k]$ and $S_{i,f} = \{(j, \ell) : 1 \leq j \neq \ell \leq n, (v_j, v_\ell) \in E\}$ for each $1 \leq i \neq f \leq k$. We call this as the Grid Tiling* problem and actually give a reduction from this problem to 2-SCSS-$(k, 1)$. To the best of our knowledge, this is the first use of the special structure of Grid Tiling* in a W[1]-hardness proof.

**Note:** In Appendix C, we show that the edge-weighted and the vertex-weighted variants of 2-SCSS-$(k_1, k_2)$ are computationally equivalent. Therefore, henceforth we consider only the edge-weighted version of 2-SCSS-$(k_1, k_2)$.

## 2 An $n^{O(k)}$ algorithm for 2-SCSS-$(k, 1)$

In this section we describe an algorithm for the 2-SCSS-$(k, 1)$ problem running in $n^{O(k)}$ time where $n$ is the number of vertices in the graph. First in Section 2.1 we present a structural property called as *reverse compatibility* for one optimal solution of this problem. Next we define a Token Game in Section 2.2 and provide an $n^{O(k)}$ algorithm to solve the game. Finally, in Subsection 2.3 we present an algorithm that finds the optimum solution of 2-SCSS-$(k, 1)$ in time $n^{O(k)}$ via a reduction to the Token Game problem.

### 2.1 The Structural Lemma

For simplicity, we replace each edge of the input graph with $k$ copies of that edge which allows us to force the forwarding paths to be edge-disjoint. Since forwarding paths can use different copies of the same edge, this change in the problem can be done without loss of generality.

**Definition 1. (path-reverse-compatible)** *Let $F$ be a $s \rightsquigarrow t$ path and $B$ be a $t \rightsquigarrow s$ path. Let $\{B'_1, B'_2, \ldots, B'_d\}$ be the set of maximal sub-paths that $F$ and $B$ share. Suppose that for all $j \in [d]$, $B'_j$ is the $j$-th sub-path that we see while traversing $F$. We say the pair $(F, B)$ is path-reverse-compatible if for all $j \in [d]$, $B'_j$ is the $(k - j + 1)$-th sub-path that we see while traversing $B$, i.e., $B'_j$ is the $j$-th sub-path that we see while traversing $B$ backward.*

**Definition 2. (reverse-compatible)** *Let $\boldsymbol{F} = \{F_1, F_2, \ldots, F_d\}$ be a set of $s \rightsquigarrow t$ paths and $b$ be a $t \rightsquigarrow s$ path. We say $(\boldsymbol{F}, B)$ is reverse-compatible, if for all $1 \leq i \leq k$ the pair $(F_i, B)$ is path-reverse-compatible.*

The next lemma shows that there exists an optimum solution for 2-SCSS-$(k, 1)$ which is reverse-compatible. The proof of this lemma is tricky but short.

**Lemma 1.** [⋆][6] **(structural lemma)** *There exists an optimum solution for* 2-*SCSS-*$(k, 1)$ *which is reverse-compatible.*

### 2.2 The Token Game

In the token game, we are given a graph $G$, a set of tokens $\mathcal{T}$, vertices $s$ and $t$, a set of moves $\mathcal{M}$, and a cost function $\hat{C} : \mathcal{M} \to \mathbb{R}$. Each move $m \in \mathcal{M}$ consists of a set of triples $(t_i, u_i, v_i)$ where $t_i \in \mathcal{T}$ is a token, and $u_i$ and $v_i$ are vertices of the graph. In order to apply a move $m = \{(t_1, u_1, v_1), (t_2, u_2, v_2), \ldots, (t_d, u_d, v_d)\}$ to a state of the game, each token $t_i$ should be on vertex $u_i$ for all $1 \leq i \leq d$ and after applying this move, for every triple $(t_i, u_i, v_i) \in m$ token $t_i$ will be transported to the vertex $v_i$. For each $m \in \mathcal{M}$, $\hat{C}(m)$ specifies the cost of applying $m$ to the game. Initially, all of the tokens are placed on vertex $s$. In each step, we apply a move $m \in \mathcal{M}$ to the game with cost $\hat{C}(m)$ and the goal is to transport all of the tokens to the vertex $t$ with minimum cost.

In the following, we present an algorithm to solve an instance $\langle G, s, t, \mathcal{T}, \mathcal{M}, \hat{C}\rangle$ of the Token game in time $\mathcal{O}(n^{|\mathcal{T}|}|\mathcal{M}| \log(n^{|\mathcal{T}|}))$ where $n$ is the number of the vertices of $G$.

**Lemma 2.** [⋆] **(algorithm for Token Game)** *There exists an algorithm which solves the Token game in time* $\mathcal{O}(n^{|\mathcal{T}|}|\mathcal{M}| \log(n^{|\mathcal{T}|}))$.

### 2.3 Reduction to the Token Game

Here, we provide a reduction from the 2-SCSS-$(k, 1)$ problem to the Token game. As a consequence, we show that one can use the presented algorithm in Subsection 2.2 to solve 2-SCSS-$(k, 1)$ in time $\mathcal{O}(n^{\mathcal{O}(k)})$.

Let $I = \langle G, s, t\rangle$ be an instance of the 2-SCSS-$(k, 1)$. We reduce $I$ to an instance $Cor(I) = \langle G', s', t', \mathcal{T}, \mathcal{M}, \hat{C}\rangle$ of the Token Game problem where $G = G'$, $s = s'$, $t = t'$ and $\mathcal{T}$ is a set of $k+1$ tokens $\{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k, \mathcal{B}\}$. Furthermore, $\mathcal{M}$ and $\hat{C}$ are constructed in the following way:

- For every edge $(u, v) \in E(G)$, we add $d$ moves $\{(\mathcal{F}_i, u, v)\}$ to $\mathcal{M}$ for all $1 \leq i \leq d$. Cost of each move is equal to the length of its corresponding edge in $G$.
- For every edge $(u, v) \in E(G)$ with weight $w$, we add a move $\{(\mathcal{B}, v, u)\}$ to $\mathcal{M}$ with cost $w$.
- For every pair of vertices $u$ and $v$ in $G$, we add $k$ moves $\{(\mathcal{F}_i, u, v), (\mathcal{B}, v, u)\}$ to $\mathcal{M}$ for all $1 \leq i \leq k$. Cost of each move is equal to the distance of vertex $v$ from vertex $u$ in $G$.

Next, we show that $opt(I) = opt(Cor(I))$ where $opt(I)$ and $opt(Cor(I))$ stand for the optimum solutions of $I$ and $Cor(I)$, respectively. Therefore we provide two lemmas to show that $opt(I) \geq opt(Cor(I))$ and $opt(I) \leq opt(Cor(I))$.

[6] The proofs of the results labeled with ⋆ have been deferred to the appendix.

**Lemma 3.** [⋆] *For a given instance $I$ of the 2-SCSS-$(k,1)$ we have $opt(I) \geq opt(Cor(I))$.*

**Lemma 4.** [⋆] *For a given instance $I$ of the 2-SCSS-$(k,1)$ we have $opt(I) \leq opt(Cor(I))$ .*

**Theorem 3.** [⋆] *There exists an algorithm that solves the 2-SCSS-$(k,1)$ in time $\mathcal{O}(n^{\mathcal{O}(k)})$.*

## 3 $f(k) \cdot n^{o(k)}$ Hardness for 2-SCSS-$(k,1)$

In this section we prove Theorem 2. We reduce from the Grid Tiling problem (see Section 1.1 for definition). Chen et al. [4] showed that for any function $f$ an $f(k) \cdot n^{o(k)}$ algorithm for Clique implies ETH fails. Marx [15] gave the following reduction which transforms the problem of finding a $k$-Clique into an instance of $k \times k$ Grid Tiling as follows: For a graph $G = (V, E)$ with $V = \{v_1, v_2, \ldots, v_n\}$ we build an instance $I_G$ of Grid Tiling

- For each $1 \leq i \leq k$, we have $(j, \ell) \in S_{i,i}$ if and only if $j = \ell$.
- For any $1 \leq i \neq j \leq k$, we have $(\ell, r) \in S_{i,j}$ if and only if $\{v_\ell, v_r\} \in E$.

It is easy to show that $G$ has a clique of size $k$ if and only if the instance $I_G$ of Grid Tiling has a solution. Therefore, assuming ETH, the following special case of $k \times k$ Grid Tiling also cannot be solved in time $f(k) \cdot n^{o(k)}$ for any computable function $f$.

$k \times k$ **Grid Tiling***
*Input* : Integers $k, n$, and $k^2$ non-empty sets $S_{i,j} \subseteq [n] \times [n]$ where $1 \leq i, j \leq k$ such that for each $1 \leq i \leq k$, we have $(j, \ell) \in S_{i,i}$ if and only if $j = \ell$
*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 $x = x'$.
- If $\gamma_{i,j} = (x, y)$ and $\gamma_{i+1,j} = (x', y')$ then $y = y'$.

Consider an instance of Grid Tiling*. We now build an instance of edge-weighted 2-SCSS-$(2k-1, 1)$ as shown in Figure 1. We consider $4k$ special vertices: $(a_i, b_i, c_i, d_i)$ for each $i \in [k]$. We introduce $k^2$ red gadgets where each gadget is an $n \times n$ grid. Let weight of each black edge be 4.

**Definition 3.** *For each $1 \leq i \leq k$, an $a_i \rightsquigarrow b_i$* canonical *path is a path from $a_i$ to $b_i$ which starts with a blue edge coming out of $a_i$, then follows a horizontal path of black edges and finally ends with a blue edge going into $b_i$. Similarly an $c_j \rightsquigarrow d_j$* canonical *path is a path from $c_j$ to $d_j$ which starts with a blue edge coming out of $c_j$, then follows a vertically downward path of black edges and finally ends with a blue edge going into $d_j$.*

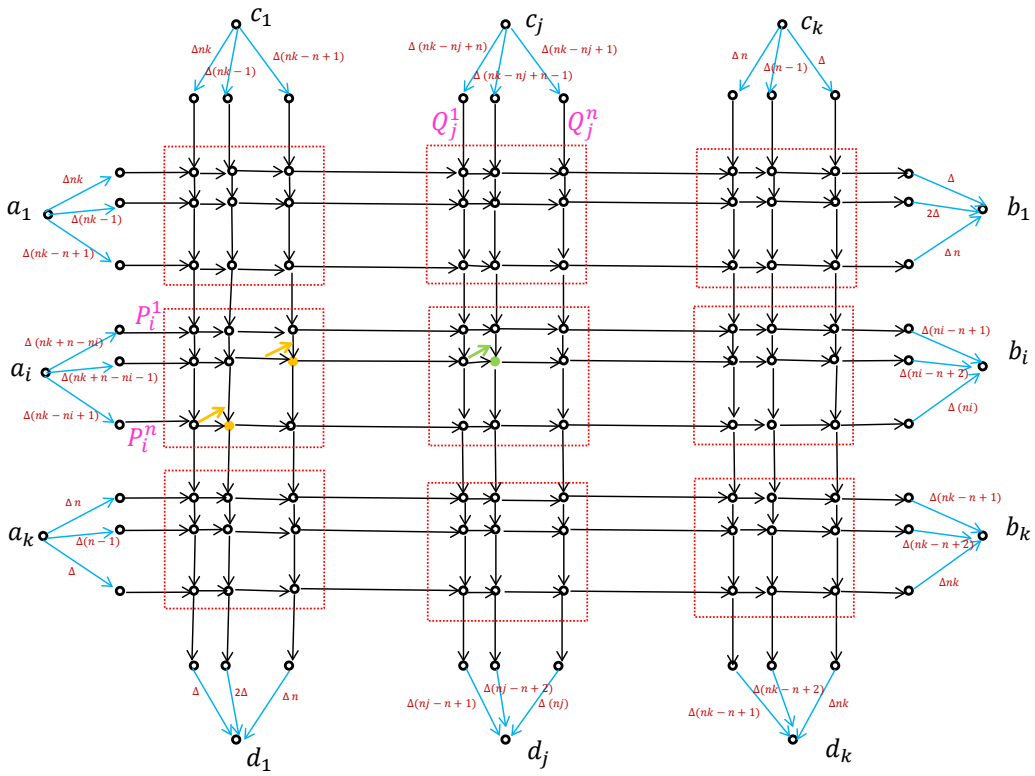


**Fig. 1.** The instance of 2-SCSS-$(k, 1)$ created from an instance of Grid Tiling*.

For each $1 \leq i \leq k$, there are $n$ edge-disjoint $a_i \rightsquigarrow b_i$ canonical paths: let us call them $P_i^1, P_i^2, \ldots, P_i^n$ as viewed from top to bottom. They are named using magenta color in Figure 1. Similarly we call the canonical paths from $c_j$ to $d_j$ as $Q_j^1, Q_j^2, \ldots, Q_j^n$ when viewed from left to right. For each $i \in [k]$ and $\ell \in [n]$ we assign a weight of $\Delta(nk - ni + n + 1 - \ell), \Delta(ni - n + \ell)$ to the first, last edges of $P_i^\ell$ (which are colored blue) respectively. Similarly for each $j \in [k]$ and $\ell \in [n]$ we assign a weight of $\Delta(nk - nj + n + 1 - \ell), \Delta(nj - n + \ell)$ to the first, last edges of $Q_j^\ell$ (which are colored blue) respectively. Thus the total weight of first and last blue edges on any canonical path is exactly $\Delta(nk+1)$. The idea is to choose $\Delta$ large enough such that in any optimum solution the paths between the terminals will be exactly the canonical paths. We will see that $\Delta = 7n^6$ will suffice for our reduction. Any canonical path uses two blue edges (which sum up to $\Delta(nk+1)$), $(k+1)$ black edges not inside the gadgets and $(n-1)$ black edges inside each gadget. Since the number of gadgets each canonical path visits is $k$ and the weight of each black edge is 4, we have the total weight of any canonical path is $\alpha = \Delta(nk+1) + 4(k+1) + 4k(n-1)$.

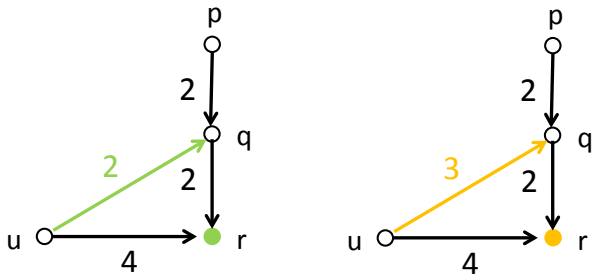


**Fig. 2.** Let $u, r$ be two consecutive vertices on the canonical path say $P_i^\ell$. Let $r$ be on the canonical path $Q_j^{\ell'}$ and let $p$ be the vertex preceding it on this path. If $r$ is a green (respectively orange) vertex then we subdivide the edge $(p, r)$ by introducing a new vertex $q$ and adding two edges $(p, q)$ and $(q, r)$ of weight 2. We also add an edge $(u, q)$ of weight 2 (respectively 3). The idea is if both the edges $(p, r)$ and $(u, r)$ were being used initially then now we can save a weight of 2 (respectively 1) by making the horizontal path choose $(u, q)$ and then we get $(q, r)$ for free, as it is already being used by the vertical canonical path.

Intuitively the $k^2$ gadgets correspond to the $k^2$ sets in the Grid Tiling* instance. Let us denote the gadget which is the intersection of the $a_i \rightsquigarrow b_i$ paths and $c_j \rightsquigarrow d_j$ paths by $G^{i,j}$. If $i = j$, then we call $G^{i,j}$ as a symmetric gadget; else we call it as a asymmetric gadget. We perform the following modifications on the edges inside the gadget: (see Figure 1)

- **Symmetric Gadgets**: For each $i \in [k]$, if $(x, y) \in S_{i,i}$ then we color green the vertex in the gadget $G_{i,i}$ which is the unique intersection of the canonical paths $P_i^x$ and $Q_i^y$. Then we add a shortcut as shown in Figure 2. The idea is if both the $a_i \rightsquigarrow b_i$ path and $c_i \rightsquigarrow d_i$ path pass through the green vertex then the $a_i \rightsquigarrow b_i$ path can save a weight of 2 by using the green edge and a vertical downward edge ((which is already being used by $c_j \rightsquigarrow d_j$ canonical path)) to reach the green vertex, instead of paying a weight of 4 to use the horizontal edge reaching the green vertex.
- **Aymmetric Gadgets**: For each $i \neq j \in [k]$, if $(x, y) \in S_{i,j}$ then we color orange the vertex in the gadget $G_{i,i}$ which is the unique intersection of the canonical paths $P_i^x$ and $Q_i^y$. Then we add a shortcut as shown in Figure 2. The idea is if both the $a_i \rightsquigarrow b_i$ path and $c_j \rightsquigarrow d_j$ path pass through the green vertex then the $a_i \rightsquigarrow b_i$ path can save a weight of 1 by using the orange edge of weight 3 followed by a vertical downward edge (which is already being used by $c_j \rightsquigarrow d_j$ canonical path) to reach the orange vertex, instead of paying a weight of 4 to use the horizontal edge reaching the green vertex.

From Figure 1, it is easy to see that each canonical path has weight equal to $\alpha$.

### 3.1 Vertices and Edges not shown in Figure 1

The following vertices and edges are not shown in Figure 1 for sake of presentation:

– Add two vertices $s$ and $t$.
– For each $1 \leq i \leq k$, add an edge $(s, c_i)$ of weight 0.
– For each $1 \leq i \leq k$, add an edge $(d_i, t)$ of weight 0
– Add edges $(t, a_k)$ and $(b_1, s)$ of weight 0.
– For each $2 \leq i \leq k$, introduce two new vertices $e_i$ and $f_i$. We call these $2k-2$ vertices as bridge vertices.
– For each $2 \leq i \leq k$, add a path $b_i \to e_i \to f_i \to a_{i-1}$. Set the weights of $(b_i, e_i)$ and $(f_i, a_{i-1})$ to be zero.
– For each $2 \leq i \leq k$, set the weight of the edge $(e_i, f_i)$ to be $W$. We call these edges as **connector** edges. The idea is that we will choose $W$ large enough so that each connector edge is used exactly once in an optimum solution for 2-SCSS-$(k, 1)$. We will see later that $W = 53n^9$ suffices for our reduction.

We need a small technical modification: we add one dummy row and column to the Grid Tiling* instance. Essentially we now have a dummy index 1. So neither the first row nor the first column of any $S_{i,j}$ has any elements in the Grid Tiling* instance. That is, no green vertex or orange vertex can be in the first row or first column of any gadget. We now prove two theorems which together give a reduction from Grid Tiling* to 2-SCSS-$(k, 1)$. Let

$$\beta = 2k \cdot \alpha + W(k-1) - (k^2 + k) \tag{1}$$

### 3.2 Grid Tiling* has a solution $\Leftrightarrow$ 2-SCSS-$(2k-1, 1)$ has a solution of weight $\leq \beta$

First we show the easy direction.

**Theorem 4.** [$\star$] *Grid Tiling* has a solution implies OPT for 2-SCSS-$(k, 1)$ is at most $\beta$.*

We now prove the other direction which is more involved. First we show some preliminary lemmas:

**Definition 4.** *For each $i \in [k]$, let us call the set of gadgets $\{G_{i,1}, G_{i,2}, \ldots, G_{i,k}\}$ as the gadgets of level $i$.*

**Lemma 5.** [$\star$] *In any optimum solution for 2-SCSS-$(k, 1)$, the $t \rightsquigarrow s$ path*

– *Must use all the $k-1$ connector edges*
– *Contains an $a_i \rightsquigarrow b_i$ path (which does not include any connector edge) for each $i \in [k]$*

**Lemma 6.** [$\star$] *In the optimum solution exactly $k$ of the $s \rightsquigarrow t$ paths cannot use any connector edge.*

We call the $s \rightsquigarrow t$ paths described in Lemma 6 as **expensive** paths. Note that the only outgoing edges from $s$ are to $\{c_1, c_2, \ldots, c_k\}$ and the only incoming edges into $t$ are from $\{d_1, d_2, \ldots, d_k\}$. So, we can think of the expensive paths as actually $k$ paths from $\{c_1, c_2, \ldots, c_k\}$ to $\{d_1, d_2, \ldots, d_k\}$. Since expensive edges do not use any connector edge, the existence of a $c_j \rightsquigarrow d_\ell$ path implies $\ell \geq j$.

**Definition 5.** *For each $i \in [k]$, let $\lambda_i, \mu_i$ denote the number of $c_i \rightsquigarrow d_i, c_i \rightsquigarrow \{d_{i+1}, d_{i+2}, \ldots, d_k\}$ expensive paths in the optimum solution.*

We know that $\sum_{i=1}^{k} \lambda_i \leq k$ and $\lambda_j \geq 0$ for each $j \in [k]$.

**Lemma 7.** [$\star$] *For each $i \in [k]$, the sum of weights of blue edges incident on $a_i$ and $b_i$ on the $a_i \rightsquigarrow b_i$ path in any optimum solution is at least $\Delta(nk+1)$.*

**Lemma 8.** [$\star$] *The sum of weights of the blue edges in any expensive path is at least $\Delta(nk+1)$, with equality iff the path is canonical.*

**Lemma 9.** [$\star$] *In any optimum solution, the weight of blue edges is at least $2k \cdot \Delta(nk+1)$ and the weight of black edges is at least $2k\Big(4(k+1)+4k(n-1)\Big)$*

**Lemma 10.** [$\star$] *In any optimum solution, the weight of black edges is at least $2k\Big(4(k+1)+4k(n-1)\Big)$, without considering the savings via orange and green edges (see Figure 2).*

**Lemma 11.** [$\star$] *Every expensive path is canonical, i.e., $\mu_j = 0$ for all $j \in [k]$.*

Note the shortcuts described in Figure 2 again bring the $a_i \rightsquigarrow b_i$ path back to the same horizontal canonical path.

**Definition 6.** *We call an $a_i \rightsquigarrow b_i$ path as an* almost canonical *path if it is basically a canonical path, but can additionally take the small detour given by the green or orange edges in Figure 2. An almost canonical path must however end on the same horizontal level on which it began.*

**Lemma 12.** [$\star$] *For each $i \in [k]$, the optimum solution contains an almost canonical $a_i \rightsquigarrow b_i$ path .*

**Theorem 5.** [$\star$] *OPT for 2-SCSS-$(k,1)$ is at most $\beta$ implies the* Grid Tiling$^*$ *instance has a solution.*

Appendix E.11 finishes the proof of Theorem 2, and hence shows that $n^{O(k)}$ algorithm for 2-SCSS-$(k,1)$ given in Section 2 is optimal.

## 4 Conclusions

In this paper, we studied the 2-SCSS-$(k,1)$ problem and presented an algorithm which finds an optimum solution in time $n^{\mathcal{O}(k)}$. This algorithm was based on the fact that there always exists an optimal solution for 2-SCSS-$(k,1)$ that has the reverse-compatibility property. However, we show in Appendix F that the 2-SCSS-$(k_1,k_2)$ problem need not always have a solution which satisfies the reverse-compatibility property. Therefore, it remains an important challenging problem to find a similar structure and generalize our method to solve the 2-SCSS-$(k_1,k_2)$ problem.

Furthermore, we can show that 2-SCSS-$(k,1)$ is W[1]-hard parameterized by $k$ and under the ETH it cannot be solved in $f(k) \cdot n^{o(k)}$ for any function $f$.

## References


1. 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)
2. Chakrabarty, D., Chekuri, C., Khanna, S., Korula, N.: Approximability of capacitated network design. In: IPCO. pp. 78–91 (2011)
3. Charikar, M., Chekuri, C., Cheung, T.Y., Goel, A., Guha, S., Li, M.: Approximation Algorithms for Directed Steiner Problems. J. Algorithms 33(1) (1999)
4. Chen, J., Huang, X., Kanj, I.A., Xia, G.: Strong Computational Lower Bounds via Parameterized Complexity. J. Comput. Syst. Sci. 72(8), 1346–1367 (2006)
5. Chitnis, R.H., Hajiaghayi, M., Marx, D.: Tight Bounds for Planar Strongly Connected Steiner Subgraph with Fixed Number of Terminals (and Extensions). In: SODA. pp. 1782–1801 (2014)
6. Dodis, Y., Khanna, S.: Design Networks with Bounded Pairwise Distance. STOC'99
7. Dreyfus, S.E., Wagner, R.A.: The Steiner Problem in Graphs. Networks 1(3) (1971)
8. 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)
9. Goemans, M.X., Goldberg, A.V., Plotkin, S.A., Shmoys, D.B., Tardos, É., Williamson, D.P.: Improved approximation algorithms for network design problems. In: SODA. pp. 223–232 (1994)
10. Guo, C., Lu, G., Li, D., Wu, H., Shi, Y., Zhang, D., Zhang, Y., Lu, S.: Hybrid butterfly cube architecture for modular data centers (Nov 22 2011), `http://www.google.com/patents/US8065433`, uS Patent 8,065,433
11. Guo, J., Niedermeier, R., Suchý, O.: Parameterized Complexity of Arc-Weighted Directed Steiner Problems. SIAM J. Discrete Math. 25(2), 583–599 (2011)
12. Hakimi, S.L.: Steiner's Problem in Graphs and its Implications. Networks 1 (1971)
13. Halperin, E., Krauthgamer, R.: Polylogarithmic Inapproximability. STOC '03
14. Impagliazzo, R., Paturi, R., Zane, F.: Which Problems Have Strongly Exponential Complexity? J. Comput. System Sci. 63(4), 512–530 (2001)
15. Marx, D.: On Optimality of Planar & Geometric Approximation Schemes. FOCS'07
16. Marx, D.: A tight lower bound for planar multiway cut with fixed number of terminals. In: ICALP (1). pp. 677–688 (2012)
17. Marx, D., Pilipczuk, M.: Everything you always wanted to know about the parameterized complexity of Subgraph Isomorphism (but were afraid to ask). In: STACS. pp. 542–553 (2014)
18. Natu, M., Shu-Cherng, F.: The point-to-point connection problem analysis and algorithms. Discrete Applied Mathematics 78(13), 207 – 226 (1997)
19. Ramachandran, K., Kokku, R., Mahindra, R., Rangarajan, S.: Wireless network connectivity in data centers (Jul 8 2010), `http://www.google.com/patents/US20100172292`, uS Patent App. 12/499,906
20. Ramanathan, S.: Multicast tree generation in networks with asymmetric links. IEEE/ACM Transactions on Networking (TON) 4(4), 558–568 (1996)
21. Teixeira, R., Marzullo, K., Savage, S., Voelker, G.M.: Characterizing and measuring path diversity of internet topologies. In: SIGMETRICS. pp. 304–305 (2003)
22. Teixeira, R., Marzullo, K., Savage, S., Voelker, G.M.: In search of path diversity in isp networks. In: Internet Measurement Comference. pp. 313–318 (2003)
23. Zelikovsky, A.: A Series of Approximation Algorithms for the Acyclic Directed Steiner Tree Problem. Algorithmica 18(1), 99–110 (1997)

## A The Directed Steiner Forest and Directed Steiner Network problems

Another standard generalization of the DST problem is the Directed Steiner Forest (DSF) problem. In the $p$-DSF problem, given a directed graph $G = (V, E)$ and a set $T = \{(s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p)\}$ of $p$ pairs of terminals, the objective is to find a set $S \subseteq V$ such that $G[S]$ contains an $s_i \to t_i$ path for each $1 \leq i \leq p$. For the more general DSF problem, the best known approximation ratio in polynomial time is $n^{2/3+\epsilon}$ for any $\epsilon > 0$ due to Berman et al. [1]. Dodis and Khanna [6] showed that DSF has no $\Omega(2^{\log^{1-\epsilon} n})$-approximation for any $0 < \epsilon < 1$, unless NP has a quasi-polynomial time algorithm.

By adding a demand requirement between each pair, we obtain a well-known generalization of the DSF problem which is known as the Directed Steiner Network (DSN) problem. Formally, in the $p$-DSN problem, given a directed graph $G = (V, E)$ and a set $T = \{(s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p)\}$ of $p$ pairs of terminals and a set of demands $\{d_1, d_2, \ldots, d_p\}$, the objective is to find a set $S \subseteq V$ such that $G[S]$ contains $d_i$ disjoint $s_i \rightsquigarrow t_i$ paths for each $1 \leq i \leq p$. The DSN problem is known to be notoriously hard: it admits no known non-trivial approximation. Essentially, the big jump in hardness from DSF to DSN comes from the introduction of the requirement of having a demand of $d_i$ *disjoint* paths between each pair $(s_i, t_i)$. Note that we can check in polynomial time whether the given instance of DSN has a feasible solution or not: for each pair $(s_i, t_i)$ check whether the flow from $s_i$ to $t_i$ is at least $d_i$.

## B Previous Work on Exact Algorithms for Directed Steiner Problem

Rather than finding approximate solutions in polynomial time, one can look for exact solutions in time that is still better than the running time obtained by brute force solutions. Brute force can be used to check in time $n^{O(\ell)}$ if a solution of size at most $\ell$ exists: one can go through all sets of size at most $\ell$. Recall that a problem is *fixed-parameter tractable* (FPT) with a particular parameter $\ell$ if it can be solved in time $f(\ell) \cdot n^{O(1)}$, where $f$ is an arbitrary function depending only on $\ell$. One can also consider parameterization by the number $p$ of terminals (terminal pairs); with this parameterization, it is not even clear if there is a polynomial-time algorithm for every fixed $p$, much less if the problem is FPT. It is known that Steiner Tree on undirected graphs is FPT: the classical algorithm of Dreyfus and Wagner [7] solves the problem in time $3^p \cdot n^{O(1)}$, where $p$ is the number of terminals. The same algorithm work for Directed Steiner Tree as well.

For the SCSS and DSF problems, we cannot expect fixed-parameter tractability: Guo et al. [11] showed that SCSS is W[1]-hard parameterized by the number of terminals $p$, and DSF is W[1]-hard parameterized by the number of terminal pairs $p$. In fact, it is not even clear how to solve these problems in polynomial time for small fixed values of the number $p$ of terminals/pairs. The case of $p = 1$

in DSF is the well-known shortest path problem in directed graphs, which is known to be polynomial time solvable. For the case $p = 2$ in DSF, Natu and Fang [18] gave an algorithm running in $O(mn + n^2 \log n)$ time. The question regarding the existence of a polynomial algorithm for DSF when $p = 3$ was open. In a paper in [FOCS '99, SICOMP '06] Feldman and Ruhl solved this question by giving an $n^{O(p)}$ algorithm for $p$-DSF, where $p$ is the number of terminal pairs [8]. They first designed an $n^{O(p)}$ algorithm for $p$-SCSS, and used it as a subroutine in the algorithm for the more general DSF problem. In a paper in [SODA '14] Chitnis, Hajiaghayi and Marx improved on these results by showing that $p$-SCSS can be solved in $2^{O(p \log p)} \cdot n^{O(\sqrt{p})}$ time, when the underlying undirected graph of $G$ is planar (or more generally, $H$-minor-free for any fixed graph $H$) [5]. They also showed a matching hardness result: the $p$-SCSS on planar graphs cannot be solved in time $f(p) \cdot n^{o(\sqrt{p})}$ (for any function $f$) unless the Exponential Time Hypothesis (ETH) fails[7]. Chitnis et al. [5] also showed that the Feldman-Ruhl algorithm for $p$-DSF running in $n^{O(p)}$ is optimal by proving that $p$-DSF does not admit an $f(p) \cdot n^{o(p)}$ algorithm for any function $f$, unless the ETH fails.

## C Equivalence of Vertex-Weighted and Edge-Weighted Versions of 2-SCSS-$(k, 1)$

**Lemma 13.** *The edge-weighted* 2-*SCSS*-$(k_1, k_2)$ *and the vertex-weighted* 2-*SCSS*-$(k_1, k_2)$ *are equivalent.*

*Proof.* First, we show that every instance of the edge-weighted 2-SCSS-$(k_1, k_2)$ can be reduced to an instance of the vertex-weighted 2-SCSS-$(k_1, k_2)$. Let $G$ be an edge weighted graph. We replace each edge of $G$ with a path of length 2 such that the middle vertex weights is that of the corresponding edge. We leave the weight of the other vertices to be zero. Clearly, this change preserves the weight of the paths.

Next, we provide a reduction from the vertex-weighted 2-SCSS-$(k_1, k_2)$ to the edge-weighted 2-SCSS-$(k_1, k_2)$. Let $G$ be a vertex-weighted graph. We replace each vertex $v$, with a pair of vertices $(v_{in}, v_{out})$ and add an edge from $v_{in}$ to $v_{out}$ with weight equal to weight of $v$. We connect the incoming edges of $v$ to $v_{in}$ and the outcoming edges to $v_{out}$. Again, this reduction preserves the weight of paths which completes the proof of the lemma. □

## D Omitted Proofs from Section 2

### D.1 Proof of Lemma 1

*Proof.* In order to prove this lemma, we introduce the notion of the rank of a solution for 2-SCSS-$(k, 1)$. Later, we show that an optimum solution of 2-SCSS-$(k, 1)$ with the minimum rank is reverse-compatible.

[7] Recall that ETH can be stated as the assumption that $n$-variable 3SAT cannot be solved in time $2^{o(n)}$ [14]

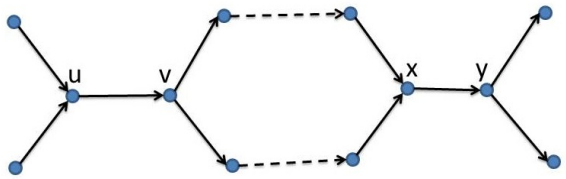


**Fig. 3.** The top path is $B$ and the bottom path is $F$. $F$ and $B$ share $u \to v$ and $x \to y$, and both contain $u \to y$ sub-paths.

**Definition 7.** *Let $\boldsymbol{F} = \{F_1, F_2, \ldots, F_k\}$ be a set of paths form $s \rightsquigarrow t$ and $B$ be a path from $t \rightsquigarrow s$. Let $d_i$ be the number of maximal sub-paths that $B$ and $F_i$ share. The* rank *of $(\boldsymbol{F}, B)$ is given as*

$$\mathcal{R}(\boldsymbol{F}, B) = \sum_{i=1}^{k} d_i$$

Let $(\mathbf{F}, B)$ be an optimum solution of 2-SCSS-$(k, 1)$ with the minimum rank. Assume for the sake of contradiction that $(\mathbf{F}, B)$ is not reverse-compatible. Let $F$ be an element of $\mathbf{F}$ such that $(F, B)$ is not path-reverse-compatible. This means that $F$ and $B$ share two maximal sub-paths $u \to v$ and $x \to y$, and at the same time $F$ and $B$ both contain $u \to y$ sub-path (See Figure 3).

We replace the $u \to y$ sub-path of $B$ by the $u \to y$ sub-path of $F$. On one hand, $B$ shares all of the $u \to y$ sub-path with $F$. Thus, this change does not increase the cost of the network, therefore it remains an optimum solution. On the other hand, by this change, the sub-paths $u \to v$ and $x \to y$ join. In addition, since forward paths are edge-disjoint, after the change $B$ shares the whole $u \to y$ sub-path with only $F$. Therefore, this change decreases the rank of the solution. The existence of an optimum solution with a smaller rank contradicts the selection of $(\mathbf{F}, B)$ and completes the proof. □

### D.2 Proof of Lemma 2

*Proof.* Let $\langle v_1, v_2, \ldots, v_{|\mathcal{T}|}\rangle$ denote a state of the game in which token $t_i$ is placed on vertex $v_i$ and $G^*$ be a graph containing $n^{|\mathcal{T}|}$ vertices, where each of its vertices corresponds to one state of the game. For every state $\langle v_1, v_2, \ldots, v_{|\mathcal{T}|}\rangle$ of the game and every move $m \in \mathcal{M}$ which is applicable to $\langle v_1, v_2, \ldots, v_{|\mathcal{T}|}\rangle$, we add an edge from vertex $\langle v_1, v_2, \ldots, v_{|\mathcal{T}|}\rangle$ of $G^*$ to vertex $\langle v_1^*, v_2^*, \ldots, v_{|\mathcal{T}|}^*\rangle$ with weight $\hat{C}(m)$, where $\langle v_1^*, v_2^*, \ldots, v_{|\mathcal{T}|}^*\rangle$ is the state of the game after applying $m$ to $\langle v_1, v_2, \ldots, v_{|\mathcal{T}|}\rangle$.

In order to solve the game, we need to find a sequence of moves which transports all of the tokens from $s$ to $t$ with the minimum cost. This is equivalent

to finding the shortest path from vertex $\langle s, s, \ldots, s\rangle$ of $G^*$ to vertex $\langle t, t, \ldots, t\rangle$ which can be determined with the Dijkstra algorithm. Since the running time of the Dijkstra algorithm is $|E(G^*)| \log |V(G^*)|$, we can find the optimum solution of the game in time $\mathcal{O}(n^{|\mathcal{T}|}|\mathcal{M}| \log(n^{|\mathcal{T}|}))$. □

### D.3 Proof of Lemma 3

*Proof.* After each move, the state of the game changes in one of the following ways:

1. A token $\mathcal{F}_i$ moves through an edge.
2. Token $\mathcal{B}$ moves through an edge in the backward direction.
3. Token $\mathcal{B}$ and a token $\mathcal{F}_i$ swap their positions.

The cost of each move of type 3 is equal to the weight of the shortest path from the position of $\mathcal{F}_i$ to the position of $\mathcal{B}$. Therefore, we assume that in these moves, token $\mathcal{F}_i$ moves to the position of $\mathcal{B}$ through the shortest path and token $\mathcal{B}$ goes back to the position of $\mathcal{F}_i$ along the same path in the opposite direction. Note that, token $\mathcal{B}$ always traverses the edges in the opposite direction, therefore we can assume that token $\mathcal{B}$ traverses a path from $t$ to $s$ in the backward direction.

Let $p_i$ be the walk that token $\mathcal{F}_i$ traverses from $s$ to $t$ and $q$ be the walk from $t$ to $s$ that $\mathcal{B}$ traverses in backward direction. The total cost of the game is equal to

$$w(p_1) + w(p_2) + \ldots + w(p_k) + w'$$

where $w(p_i)$ is the length of the path $p_i$ and $w'$ is the sum of all moves of type 2. Therefore we pay at least $\max\{f^*(e), b^*(e)\}$ times the weight of each edge $e$ where $f^*(e)$ and $b^*(e)$ denote the number of occurrences of $e$ in $\{p_1, p_2, \ldots, p_k\}$ and $q$, respectively. Thus, the cost of the game is at least $\mathcal{C}(p_1, p_2, \ldots, q)$, hence

$$opt(I) \leq opt(Cor(I)).$$

□

### D.4 Proof of Lemma 4

*Proof.* In order to prove this lemma we use Lemma 1 which states there exists an optimal solution for $I$ which is reverse-compatible; Let it be $(F_1, F_2, \ldots, F_k, B)$. We provide a solution for $Cor(I)$ with the total cost equal to $\mathcal{C}(F_1, F_2, \ldots, F_k, B)$.

Let $\{r_1, r_2, \ldots, r_d\}$ be the set of maximal sub-paths of $B$ which are shared with paths $F_1, F_2, \ldots, F_k$.

Initially all of the tokens are placed on vertex $s$. While token $\mathcal{B}$ has not reached vertex $t$, we do the following:

– We move token $\mathcal{B}$ along the path $B$ in the opposite direction with moves of type (2) until it arrives at the end of a sub-path in $\{r_1, r_2, \ldots, r_d\}$ or reaches the vertex $t$. In the former case, let $r_i$ be the sub-path that $\mathcal{B}$ is standing on its end and $F_j$ be the path that shares $r_i$ with $B$. We move token $\mathcal{F}_j$ to the

beginning of the sub-path $r_i$ using moves of type (1) and swap the positions of the tokens $\mathcal{B}$ and $\mathcal{F}_j$. In the latter case, we move each token $\mathcal{F}_i$ along the path $F_i$ with moves of type (1) until it reaches vertex $t$.

Since each pair of paths $(F_i, B)$ is reverse-compatible, all of the tokens $\mathcal{F}_i$ traverse sub-paths $r_1, r_2, \ldots, r_d$ with moves of type (3), therefore the total cost of the moves is

$$w(F_1) + w(F_2) + \ldots + w(F_k) + w(B) - w(\hat{b}_1) - \ldots - w(\hat{b}_{|B|})$$

where $w(x)$ is the length of the path $x$. This is equal to $\mathcal{C}(F_1, F_2, \ldots, F_k, B)$. Therefore, we have $opt(I) \geq opt(Cor(I))$. □

### D.5 Proof of Theorem 3

*Proof.* Let $I$ be an instance of the 2-SCSS-$(k, 1)$. According to Lemmas 3 and 4, we have $opt(I) = opt(Cor(I))$.

Since the number of moves in $\mathcal{M}$ is $\mathcal{O}(n^2 d)$, by Lemma 2 we can solve $Cor(I)$ in time $\mathcal{O}(n^{|\mathcal{T}|}|\mathcal{M}|\log(n^{|\mathcal{T}|}))$ which is $\mathcal{O}(n^{\mathcal{O}(k)})$. Let $F_i$ be the path of token $\mathcal{F}_i$ and $B$ be the path that token $\mathcal{B}$ traverses in the opposite direction in an optimal solution of $Cor(I)$. Since $opt(I) = opt(Cor(i))$, $\{F_1, F_2, \ldots, F_k B\}$ is an optimal solution for $I$. □

## E Omitted Proofs from Section 3

### E.1 Proof of Theorem 4

*Proof.* For each $1 \leq i, j \leq k$ let $s_{i,j} \in S_{i,j}$ be the vertex in the solution of the Grid Tiling* instance. Therefore for every $i \in k$ we know that each of the $k$ vertices $s_{i,1}, s_{i,2}, \ldots, s_{i,k}$ have the same $x$-coordinate, say $\delta_i$. Similarly for every $j \in [k]$ each of the $k$ vertices $s_{1,j}, s_{2,j}, \ldots, s_{k,j}$ has the same $x$-coordinate, say $\gamma_j$. We use the following path for the $t \rightsquigarrow s$ path in our solution:

- First use the edge $(t, a_k)$. This incurs weight 0.
- For each $k \geq i \geq 2$, use the canonical $a_i \rightsquigarrow b_i$ path $P_i^{\delta_i}$ followed by the path $b_i \to e_i \to f_i \to a_{i-1}$. This way we reach $a_1$. Finally use the canonical path $P_1^{\delta_1}$ to reach $b_1$. The total weight of these edges is $\alpha \cdot k + W(k-1)$.
- Finally use the edge $(b_1, s)$ of weight 0.

Therefore, with a total cost of $\alpha \cdot k + W(k-1)$ we have an $t \rightsquigarrow s$ path. Since we have used all the $k-1$ connector edges in the $t \rightsquigarrow s$ path, we can now use them for free in $s \rightsquigarrow t$ paths. In particular, we get $k-1$ $s \rightsquigarrow t$ paths given by $s \to e_i \to f_i \to t$ for each $2 \leq i \leq k$. Note that the total cost of these $k-1$ $s \rightsquigarrow t$ paths is 0, since for each $2 \leq i \leq k$ the edge $(e_i, f_i)$ is obtained for free (since it was used in the $t \rightsquigarrow s$ path) and both the edges $(s, e_i)$ and $(f_i, t)$ have weight 0.

Now, for each $j \in [k]$, we add the canonical $c_j \rightsquigarrow d_j$ path $Q_j^{\gamma_j}$. For each $j \in [k]$, note that the edges $(s, c_j)$ and $(d_j, t)$ have weight 0. Hence, for each

$j \in [k]$ we get an $s \rightsquigarrow t$ path whose weight is exactly equal to $\alpha$. However, now the canonical paths will encounter a green or orange vertex in each gadget (depending on whether the gadget is symmetric or asymmetric). As shown in Figure 2, we can save 2 in every symmetric gadget and 1 in every asymmetric gadget. Since number of symmetric gadgets is $k$ and number of asymmetric gadgets is $(k^2 - k)$, we save a total weight of $2k + (k^2 - k) = (k^2 + k)$.

Hence, the total weight of the solution is equal to $\Big(\alpha \cdot k + W(k-1)\Big) + \alpha \cdot k - (k^2 + k) = \beta$. □

### E.2 Proof of Lemma 5

*Proof.* The only outgoing edge from $t$ is $(t, a_k)$ and the only incoming edge into $s$ is $(b_1, s)$. Hence, the $t \rightsquigarrow s$ is essentially a path from $a_k \rightsquigarrow b_1$. Since the edges in the gadgets are oriented downwards and rightwards, the only way to reach a gadget of level $i-1$ from a gadget of level $i$ is to go to the vertex $b_i$ and then use the path $b_i \to e_i \to f_i \to a_{i-1}$. That is, we must use all the $(k-1)$ connector edges which are given by $(e_i, f_i)$ for each $2 \leq i \leq k$.

The above argument also implies that we have an $a_i \rightsquigarrow b_i$ path for each $2 \leq i \leq k$. Since the only coming incoming edge into $s$ is $(b_1, s)$ we must also have an $a_1 \rightsquigarrow b_1$ path in the solution. Therefore, the $t \rightsquigarrow s$ path contains an $a_i \rightsquigarrow b_i$ path for each $i \in [k]$. Suppose there is some $i \in [k]$ such that the $a_i \rightsquigarrow b_i$ path used in the $t \rightsquigarrow s$ path uses any connector edge. Since we have already used the connector edges to go from a gadgets of a certain level to gadgets of a level above it, we now need to pay again for this connector edge. Therefore, the weight of the optimum solution is at least $W(k-1) + W$. We show below that this is greater than $\beta$.

$$
\begin{aligned}
\beta &= W(k-1) + 2k\Big(\Delta(nk+1) + 4(k+1) + 4k(n-1)\Big) - (k^2+k) \\
&\leq W(k-1) + 2k\Big(\Delta(nk+1) + 4(k+1) + 4k(n-1)\Big) \\
&= W(k-1) + 2k\Big(7n^6(nk+1) + 4(k+1) + 4k(n-1)\Big) \quad [\text{Since } \Delta = 7n^6] \\
&\leq W(k-1) + 2n\Big(7n^6(2n^2) + 4(2n) + 4n^2\Big) \quad [\text{Since } k \leq n] \\
&\leq W(k-1) + 2n\Big(14n^8 + 8n^8 + 4n^8\Big) \\
&= W(k-1) + 52n^9 \\
&\leq W(k-1) + 53n^9 \\
&= W(k-1) + W \quad [\text{Since } W = 53n^9]
\end{aligned}
$$

Contradiction. □

### E.3 Proof of Lemma 6

*Proof.* We can get at most $(k-1)$ $s \rightsquigarrow t$ paths for free given by $s \to e_i \to f_i \to t$ since all the $(k-1)$ connector edges have been used by the $t \rightsquigarrow s$ path. Let

us call these paths as **cheap** paths. We now claim that the optimum solution uses all the $(k-1)$ cheap paths. Suppose not. Let $P$ be the $s \rightsquigarrow t$ path used in optimum instead of a cheap path. Note that the only outgoing edges from $s$ are to $\{c_1, c_2, \ldots, c_k\}$ and the only incoming edges into $t$ are from $\{d_1, d_2, \ldots, d_k\}$. Moreover, the $t \rightsquigarrow s$ path in the optimum does not use any blue edge incident on $\{c_1, c_2, \ldots, c_k\}$ or $\{d_1, d_2, \ldots, d_k\}$ since that either brings us back to $t$ or we have already reached $s$. Hence, $P$ pays now at least two blue edges. Replacing $P$ by a cheap path gives a solution of smaller weight than optimum, which is a contradiction. Therefore, the optimum solution contains exactly $(k-1)$ cheap $s \rightsquigarrow t$ paths.

Suppose there is another $s \rightsquigarrow t$ path uses a connector edge. Since there are exactly $(k-1)$ connector edges, some connector edge is used by two different $s \rightsquigarrow t$ paths. Hence, the weight of the optimum solution is $\geq W(k-1) + W = Wk > \beta$, which is a contradiction. □

### E.4 Proof of Lemma 7

*Proof.* From Lemma 5, for each $i \in [k]$ we know that any optimum solution contains an $a_i \rightsquigarrow b_i$ path which does not include any connector edge, i.e., the edges of this $a_i \rightsquigarrow b_i$ path are contained among the gadgets of level $i$. We must use at least one blue edge incident on $a_i$ and one blue edge incident on $b_i$. Let the blue edges incident on $a_i, b_i$ be from the canonical paths $P_i^{\ell}, P_i^{\ell'}$. Since the edges in gadgets are oriented downwards and rightwards, it follows that $\ell' \geq \ell$. Hence the sum of weights of the blue edges is given by $\Delta(nk-ni+n+1-\ell)+\Delta(ni-n+\ell') = \Delta(nk+1) + (\ell' - \ell) \geq \Delta(nk+1)$. □

### E.5 Proof of Lemma 8

*Proof.* Suppose the expensive path is $c_j \rightsquigarrow d_\ell$ path. Since expensive paths do not use connector edges, we have $\ell \geq j$. We consider two cases: $\ell = j$ and $\ell > j$. Suppose $\ell > j$. The minimum weights of any blue edges incident on $c_j, d_\ell$ are $\Delta(nk-nj+1), \Delta(n\ell-n+1)$ respectively. Hence, the sum of weights of these edges is $\Delta(nk-nj+1)+\Delta(n\ell-n+1) = \Delta(nk+1)+\Delta+\Delta(n(\ell-j-1)) \geq \Delta(nk+2)$.

If $\ell = j$, then let the blue edges incident on $c_j, d_j$ be from the canonical paths $Q_j^r, Q_j^{r'}$. Since expensive paths do not use connector edges, we have $r' \geq r$. The weight of blue edges incident on $c_j$ from canonical path $Q_j^r$ is $\Delta(nk-nj+n+1-r)$ and the weight of the blue edge incident on $d_j$ from the canonical path $Q_j^{r'}$ is $\Delta(nj - n + r')$. Hence, the sum of weights of these edges is $\Delta(nk - nj + n + 1 - r) + \Delta(nj - n + r') = \Delta(nk+1) + \Delta(r' - r)) \geq \Delta(nk+1)$, with equality if and only if the path is canonical (recall an expensive path does not use any connector edges).

□

### E.6 Proof of Lemma 9

*Proof.* From Lemma 7, we know that the sum of weights of blue edges incident on $a_i$ and $b_i$ on the $a_i \rightsquigarrow b_i$ path in any optimum solution is at least $\Delta(nk+1)$

for each $i \in [k]$. From Lemma 8, we know that the sum of weights of the blue edges in any expensive path is at least $\Delta(nk+1)$. Moreover, these blue edges are incident on some $c_j$ and $d_\ell$ for some $k \geq \ell \geq j \geq 1$. Hence, the total weight of blue edges is at least $2k \cdot \Delta(nk+1)$. □

### E.7 Proof of Lemma 10

*Proof.* From Lemma 5, we know that for each $i \in [k]$ there is an $a_i \rightsquigarrow b_i$ path in the optimum solution which does not include any connector edge. Hence, the edges of this $a_i \rightsquigarrow b_i$ path are contained in the gadgets of level $i$. Hence, we need to at least buy the set of horizontally right black edges which take us from $a_i$ to $b_i$. These black edges have cost $4(k+1) + 4k(n-1)$. Since the edges of the $a_i \rightsquigarrow b_i$ paths are contained in the gadgets of level $i$ and the sets of horizontally right black edges in gadgets of different levels are disjoint, the total weight of horizontally right black edges is at least $k\Big(4(k+1) + 4k(n-1)\Big)$. Similarly, let $c_j \rightsquigarrow d_\ell$ be an expensive path for some $\ell \geq j$. Again, we need to at least buy at least the set of vertically downward black edges which take us from $c_j$ to $d_\ell$. These vertically downward black edges have total cost $4(k+1) + 4k(n-1)$. Even though two expensive paths may use the same vertically downward edges, they are both to be used in $s \rightsquigarrow t$ paths and hence we must pay for them each time. Hence, the total weight of the horizontally right black edges is at least $k\Big(4(k+1) + 4k(n-1)\Big)$. Combining the two observations above, we get that the total weight of black edges (horizontally right and vertically downward) in the optimum solution is $2k\Big(4(k+1) + 4k(n-1)\Big)$. □

### E.8 Proof of Lemma 11

*Proof.* Suppose an expensive path is not canonical. Hence, from Lemma 8, the contribution of the blue edges of this expensive path is $\geq \Delta(nk+2)$. From Lemma 9, it follows that the contribution of the blue edges to the optimum is at least $2k \cdot \Delta(nk+1) + \Delta$.

Refer to Figure 2. Note that we can use each shortcut at most $\binom{k+1}{2}$ times, once for each pair of paths that will meet at the orange or green vertex (note that there are total $k+1$ paths ). There are $k \cdot n$ green edges ($n$ in each of the $k$ symmetric gadgets). Since each green shortcut can save a weight of 2, we can save at most $2k \cdot n$ from the green edges. Note that in the asymmetric gadgets, there are no shortcuts along the diagonal. Hence, an asymmetric gadget can have at most $(n^2 - n)$ orange edges. There are $(k^2 - k)$ asymmetric gadgets and we can save a weight of 1 from each orange edge. So, we can save at most $(n^2 - n)(k^2 - k)$ from the orange edges. Hence, total maximum saving is

$$\begin{aligned}
\binom{k+1}{2}\Big(2k \cdot n + (n^2 - n)(k^2 - k)\Big) &\leq \frac{4k^2}{2} \cdot (2n^2 + n^4) \qquad [\text{Since } k \leq n] \\
&\leq 2k^2 \cdot (3n^4) \\
&\leq 6n^6
\end{aligned}$$

We now claim that the cost of our solution exceeds $\beta$, even if we allow this maximum possible saving. Recall that we have cost of $W(k-1)$ from the connector edges. Hence, the cost of our optimum solution is at least

$$\begin{aligned}
\text{OPT} &\geq W(k-1) + 2k \cdot \Delta(nk+1) + \Delta + 2k\Big(4(k+1) + 4k(n-1)\Big) - 6n^6 \\
&= W(k-1) + 2k \cdot \Delta(nk+1) + 2k\Big(4(k+1) + 4k(n-1)\Big) + \Big(\Delta - 6n^6\Big) \\
&= W(k-1) + 2k \cdot \Delta(nk+1) + 2k\Big(4(k+1) + 4k(n-1)\Big) + n^6 \quad [\text{Since } \Delta = 7n^6] \\
&> W(k-1) + 2k \cdot \Delta(nk+1) + 2k\Big(4(k+1) + 4k(n-1)\Big) \\
&> W(k-1) + 2k \cdot \Delta(nk+1) + 2k(4(k+1) + 4k(n-1)) - (k^2 - k) \\
&= \beta \quad [\text{From Equation 1}]
\end{aligned}$$

Contradiction. □

### E.9 Proof of Lemma 12

*Proof.* Fix some $i \in [k]$. From Lemma 5, we know that the $a_i \rightsquigarrow b_i$ path in the optimum solution does not include any connector edge, i.e., this path is completely contained in the gadgets of level $i$. Suppose to the contrary that the $a_i \rightsquigarrow b_i$ path in the optimum solution is not canonical. From the orientation of the edges in the gadgets of level $i$ (rightwards and downwards), we know that there is a $a_i \rightsquigarrow b_i$ path in the optimum solution that starts with the blue edge from $P_i^{\ell}$ and ends with a blue edge from $P_i^{\ell'}$ for some $\ell' > \ell$. Hence, the contribution of these blue edges is $\Delta(nk - ni + n + 1 - \ell) + \Delta(ni - n + \ell') = \Delta(nk+1) + \Delta(\ell' - 1) \geq \Delta(nk+1) + \Delta$. Now, a similar argument as in Lemma 11 can be applied to show that the cost of this optimal solution is greater than $\beta$. Contradiction. □

### E.10 Proof of Theorem 5

*Proof.* By Lemma 11, we know that $\sum_{i=1}^{k} \lambda_i = k$ and $\lambda_i \geq 0$ for each $i \in [k]$. We now claim that $\lambda_i = 1$ for each $i \in [k]$.

Let our optimum solution be $\mathcal{X}$. By Lemma 11 and Lemma 12, we know that $\mathcal{X}$ contains

- An $a_i \rightsquigarrow b_i$ almost canonical path for every $1 \leq i, j \leq k$.
- $k$ canonical expensive paths.

In addition, $\mathcal{X}$ contains $(k-1)$ connector edges. For the moment let us forget the shortcuts we did in Figure 2. The weight of $\mathcal{X}$, without considering the shortcuts from Figure 2, is equal to $W(k-1) + 2k\Big(\Delta(nk+1) + 4(k+1) + 4k(n-1)\Big) = \beta + (k^2 + k)$. Therefore, we must have at a saving of $\geq (k^2 + k)$ from the orange and green shortcuts.

By Lemma 12, we know that for each $i \in [k]$ there is exactly one $a_i \rightsquigarrow b_i$ path. Moreover it is almost canonical. Recall that only the horizontal edges can save some weight (see Figure 2). Therefore, we can use at most $k$ green edges (one for each symmetric gadget). Each canonical expensive path can use $(k-1)$ orange edges; once for each of the $(k-1)$ asymmetric gadget that it encounters along the way. Suppose we use $\delta$ green edges for some $\delta \leq k$. Then the total saving is $(k-1)\sum_{i=1}^{k} \lambda_i + 2\delta = k(k-1) + 2\delta$. Since we want the total saving to be at least $k(k-1)+2k$, this forces $\delta \geq k$. But, we already know that $\delta \leq k$, and hence $\delta = k$. This forces that $\lambda_i = 1$ for each $i \in [k]$ as follows: If any $\lambda_i = 0$, then we cannot use the green edge in the symmetric gadget $G_{i,i}$. If any $\lambda_i \geq 2$, then some other $\lambda_j = 0$ (since $\sum_{i=1}^{k} \lambda_i = k$) and we return to previous case. Therefore, the total saving is exactly $k(k-1) + 2k$

So, we have that for each $j \in [k]$, there is a canonical $c_j \rightsquigarrow d_j$ path in $\mathcal{X}$, say $Q_j^{\gamma_j}$. Further, $\mathcal{X}$ also contains an $a_i \rightsquigarrow b_i$ almost canonical path for any $i \in [k]$, say $P_i^{\alpha_i}$. The fact that we have a saving of at least $k(k-1) + 2k$ implies we have exactly one intersection in each symmetric gadget and each non-symmetric gadget. By construction of the gadgets, it follows that

- $\gamma_i = \alpha_i$ for each $i \in [k]$
- For each $1 \leq i \neq j \leq k$ there is an edge $(\alpha_i, \gamma_j)$.

That is, the set of values $(\alpha_i, \gamma_j) \in S_{i,j}$ for each $1 \leq i, j \leq k$ form a solution for the Grid Tiling* instance. □

### E.11 Proof of Theorem 2

*Proof.* Theorem 4 and Theorem 5 together imply the W[1]-hardness. They also give a reduction which transforms the problem of $k \times k$ Grid Tiling* into an instance of 2-SCSS-$(k, 1)$ where we want to find $k$ paths from $s \rightsquigarrow t$ and one path from $t \rightsquigarrow s$.

Chen et al. [4] showed for any function $f$ an $f(k)n^{o(k)}$ algorithm for Clique implies ETH fails. Composing the reduction of [15] from Clique to Grid Tiling*, along with our reduction from Grid Tiling* to 2-SCSS-$(k, 1)$, we obtain under ETH there is no $f(k)n^{o(k)}$ algorithm for 2-SCSS-$(k, 1)$ for any function $f$. This shows that the $n^{O(k)}$ algorithm for 2-SCSS-$(k, 1)$ given in Section 2 is optimal. □

## F 2-SCSS-$(k_1, k_2)$ Does Not Always Have Reverse-Compatibility

In the 2-SCSS-$(k_1, k_2)$ problem we want $k_1$ paths from $s \rightsquigarrow t$ and $k_2$ paths from $t \rightsquigarrow s$. So, we define a natural generalization of Definition 1 to reverse-compatibility of a set of forward paths and a set of backward paths as follows.

**Definition 8.** *Let $\boldsymbol{F} = \{F_1, F_2, \ldots, F_{k_1}\}$ be a set of paths form $s$ to $t$ and $\boldsymbol{B} = \{B_1, B_2, \ldots, B_{k_2}\}$ be a set of paths form $t$ to $s$. We say $(\boldsymbol{F}, \boldsymbol{B})$ is reverse-compatible, if for all $1 \leq i \leq k_2$, $(\boldsymbol{F}, B_i)$ is reverse-compatible.*

The following example has a unique optimum solution to 2-SCSS-$(2, 2)$. However, this solution is not reverse-compatible. This shows that Lemma 1 does not hold for the 2-SCSS-$(k_1, k_2)$ problem when both $k_1$ and $k_2$ are greater than 1. This shows that Lemma 1 is in its most general form.

*Example 1.* In Figure 4 each solid edge has weight one and each dashed edge has weight zero. Note that, solid edges are from left to right and dashed edges are generally from right to left. Length of each shortest path from $s$ to $t$ is 7, and there are exactly two such paths. $p_1 := s, v_1, v_2, v_3, v_4, v_5, v_6, t$ and $p_2 := s, u_1, u_2, u_3, u_4, u_5, u_6, t$. Thus, any optimum solution to 2-SCSS-$(2, 2)$ has cost at least 14. In addition, if we select both $p_1$ and $p_2$ we can select two paths $q_1 = t, u_3, u_4, s$ and $q_2 = t, u_1, u_2, v_1, v_2, v_3, v_4, v_5, v_6, u_5, u_6, s$ from $t$ to $s$ without any cost. Therefore, $(\{p_1, p_2\}, \{q_1, q_2\})$ is an optimum solution to 2-SCSS-$(2, 2)$ with cost 14.

If we select paths $p_1$ and $p_2$ as forward paths, $(q_1, q_2)$ is the only pair of backward paths which is free. On the other hand, if we select one of $p_1$ or $p_2$ twice, there is no free backward path. Thus, $(\{p_1, p_2\}, \{q_1, q_2\})$ is the unique optimum solution to 2-SCSS-$(2, 2)$. However, one can see that paths $p_2$ and $q_2$ are not reverse-compatible.

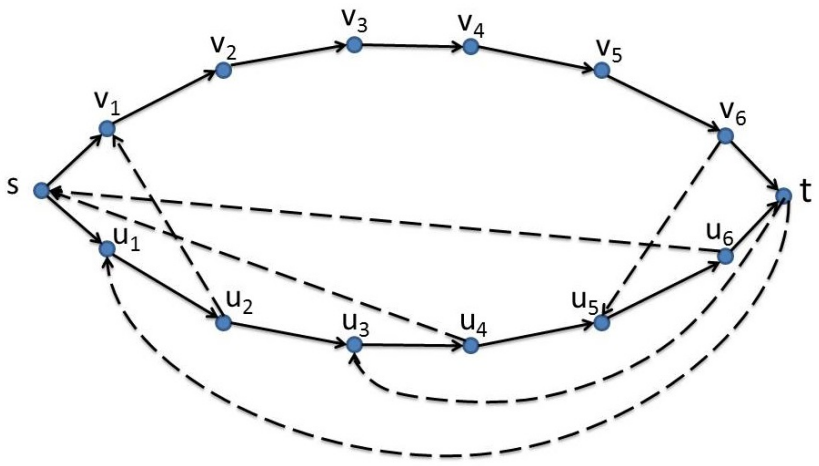


**Fig. 4.** An example of a graph which does not have a reverse-compatible optimum solution for 2-SCSS-$(2, 2)$.