UNIVERSITYOF BIRMINGHAM

A tight algorithm for Strongly Connected Steiner Subgraph on two terminals with demands

Chitnis, Rajesh; Esfandiari, Hossein; Hajiaghayi, Mohammad Taghi; Khandekar, Rohit; Kortsarz, Guy; Seddighin, Saeed

[Link to publication on Research at Birmingham portal](#)

A Tight Algorithm for Strongly Connected Steiner Subgraph On Two Terminals With Demands[*]

Rajesh Chitnis[†]   Hossein Esfandiari[‡]   MohammadTaghi Hajiaghayi[‡]   Rohit Khandekar[§]

Guy Kortsarz[¶]   Saeed Seddighin[‡]

June 25, 2017

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 (p-SCSS) problem is to find an edge set $H \subseteq E$ of minimum weight such that $G[H]$ contains an $t_i \to t_j$ path for each $1 \le i \ne j \le p$. The p-SCSS problem is NP-hard, but Feldman and Ruhl [FOCS '99; SICOMP '06] gave a novel $n^{O(p)}$ algorithm.

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 $\omega : E \to \mathbb{R}^{\ge 0}$, 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 $\phi(e) = \max \left\{ |\{i \in [k_1] : e \in F_i\}| \, , \, |\{j \in [k_2] : e \in B_j\}| \right\}$. For each $k \ge 1$, 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 an optimal solution followed by using the idea of a "token game" similar to that of Feldman and Ruhl. We show with an example that the structural result does not hold for the 2-SCSS-(k_1, k_2) problem if $\min\{k_1, k_2\} \ge 2$. Therefore 2-SCSS-$(k,1)$ is the most general problem one can attempt to solve with our techniques. 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].

---

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 $\omega : E \to \mathbb{R}^{\geq 0}$ and a set $T \subseteq V$ of terminals, the objective is to find a tree S of minimum weight $\omega(S) := \sum_{e \in S} \omega(e)$ which spans all the terminals. The STEINER TREE problem is believed to have been first formally defined by Gauss in a letter in 1836. In the directed version, called the DIRECTED STEINER TREE (DST) problem, 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.

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 [16] that the connection cost in various networks such as satellite or radio networks are not symmetric. Therefore, directed graphs are the most suitable model for such networks. In addition, Ramanathan [16] also used the DST problem to find low-cost multicast trees, which have applications in point-to-multipoint communication in high bandwidth networks. If we require two-way connectivity, then we obtain a generalization of the DST problem known as 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 $H \subseteq E$ of minimum size such that $G[H]$ contains an $t_i \to t_j$ path for each $1 \leq i \neq j \leq p$. The SCSS problem is also NP-hard. The best known approximation ratio in polynomial time for SCSS is $|T|^\varepsilon$ for any $\varepsilon > 0$ due to Charikar et al. [2]. A result of Halperin and Krauthgamer [9] implies SCSS has no $\Omega(\log^{2-\varepsilon} n)$-approximation for any $\varepsilon > 0$, unless NP has quasi-polynomial Las Vegas algorithms. Regarding exact algorithms, Feldman and Ruhl [6] gave a novel $n^{O(p)}$ algorithm for p-SCSS. Chitnis et al. [5] showed that this algorithm is almost tight by the following result: for any computable function f, the p-SCSS problem has no $f(p) \cdot n^{o(p/\log p)}$ algorithm unless the Exponential Time Hypothesis (ETH) fails. Chitnis et al. [5] showed that on certain special graph classes such as planar graphs (and more generally H-minor-free graphs) one can obtain faster algorithms than that of Feldman and Ruhl: more specifically, if the underlying undirected graph is planar, then p-SCSS can be solved in $2^{O(p \log p)} \cdot n^{O(\sqrt{p})}$ time. In addition, Chitnis et al. [5] also showed that this algorithm is optimal: for any computable function f, the existence of a $f(p) \cdot n^{o(\sqrt{p})}$ algorithm for p-SCSS on planar graphs implies ETH fails.

The 2-SCSS-(k_1, k_2) Problem: We now 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 $\phi(e) = \max \left\{ |\{i \in [k_1] : e \in F_i\}| , |\{j \in [k_2] : e \in B_j\}| \right\}$.

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. Teixeira et al. [17, 18] model path diversity in Internet Service Provider (ISP) networks and the Sprint network by disjoint paths between two hosts. There have been several patents [8, 15] attempting to design multiple paths between the components of Google Data Centers.

The 2-SCSS-(k_1, k_2) problem is a special case of the DIRECTED SURVIVABLE NETWORK DESIGN

(DIR-CAP-SNDP) problem [7] in which we are given an directed multigraph with weights and capacities on the edges, and the question is to find a minimum weight 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. [1] and Goemans et al. [7], 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. Note that the sum of demands is $O(k)$. To the best of our knowledge, 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 1.1. *The* 2-SCSS-$(k, 1)$ *problem can be solved in $n^{O(k)}$ time, where n is the number of vertices in the input graph.*

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 [6]), and show that the SOLVING-TOKEN-GAME problem can be solved in $n^{O(k)}$ time. 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 SOLVING-TOKEN -GAME problem which gives an $n^{O(k)}$ time algorithm for the 2-SCSS-$(k, 1)$ problem. This algorithm also generalizes the result of Feldman and Ruhl [6] for 2-SCSS, since 2-SCSS is equivalent to 2-SCSS-$(1, 1)$. In Section 2.4, we show with an example (see Figure 3) 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 that one can attempt to solve with our techniques.

Theorem 1.1 does not rule out the possibility that the 2-SCSS-$(k, 1)$ problem is actually solvable in polynomial time. Our main hardness result rules out this possibility by showing that our algorithm is *tight* in the sense that the exponent of $O(k)$ is best possible.

Theorem 1.2. *The* 2-SCSS-$(k, 1)$ *problem is W[1]-hard parameterized by k. Moreover, under the Exponential Time Hypothesis (ETH) of Impagliazzo and Paturi [10], the* 2-SCSS-$(k, 1)$ *problem cannot be solved in $f(k) \cdot n^{o(k)}$ time for any computable function f where n is the number of vertices in the graph.*

To prove Theorem 1.2, we reduce from the GRID TILING problem formulated in the pioneering work of Marx [11]:

$k \times k$ **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 [5, 12, 13, 14]. Under the ETH, Chen et al. [3] showed that k-CLIQUE[1] does not admit an algorithm running in time $f(k) \cdot n^{o(k)}$ for any function f. Marx [11] 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-$(2k - 1, 1)$. Since the parameter blowup is linear, the $f(k) \cdot n^{o(k)}$ lower bound for GRID TILING from [11] transfers to 2-SCSS-$(k, 1)$. In

---

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

fact, the reduction in [11] 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 $S_{i,i} = \{(j,j) : 1 \le j \le n\}$ for each $i \in [k]$ and $S_{i,f} = \{(j,\ell) : 1 \le j \ne \ell \le n, (v_j, v_\ell) \in E\}$ for each $1 \le i \ne f \le k$. 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.

Before proceeding further, we show that the edge-weighted and the vertex-weighted variants of 2-SCSS-(k_1, k_2) are computationally equivalent. First we define the vertex-weighted variant of 2-SCSS-(k_1, k_2).

Vertex-weighted 2-SCSS-(k_1, k_2)

Input : A vertex-weighted digraph $G = (V,E)$ with weight function $\omega' : V \to \mathbb{R}^{\ge 0}$, 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 $\sum_{v \in V \setminus \{s,t\}} \omega'(v) \cdot \phi'(v)$ is minimized where $\phi'(v) = \max \left\{ |\{i \in [k_1] : v \in F_i\}| \, , \, |\{j \in [k_2] : v \in B_j\}| \right\}$.

Lemma 1.3. *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 the edge-weighted version can be solved using the vertex-weighted version. Let $G = (V,E)$ be an edge-weighted graph with weight function ω. We create a vertex-weighted graph $G' = (V', E')$ with weight function ω' as follows: subdivide each edge (u,v) of G by adding a new vertex $\beta_{u,v}$ to get a path $u \to \beta_{u,v} \to v$ of length two. Let $V' = V \cup \{\beta_{u,v} : (u,v) \in E\}$. Set $\omega'(v) = 0$ for each $v \in V$ and $\omega'(\beta_{u,v}) = \omega(u,v)$ for each edge $(u,v) \in E$. Consider any solution $H \subseteq E$ of edge-weighted 2-SCSS-(k_1, k_2). Consider the solution H' obtained by including (the subdivision) of each edge in H. The weights of all vertices from V is zero in G'. Also, for any edge $e = (u,v)$ it is easy to see that $\phi(e) = \phi'(\beta_{u,v})$, and hence both solutions H and H' have same cost.

Next, we show that the vertex-weighted version can be solved using the edge-weighted version. Let $G' = (V', E')$ be a vertex-weighted graph with weight function ω'. We create an edge-weighted graph $G = (V,E)$ with weight function ω as follows: Replace each vertex $v \in V \setminus \{s,t\}$ with a pair of vertices $(v_{\text{in}}, v_{\text{out}})$. Let $s_{\text{in}} = s = s_{\text{out}}$ and $t_{\text{in}} = t = t_{\text{out}}$. Make all in-neighbors, out-neighbors of v in G incident to $v_{\text{in}}, v_{\text{out}}$ respectively and add an edge $(v_{\text{in}}, v_{\text{out}})$. Set $\omega(v_{\text{in}}, v_{\text{out}}) = \omega'(v)$ for all $v \in V' \setminus \{s,t\}$, and weight of all other edges to be zero. Consider any solution $H' \subseteq V'$ of vertex-weighted 2-SCSS-(k_1, k_2). Consider the solution H obtained as follows: for each $s \rightsquigarrow t$ path in H' say $s = x_1 \to x_2 \to x_3 \to \ldots \to x_{r-1} \to x_r = t$ we add the path $x_1 \to x_{2,\text{in}} \to x_{2,\text{out}} \to x_{3,\text{in}} \to x_{3,\text{out}} \to \ldots \to x_{r-1,\text{in}} \to x_{r-1,\text{out}} \to x_r$. Similarly for the $t \rightsquigarrow s$ path. Also it is easy to see that for any $v \in V' \setminus \{s,t\}$ we have $\phi(v_{\text{in}}, v_{\text{out}}) = \phi'(v)$, and hence both solutions H and H' have same cost. $\qquad \square$

Henceforth we consider only the edge-weighted version of 2-SCSS-(k_1, k_2).

1.2 Notation

The set $\{1, 2, \ldots, n\}$ is denoted by $[n]$. We denote a directed edge from u to v by (u,v) or $u \to v$. A directed path from u to v is denoted by $u \rightsquigarrow v$. Given a directed graph $G = (V,E)$ the in-degree of v is the number of in-neighbors of v and is denoted by $d_G^-(v) = |\{w : (w,v) \in E\}|$. Similarly, the out-degree of v is the number of out-neighbors of v and is denoted by $d_G^+(v) = |\{x : (v,x) \in E\}|$. Given a set S and an integer r, the set S^r denotes the set of all r-element vectors which have each coordinate from S, i.e., $S^r = \{(s_1, s_2, \ldots, s_r) : s_i \in S \, \forall i \in [r]\}$. Similarly, for $s \in S$ we use s^r to denote the vector $(s_1, s_2, \ldots, s_r)$ where $s_i = s$ for each $i \in [r]$.
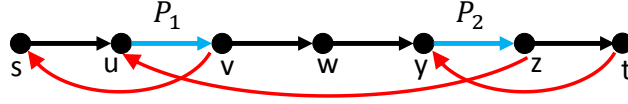
Figure 1: Let F be an $s \rightsquigarrow t$ path given by $s \to u \to v \to w \to y \to z \to t$ and B be an $t \rightsquigarrow s$ path given by $t \to y \to z \to u \to v \to s$. The two paths P_1 and P_2 shown in blue are the maximal common sub-paths between F and B. From Definition 2.2, it follows that F and B are *path-reverse-compatible* since B first sees P_2 and then P_1.

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 some optimal solution of this problem. Next we define a Token Game in Section 2.2 and describe an $n^{O(k)}$ time algorithm for the SOLVING-TOKEN-GAME problem. 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 SOLVING-TOKEN GAME problem.

2.1 Structural Lemma for Some Optimal Solution of 2-SCSS-$(k, 1)$

Remark 2.1. *For simplicity, we replace each edge e of the input graph G with k copies $e_1, e_2, \ldots, e_k$, each having the same weight as that of e. Let the new graph constructed in this way be G'. In G, different $s \rightsquigarrow t$ paths must pay each time they use different copies of the same edge. We can alternately view this as the $s \rightsquigarrow t$ paths in G' being **edge-disjoint**.*

Definition 2.2. (path-reverse-compatible) *Let F be a $s \rightsquigarrow t$ path and B be a $t \rightsquigarrow s$ path. Let $\{P_1, P_2, \ldots, P_d\}$ be the set of maximal sub-paths that F and B share and for all $j \in [d]$, P_j is the j-th sub-path as seen while traversing F. We say the pair (F, B) is path-reverse-compatible if for all $j \in [d]$, P_j is the $(d - j + 1)$-th sub-path that is seen while traversing B, i.e., P_j is the j-th sub-path that is seen while traversing B backward.*

See Figure 1 for an illustration of path-reverse-compatibility.

Definition 2.3. (reverse-compatible) *Let $F = \{F_1, F_2, \ldots, F_r\}$ be a set of $s \rightsquigarrow t$ paths and b be an $t \rightsquigarrow s$ path. We say (F, B) is reverse-compatible, if for all $1 \le i \le r$ 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.

Lemma 2.4. (structural lemma) *There exists an optimum solution for 2-SCSS-$(k, 1)$ which is reverse-compatible.*

Proof. In order to prove this lemma, we first introduce the notion of 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.

Definition 2.5. (rank) *Let $F = \{F_1, F_2, \ldots, F_k\}$ be a set of paths from $s \rightsquigarrow t$, and B be a path from $t \rightsquigarrow s$. For each $i \in [k]$, let d_i be the number of maximal sub-paths that B and F_i share. The rank of (F, B) is given by*

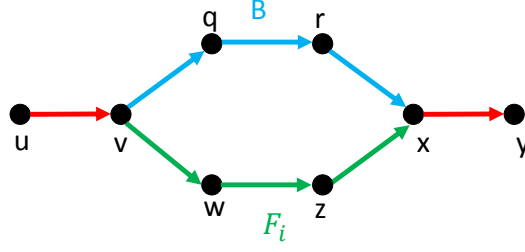$$\mathcal{R}(F, B) = \sum_{i=1}^{k} d_i$$

5

Figure 2: Let the $u \rightsquigarrow y$ sub-path of F_i be a $u \rightsquigarrow v \rightsquigarrow w \rightsquigarrow z \rightsquigarrow x \rightsquigarrow y$ and the $u \rightsquigarrow y$ sub-path of B be $u \rightsquigarrow v \rightsquigarrow q \rightsquigarrow r \rightsquigarrow x \rightsquigarrow y$. From Definition 2.2, it follows that F_i and B are not *path-reverse-compatible* since they both first see $u \rightsquigarrow v$ and then see $x \rightsquigarrow y$.

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, i.e., there exists some $F_i \in \mathbf{F}$ such that (F_i, B) is not path-reverse-compatible. From Definition 2.2, this means that F_i and B share two maximal sub-paths $u \to v$ and $x \to y$, and at the same time F_i and B both contain $u \to y$ sub-paths (see Figure 2).

We replace the $u \to y$ sub-path of B by the $u \to y$ sub-path of F_i. On one hand, B shares all of the $u \to y$ sub-path with F_i. Thus, this change does not increase the weight 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. Hence, d_i decreases by 1. Also, since the forward paths are edge-disjoint, after the change all other d_j's remain same (for $i \neq j$) since B shares the whole $u \to y$ sub-path with only F_i. Therefore, this change strictly decreases the rank of the solution. Existence of an optimum solution with a smaller rank contradicts the selection of $(\mathbf{F}, B)$ and completes the proof. □

2.2 The Token Game

A Token Game is given by $\langle H, h_1, h_2, \mathcal{T}, \mathcal{M}, \hat{C} \rangle$ where

- $H = (V_H, E_H)$ is a graph
- h_1, h_2 are two special vertices in V_H
- $\mathcal{T}$ is a set of tokens
- $\mathcal{M} \subseteq V_H^{|\mathcal{T}|} \times V_H^{|\mathcal{T}|}$ is a set of moves
- $\hat{C} : \mathcal{M} \to \mathbb{R}_{\geq 0}$ is a cost function

We now define a *state* of the Token Game:

Definition 2.6. *A state of the Token Game is an element from the set $V_H^{|\mathcal{T}|}$, i.e., it is a vector of size $|\mathcal{T}|$ where each coordinate comes from H. It gives the location of each token from $\mathcal{T}$.*

A state of the Token Game gives the current location of each token (which vertex of H it is currently on). The start state is $h_1^{|\mathcal{T}|}$, i.e., when all the tokens are the vertex h_1. The end state is $h_2^{|\mathcal{T}|}$, i.e., when all the tokens are at the vertex h_2. The cost function $\hat{C}$ gives the cost of going from one state to another. The goal is to transport all tokens from the start state to the end state with minimum cost. Formally, we have the following problem:

SOLVING-TOKEN-GAME

Input : A token game $\langle H, h_1, h_2, \mathcal{T}, \mathcal{M}, \hat{C} \rangle$

Question: Find a set of moves of minimum cost to go from the start state $h_1^{|\mathcal{T}|}$ to the end state $h_2^{|\mathcal{T}|}$

Now we present an algorithm to solve an instance $\langle H, h_1, h_2, \mathcal{T}, \mathcal{M}, \hat{C} \rangle$ of the Token game in time $O(|\mathcal{M}| + n^{|\mathcal{T}|} \log(n^{|\mathcal{T}|}))$ where n is the number of the vertices of H.

Lemma 2.7. (algorithm for Token Game) *There exists an algorithm for* SOLVING-TOKEN-GAME *which runs in time* $O(|\mathcal{M}| + n^{|\mathcal{T}|} \log(n^{|\mathcal{T}|}))$ *time.*

Proof. We build a *game graph* $\hat{H} = (\hat{V}, \hat{E})$ from H as follows: let $\hat{V} = V_H^{|\mathcal{T}|}$. Recall that $\mathcal{M} \subseteq V_H^{|\mathcal{T}|} \times V_H^{|\mathcal{T}|}$. For each move $M = (\bar{x}, \bar{y}) \in \mathcal{M}$ we add an edge $\bar{x} \to \bar{y}$ in $\hat{H}$ of cost equal to $\hat{C}(M)$.

Note that the starting state $h_1^{|\mathcal{T}|}$ and the end state $h_2^{|\mathcal{T}|}$ are both vertices in $\hat{H}$. By the choice of the edges in $\hat{H}$, it is easy to see that finding a shortest $h_1^{|\mathcal{T}|} \rightsquigarrow h_2^{|\mathcal{T}|}$ path in $\hat{H}$ gives a solution to SOLVING-TOKEN-GAME. We can do this by running Dijkstra's algorithm which takes $O(|E| + |V| \log |V|)$ time on a graph $G = (V, E)$. In our case $|\hat{V}| = n^{|\mathcal{T}|}$ and $|\hat{E}| = |\mathcal{M}|$. Therefore, we can solve SOLVE-TOKEN-GAME in $O(|\mathcal{M}| + n^{|\mathcal{T}|} \log(n^{|\mathcal{T}|}))$ time. $\qquad\square$

2.3 Reduction from 2-SCSS-$(k,1)$ to SOLVING-TOKEN-GAME

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

Let $I = \langle G = (V,E), s, t, \omega \rangle$ be an instance of 2-SCSS-$(k,1)$. We now build a Token Game give by $I' = \langle H, h_1, h_2, \mathcal{T}, \mathcal{M}, \hat{C} \rangle$ where $H = G$, $h_1 = s$, $h_2 = t$ and $\mathcal{T} = \{\mathbf{b}, \mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_k\}$. Note that $|\mathcal{T}| = k+1$. We now describe the set of moves $\mathcal{M}$ and the associated cost function $\hat{C}$. Fix a state $\bar{v} \in V^{k+1}$, say $\bar{v} = (v_0, v_1, v_2, \ldots, v_k)$.

1. **Backward Move**: For every edge $(w, v_1) \in E(G)$, we add a move $(\bar{v}, \bar{w})$ of cost $\omega(w, v_1)$ where

   - $\bar{w} = (w_0, w_1, w_2, \ldots, w_k)$
   - $w_0 = w$
   - $w_i = v_i$ for each $i \in [k]$

2. **Forward Moves**: For every $i \in [k]$ and every edge $(v_i, x) \in E(G)$, we add a move $(\bar{v}, \bar{x})$ of cost $\omega(v_i, x)$ where

   - $\bar{x} = (x_0, x_1, x_2, \ldots, x_k)$
   - $x_i = x$
   - $x_j = v_j$ for each $0 \le j \le k, j \ne i$

3. **Flip Move**: For each $i \in [k]$ we add a move $(\bar{v}, \bar{y})$ of cost equal to that of the shortest $v_i \rightsquigarrow v_0$ path in G where

   - $\bar{y} = (y_0, y_1, y_2, \ldots, y_k)$
   - $y_0 = v_i$
   - $y_i = v_0$
   - $y_j = v_j$ for each $0 \le j \le k, j \notin \{0, i\}$

As in Lemma 2.7, we build a *game graph* $\hat{G} = (\hat{V}, \hat{E})$ from $G = (V, E)$ as follows: let $\hat{G} = V^{k+1}$. Recall that $\mathcal{M} \subseteq V^{k+1} \times V^{k+1}$. For each move $M = (\bar{x}, \bar{y}) \in \mathcal{M}$ we add an edge $\bar{x} \to \bar{y}$ in $\hat{H}$ of cost equal to $\hat{C}(M)$.

We now bound the number of moves in $\mathcal{M}$, which is also equal to the number of edges in $\hat{G}$.

Lemma 2.8. *The number of moves in $\mathcal{M}$ (or equivalently the number of edges in $\hat{G}$) is $n^{O(k)}$,*

7

Proof. Fix a state $\bar{v} \in V^{k+1}$, say $\bar{v} = (v_0, v_1, v_2, \ldots, v_k)$. The number of Backward moves from $\bar{v}$ is $d_G^-(v_0)$ since we add a backward move for each incoming edge into v_0. The number of Forward moves from $\bar{v}$ is $\sum_{i=1}^{k} d_G^+(v_i)$ since we add a forward move for each outgoing edge from v_j where $j \in [k]$. The number of Flip moves is exactly k since we add a flip move for each $i \in [k]$. Therefore, the degree of $\bar{v}$ in $\hat{G}$ is $d_G^-(v_0) + \left(\sum_{i=1}^{k} d_G^+(v_i) \right) + k \leq |E| + |E| + n$ since $k \leq n$ and $\sum_{v \in V} d_G^+(v) = |E| = \sum_{v \in V} d_G^-(v)$. Hence, the max degree of $\hat{G}$ is $2|E| + n$. Since $|\hat{V}| = n^{k+1}$ it follows that $|\mathcal{M}| = |\hat{E}| \leq n^{k+1} \cdot (2|E| + n) = n^{O(k)}$ as $|E| = O(n^2)$ ◻

Next we show that $\text{OPT}(I) = \text{OPT}(I')$, where $\text{OPT}(I)$ and $\text{OPT}(I')$ denote for the optimum solutions of I and I' respectively. We do this by the following two lemmas:

Lemma 2.9. $OPT(I) \leq OPT(I')$.

Proof. Let $\mathcal{S}'$ be a solution of I' of cost $\text{OPT}(I')$. Then $\mathcal{S}'$ is a shortest $s^{k+1} \rightsquigarrow t^{k+1}$ path in $\hat{G}$. Each edge in $\hat{G}$ corresponds to a move from $\mathcal{M}$. Let the moves corresponding to the path $\mathcal{S}'$ be $M_1, M_2, \ldots, M_r$. Consider a move $M \in \{M_1.M_2, \ldots, M_r\}$ and say $M = (\bar{v}, \bar{w})$ where $\bar{v} = (v_0, v_1, v_2, \ldots, v_k)$ and $\bar{w} = (w_0, w_1, w_2, \ldots, w_k)$. We now build a solution $\mathcal{S}$ for 2-SCSS-$(k, 1)$ as follows: there are three cases to consider depending on the type of the move M

- M **is a Backward Move**: Then we add the edge (w_0, v_0) to $\mathcal{S}$. Note that $\hat{C}(M) = \omega(w_0, v_0)$.
- M **is a Forward Move, say for token $\mathbf{f}_i$**: Then we add the edge (v_i, w_i) to $\mathcal{S}$. Note that $\hat{C}(M) = \omega(v_i, w_i)$.
- M **is a Flip Move, say between token $\mathbf{f}_i$ and $\mathbf{b}$**: Let P be the shortest $v_i \rightsquigarrow v_0$ path. Then we add the path P to $\mathcal{S}$. Note that $\hat{C}(M) = \sum_{e \in P} \omega(e)$.

Since $\mathcal{S}'$ is a solution of I' it follows that $\mathcal{S}$ is indeed a solution of I. Also the cost of $\mathcal{S}'$ is equal to $\sum_{i=1}^{r} \hat{C}(M_i)$. As we have seen above, we were able to construct a solution $\mathcal{S}$ for I from $\mathcal{S}'$. Note that for each edge e, its contribution to cost of $\mathcal{S}'$ is $\omega(e)$ which is greater equal to its contribution to cost of $\mathcal{S}$, since there might be some savings due to sharing of edges between the $t \rightsquigarrow s$ path and some $s \rightsquigarrow t$ path[2]. Thus the cost of $\mathcal{S}$ is at most the cost of $\mathcal{S}'$. Given any solution $\mathcal{S}'$ for I', we were able to construct a solution $\mathcal{S}$ for I of cost less equal that of $\mathcal{S}'$. Therefore, it follows that $\text{OPT}(I) \leq \text{OPT}(I')$. ◻

Lemma 2.10. $OPT(I) \geq OPT(I')$.

Proof. In order to prove this lemma we use Lemma 2.4 which states there exists an optimal solution, say $\mathcal{S}$, for I which is reverse-compatible. We now build a solution $\mathcal{S}'$ for I' which has the same cost as that of $\mathcal{S}$. This is sufficient to prove the lemma.

As observed in Remark 2.1, we can assume that the $s \rightsquigarrow t$ paths are pairwise edge-disjoint. Let the $t \rightsquigarrow s$ path in $\mathcal{S}$ be Q and the k paths from $s \rightsquigarrow t$ be $P_1, P_2, \ldots, P_k$. For $i \in [k]$, let $P_{i,1}, P_{i,2}, \ldots, P_{i,r_i}$ be the maximal sub-paths shared between P_i and Q as seen in order from P_i. The reverse-compatibility of $\mathcal{S}$ implies that if we traverse Q backwards then we again see the paths in the same order, namely $P_{i,1}, P_{i,2} \ldots, P_{i,r_i}$. Let us call these paths which are shared between Q and an $s \rightsquigarrow t$ path as **common-paths**.

We build $\mathcal{S}'$ by adding moves as follows: for each $i \in [k]$, add Forward moves (by selecting shortest paths) to transport each $\mathbf{f}_i$ from s to the starting point of $P_{i,1}$. Follow Q in backwards direction as it travels

---

[2]Consider a path $x \to y \to z$. Let $e_1 = (x, y)$ and $e_2 = (y, z)$. Suppose token $\mathbf{f}_1$ is at x and token $\mathbf{b}$ is at z and they want to exchange positions. A flip move would result in cost equal to $\omega(e_1) + \omega(e_2)$. However, we can have a move sequence of higher cost which results in same final positions for $\mathbf{f}_1$ and $\mathbf{b}$ as follows: first $\mathbf{f}_1$ makes a forward move and reaches y with cost $\omega(e_1)$. Then there is a flip move of cost $\omega(e_2)$ which brings $\mathbf{b}$ to y and takes $\mathbf{f}_1$ to z. Finally $\mathbf{b}$ makes a backward move of cost $\omega(e_1)$ to reach x. The total cost of this move sequence is $2\omega(e_1) + \omega(e_2)$, which is greater than the original cost of $\omega(e_1) + \omega(e_2)$
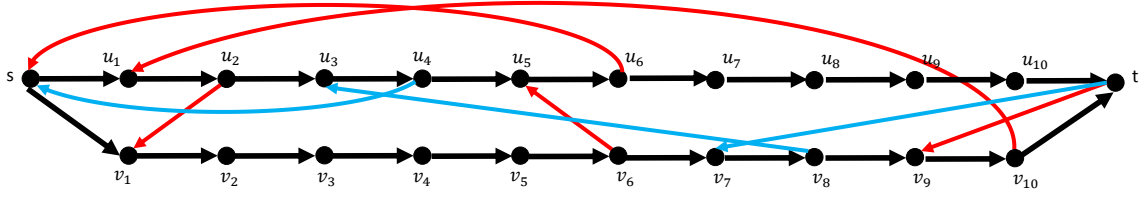
Figure 3: Each black edge has weight 1, each red edge and each blue edge has weight 0.

from s to t. We move $\mathbf{b}$ backwards along Q until it reaches end-point y of some common-path say $x \rightsquigarrow y$. Since we only require one $t \rightsquigarrow s$ path it follows that there exists unique $j \in [k]$ such that $P_{j,1} = x \rightsquigarrow y$. Then we add a Flip move between $\mathbf{f}_j$ (which is located at x) and $\mathbf{b}$ (which is located at y). Continue to follow Q backwards and move $\mathbf{b}$ along it by Backward moves until either $\mathbf{b}$ reaches t or $\mathbf{b}$ reaches end-point of another common path. If $\mathbf{b}$ reaches end-point of another common path then we again do a Flip move as above. Otherwise if $\mathbf{b}$ reaches t, then each forward token $\mathbf{f}_i$ is at the end-point of P_{i,r_i}. Add Forward moves (by selecting shortest paths) to transport the forward tokens from their current locations to t. Let the final set of moves be the solution $\mathcal{S}'$.

$\mathcal{S}$ is a valid solution for I implies that we have k paths from $s \rightsquigarrow t$ and one path from $t \rightsquigarrow s$. So, the moves we add indeed take all the $(k+1)$ tokens from the start state s^{k+1} to the end state t^{k+1}, i.e., $\mathcal{S}'$ is a valid solution for I'. We now show that the cost $\mathcal{S}'$ is exactly equal to that of $\mathcal{S}$. Let e be any edge in $\mathcal{S}$. If e is not part of a common-path, then in $\mathcal{S}'$ we only pay for it once in either a Backward move or a Forward move. On the other hand, if e is part of a common path then in $\mathcal{S}'$ again we also pay for it only once in a Flip move. Therefore, cost of $\mathcal{S}'$ is equal to cost of $\mathcal{S}$ which implies that $\text{OPT}(I) \geq \text{OPT}(I')$.
$\square$

Theorem 2.11. *There exists an algorithm that solves the* 2-SCSS-$(k,1)$ *in time $n^{O(k)}$, where n is the number of vertices of the input graph.*

Proof. Let $I = \langle G, s, t, \omega \rangle$ be an instance of the 2-SCSS-$(k,1)$. As described in Section 2.3, we build an instance $I' = \langle G, s, t, \mathcal{T} = (\mathbf{b}, \mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_k), \mathcal{M}, \hat{C} \rangle$ of SOLVING-TOKEN-GAME. Lemmas 2.9 and 2.10 imply that I can be solved by instead solving the instance I'. By Lemma 2.8, the number of moves in I' is $|\mathcal{M}| = n^{O(k)}$. By Lemma 2.7 we can solve I' in time $O(|\mathcal{M}| + n^{|\mathcal{T}|} \log(n^{|\mathcal{T}|})) = O(n^{O(k)} + n^{k+1} \log(n^{k+1})) = n^{O(k)}$ $\square$

2.4 Structural Lemma fails for 2-SCSS-(k_1, k_2) when $\min\{k_1, k_2\} \geq 2$

Recall that 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 extension of Definition 2.2 to reverse-compatibility of a set of forward paths and a set of backward paths as follows.

Definition 2.12. **(general-reverse-compatible)** *Let $\mathbf{F} = \{F_1, F_2, \ldots, F_{k_1}\}$ be a set of $s \rightsquigarrow t$ paths and $\mathbf{B} = \{B_1, B_2, \ldots, B_{k_2}\}$ be a set of $t \rightsquigarrow s$ paths. We say $(\mathbf{F}, \mathbf{B})$ is general-reverse-compatible, if for all $1 \leq i \leq k_2$, $(\mathbf{F}, B_i)$ is reverse-compatible.*

The following theorem shows that Lemma 2.4 does not hold for the 2-SCSS-(k_1, k_2) problem when $\min\{k_1, k_2\} \geq 2$, i.e., Lemma 2.4 is in its most general form.

Theorem 2.13. *There exists an instance of* 2-SCSS-$(2,2)$ *in which no optimum solution is general-reverse-compatible.*

9

Proof. Figure 3 illustrates an example of the 2-SSS(2,2) problem in which no optimal solution satisfies the reverse compatibility condition. Let the weight of the black edges be 1, and weight of all the other edges be 0. Since we have edges of weight 0, we will henceforth only consider the paths which do not have vertices repeating.

Let P_1 be the path $s \to u_1 \to u_2 \to \ldots \to u_9 \to u_{10} \to t$ and P_2 be the path $s \to v_1 \to v_2 \to \ldots \to v_9 \to v_{10} \to t$. Note that P_1 and P_2 are edge-disjoint and have weight 11 each. We now give a solution of total weight 22: take P_1 and P_2 as the two $s \rightsquigarrow t$ paths. For the two $t \rightsquigarrow s$ paths take $P_3 := t \to v_7 \to v_8 \to u_3 \to u_4 \to s$ and $P_4 := t \to v_9 \to v_{10} \to u_1 \to u_2 \to v_1 \to v_2 \to v_3 \to v_4 \to v_5 \to v_6 \to u_5 \to u_6 \to s$. Since every black edge is used exactly once in the outgoing path and incoming path, it is easy to verify that the total weight of this solution is 22. Moreover, this solution is not general-reverse-compatible since the paths P_1 and P_3 do not satisfy the path-reverse-compatibility condition (recall Definition 2.2).

Therefore, to prove the theorem, it is now enough to show that all other solutions have a weight at least 23. A simple observation is that any solution has weight at least 22 since the shortest path from s to t has weight 11. Moreover, there are exactly two such $s \rightsquigarrow t$ paths of weight 11, viz. P_1 and P_2. Hence suppose to the contrary that there is a solution, say **S**, of weight exactly 22. We now show that **S** must exactly be the solution described in above paragraph. We first show the following lemma:

Lemma 2.14. *Any $t \rightsquigarrow s$ path uses at least one black edge from each of P_1 and P_2.*

Proof. Note that there are only two edges outgoing from t: a blue edge and a red edge. Suppose the first edge on $t \rightsquigarrow s$ path is the red edge $t \to v_9$. Then we must reach v_{10} since the only outgoing edge from v_9 is $v_9 \to v_{10}$. From v_{10}, we can either go back to t (and start the argument again) or the other option is to go to u_1 which forces the use of edge $u_1 \to u_2$. So we have used $v_9 \to v_{10}$ from P_2 and $u_1 \to u_2$ from P_1.

Suppose the first edge on $t \rightsquigarrow s$ path is the blue edge $t \to v_7$. This forces the use of the edge $v_7 \to v_8$ from P_2 since it is the only outgoing edge from v_7. From v_8, we can either reach v_9 (and the same argument applies as in previous case) or u_3. Reaching u_3 forces the use of the edge $u_3 \to u_4$ from P_1 since it is the only outgoing edge from u_3. $\square$

Hence, in order to obtain a solution of weight exactly 22 we cannot take either P_1 twice or P_2 twice for the choice of the two $s \rightsquigarrow t$ paths: since this itself gives a weight of 22, and the above claim implies a weight of at least 1 from the "other" path. This shows the correctness of the following lemma:

Lemma 2.15. *The two $s \rightsquigarrow t$ paths in S are exactly P_1 and P_2. Hence, to maintain a weight of exactly 22 it follows that we cannot use any black edge twice in the $t \rightsquigarrow s$ paths in S.*

Observe that we still need to choose two $t \rightsquigarrow s$ paths, say Q_1 and Q_2, in **S**. The following lemma shows that **S** needs to use both the red edge and blue edge outgoing from t:

Lemma 2.16. *Without loss of generality, the first edges of Q_1 and Q_2 are $t \to v_7$ and $t \to v_9$*

Proof. Suppose not. Since the only two outgoing edges from t are the blue edge $t \to v_7$ and the red edge $t \to v_9$, it follows that the first edge of both Q_1 and Q_2 is the same (and is either $t \to v_7$ or $t \to v_9$). Suppose the first edge of both Q_1 and Q_2 is $t \to v_7$ (the argument for the first edge being $t \to v_9$ is similar). Since $v_7 \to v_8$ is the only outgoing edge from v_7, this implies that we must choose this edge in both Q_1 and Q_2. Since the two $s \rightsquigarrow t$ paths in **S** are P_1 and P_2, this shows that the weight of **S** is at least 23. $\square$

Let us now consider the path Q_1: it starts with the edge $t \to v_7$. Since the only outgoing edges from v_7, u_3 are $v_7 \to v_8, u_3 \to u_4$ respectively it follows that Q_1 contains the sub-path $Q_1' := t \to v_7 \to v_8 \to u_3 \to u_4$. Similarly for Q_2, the first edge being $t \to v_9$ implies that it contains the sub-path $Q_2' := t \to v_9 \to v_{10} \to u_1 \to u_2$. After this, Q_2 cannot contain the edge $u_2 \to u_3$ (since this would force it to also use the edge $u_3 \to u_4$, which was already used by Q_1). Hence after Q_2', the path Q_2 must follow the sub-path $u_2 \to v_1 \to$

$v_2 \to v_3 \to v_4 \to v_5 \to v_6$. After reaching v_6, the path Q_2 has two choices: either use the edge $v_6 \to v_7$ or $v_6 \to v_5$. But it cannot use the edge $v_6 \to v_7$ since that would force it to use the edge $v_7 \to v_8$, which was already used by Q_1. Therefore, from v_6 the path Q_2 reaches u_5 and is then forced to reach u_6. At this point Q_2 has two choices: either continue from u_6 to t (in which case we again apply the whole argument starting from Lemma 2.16), or use the edge $u_6 \to s$ of weight 0. Therefore we have that Q_2 is exactly the path $P_4 := t \to v_9 \to v_{10} \to u_1 \to u_2 \to v_1 \to v_2 \to v_3 \to v_4 \to v_5 \to v_6 \to u_5 \to u_6 \to s$. It remains to show that the path Q_1 is exactly P_3. We know that Q_1 contains the sub-path $Q_1' := t \to v_7 \to v_8 \to u_3 \to u_4$. From u_4, there are two choices: either use the edge $u_4 \to s$ of weight 0, or use the edge $u_4 \to u_5$. However, in the second choice, the next edge on Q_2 must be $u_5 \to u_6$. But this edge was already used by Q_2 which contradicts Lemma 2.16. This shows that Q_1 is exactly the path $P_3 = t \to v_7 \to v_8 \to u_3 \to u_4 \to s$, which completes the proof of the theorem. □

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

In this section we prove Theorem 1.2. We reduce from the GRID TILING problem (see Section 1.1 for definition). Chen et al. [3] showed that for any computable function f, the existence of an $f(k) \cdot n^{o(k)}$ algorithm for k-CLIQUE implies ETH fails. Marx [11] 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 4. 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 3.1. *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.*

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 4. 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 $\ell \in [n]$
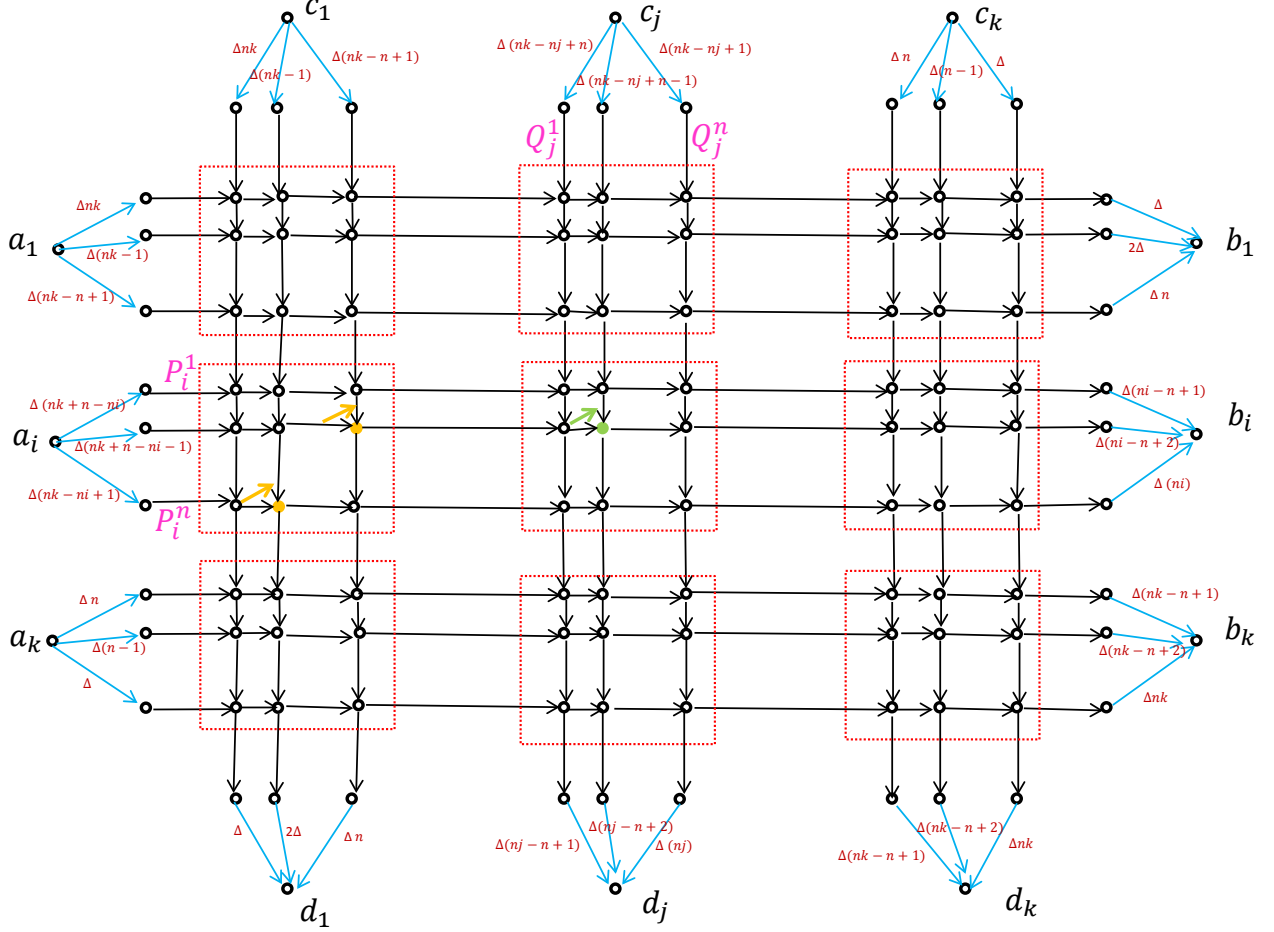
11

Figure 4: The instance of 2-SCSS-$(2k-1,1)$ created from an instance of Grid Tiling*.

we assign a weight of $\Delta(nk-ni+n+1-\ell), \Delta(ni-n+\ell)$ to the first, last edges of P_i^ℓ (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^ℓ (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 Δ 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 red gadgets and $(n-1)$ black edges inside each gadget. Since the number of gadgets that each canonical path visits is k and the weight of each black edge is 4, it follows that the total weight of any canonical path is

$$\alpha = \Delta(nk+1) + 4(k+1) + 4k(n-1) \tag{1}$$

Intuitively the k^2 red 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 an asymmetric gadget. We perform the following modifications on the edges inside the gadget: (see Figure 4)

- **Symmetric Gadgets**: For each $i \in [k], x \in [n]$ 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^x. Then we add a shortcut as shown in Figure 5.

12
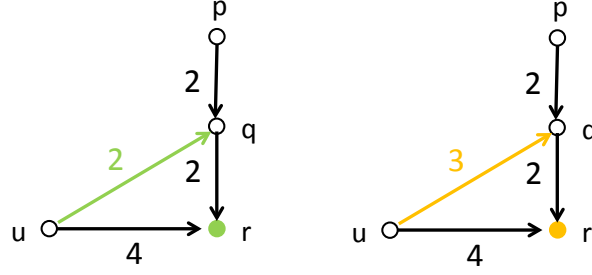
Figure 5: Let u, r be two consecutive vertices on the canonical path say P_i^ℓ. Let r be on the canonical path $Q_j^{\ell'}$ and let p be the vertex preceding it on this path. If r is a green (respectively orange) vertex then we subdivide the edge (p, r) by introducing a new vertex q and adding two edges (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.

The idea is that we will enforce that the $a_i \rightsquigarrow b_i$ path is used as part of the $t \rightsquigarrow s$ path and the $c_i \rightsquigarrow d_i$ path is used as part of one of the $(2k - 1)$ $s \rightsquigarrow t$ paths. Hence, 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 (which has weight 2) and a vertical downward edge (which is free since 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.

- **Asymmetric Gadgets**: For each $i \neq j \in [k]$, if $(x, y) \in S_{i,j}$ then we color orange the vertex in the gadget $G_{i,j}$ which is the unique intersection of the canonical paths P_i^x and Q_j^y. Then we add a shortcut as shown in Figure 5. Similar to above, 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 free since already being used by the $c_j \rightsquigarrow d_j$ canonical path) to reach the orange vertex, instead of paying a weight of 4 to use the horizontal edge reaching the orange vertex.

From Figure 4, it is easy to see that each canonical path has weight equal to α.

3.1 Vertices and Edges not shown in Figure 4

The following vertices and edges are not shown in Figure 4 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.
- For each $2 \leq i \leq k$, add edges $s \rightarrow e_i$ and $f_i \rightarrow t$ of weight 0.
- For each $2 \leq i \leq k$, add a path $b_i \rightarrow e_i \rightarrow f_i \rightarrow a_{i-1}$. Set the weights of (b_i, e_i) and (f_i, a_{i-1}) to be 0.
- 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-$(2k - 1, 1)$. We will see later that $W = 53n^9$ suffices for our reduction.

Remark 3.2. We need a small technical modification: 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-$(2k-1,1)$. Let

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

3.2 GRID TILING* has a solution $\Rightarrow$ 2-SCSS-$(2k-1,1)$ has a solution of weight $\leq \beta$

First we show the easy direction.

Theorem 3.3. GRID TILING* *has a solution implies* 2-SCSS-$(2k-1,1)$ *has a solution of weight at most* β.

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, there exist k numbers $\delta_1, \delta_2, \ldots, \delta_k$ such that $s_{i,j} = (\delta_i, \delta_j)$ for each $1 \leq i, j \leq k$. 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 $b_i \rightarrow e_i \rightarrow f_i \rightarrow a_{i-1}$. 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 $\alpha \cdot k + W(k-1)$ since we have used k canonical paths and $(k-1)$ connector edges.
- Finally use the edge (b_1, s) of weight 0.

Therefore, with a total weight of $\alpha \cdot k + W(k-1)$ we have obtained 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 (different) $s \rightsquigarrow t$ paths. In particular, we get $(k-1)$ $s \rightsquigarrow t$ paths given by $s \rightarrow e_i \rightarrow f_i \rightarrow t$ for each $2 \leq i \leq k$. Note that the total weight 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 (as it was used in the $t \rightsquigarrow s$ path) and both the edges (s, e_i) and (f_i, t) have weight 0.

Note that we still need k more $s \rightsquigarrow t$ paths. For each $j \in [k]$, we add the canonical $c_j \rightsquigarrow d_j$ path $Q_j^{\delta_j}$. 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 a $s \rightsquigarrow t$ path whose weight is exactly equal to α. However, now the canonical paths will encounter exactly one vertex in each gadget: either green or orange depending on whether the gadget is symmetric or asymmetric respectively. As shown in Figure 5, 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 $k(k-1)$, we save a total weight of $2k + k(k-1) = (k^2 + k)$.

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

3.3 2-SCSS-$(2k-1,1)$ has a solution of weight $\leq \beta \Rightarrow$ GRID TILING* has a solution

We now prove the other direction which is more involved. Fix a solution $\mathcal{X}$ of 2-SCSS-$(2k-1,1)$ which has cost $\leq \beta$. First we show some preliminary lemmas:

Definition 3.4. *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 3.5. *The $t \rightsquigarrow s$ path in $\mathcal{X}$*

- *Must use all the $k-1$ connector edges*

14

- *Contains an $a_i \rightsquigarrow b_i$ path (for each $i \in [k]$) which does not include any connector edge*

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 $b_i \rightarrow e_i \rightarrow f_i \rightarrow a_{i-1}$. That is, we must use all the $(k - 1)$ connector edges which are given by (e_i, f_i) for each $2 \leq i \leq k$.

Now we show the second part of the lemma. First observe that the above argument also implies that $\mathcal{X}$ contains an $a_i \rightsquigarrow b_i$ path for each $2 \leq i \leq k$. Since the only incoming edge into s is (b_1, s), we must also have an $a_1 \rightsquigarrow b_1$ path in $\mathcal{X}$. Therefore, the $t \rightsquigarrow s$ path contains an $a_i \rightsquigarrow b_i$ path for each $i \in [k]$. Clearly, the $a_k \rightsquigarrow b_k$ path cannot use any connector edge due to orientation of the edges. For $1 \leq i \leq k - 1$ consider a $a_i \rightsquigarrow b_i$ path P in $\mathcal{X}$. If it uses any connector edge, say (e_j, f_j), then it follows from the orientation of the edges that $j > i$. Hence this path P reaches the vertex b_j which is at level j. Recall that the only way to climb a level above in the graph (that is, one with a smaller index) is through connector edges. Therefore, the next time that the path P reaches level i (which it has to in order to reach vertex b_i) it must do so at vertex a_i. Hence, the $a_i \rightsquigarrow b_i$ sub-path of P which starts at the last occurrence of a_i on P satisfies the condition of not using any connector edge. $\square$

Lemma 3.6. *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 $\mathcal{X}$ is at least $\Delta(nk + 1)$.*

Proof. From Lemma 3.5, for each $i \in [k]$ we know that $\mathcal{X}$ 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 $\Delta(nk - ni + n + 1 - \ell) + \Delta(ni - n + \ell') = \Delta(nk + 1) + (\ell' - \ell) \geq \Delta(nk + 1)$. $\square$

Lemma 3.7. *At least k of the $s \rightsquigarrow t$ paths in $\mathcal{X}$ cannot use any connector edge.*

Proof. If less than k of the $s \rightsquigarrow t$ paths in $\mathcal{X}$ do not use connector edges, then this implies that at least k of the $s \rightsquigarrow t$ paths in $\mathcal{X}$ use a connector edge, since we require $(2k - 1)$ paths from $s \rightsquigarrow t$. Since there are exactly $(k - 1)$ connector edges, some connector edge is used by two different $s \rightsquigarrow t$ paths. As we have seen in Lemma 3.5, the $t \rightsquigarrow s$ path in $\mathcal{X}$ must use all the $(k - 1)$ connector edges. Hence, the weight of $\mathcal{X}$ is $\geq W(k - 1) + W = Wk$. We show below that this is greater than β, which gives a contradiction.

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

$\square$

15

We call the $s \rightsquigarrow t$ paths described in Lemma 3.7 as **expensive** paths. Since these paths do not use a connector edge, it follows that the only outgoing edges from s to be considered are to $\{c_1, c_2, \ldots, c_k\}$ and the only incoming edges into t to be considered 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 3.8. *For each $i \in [k]$, let λ_i denote the number of $c_i \rightsquigarrow d_i$ expensive paths and μ_i denote the number of $c_i \rightsquigarrow \{d_{i+1}, d_{i+2}, \ldots, d_k\}$ expensive paths respectively in $\mathcal{X}$.*

From Lemma 3.7, it follows that

$$\sum_{i=1}^{k} (\lambda_i + \mu_i) \geq k \tag{3}$$

Lemma 3.9. *Let $c_j \rightsquigarrow d_\ell$ be an expensive path in $\mathcal{X}$. Then the sum of weights of the blue edges in this path is exactly $\Delta(nk+1)$ if the path is canonical, and at least $\Delta(nk+1) + \Delta$ otherwise.*

Proof. Since expensive paths do not use connector edges, we have $\ell \geq j$. We consider two cases: either $\ell = j$ or $\ell > j$.

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 $\Delta(nk - nj + n + 1 - r) + \Delta(nj - n + r') = \Delta(nk+1) + \Delta(r' - r)) \geq \Delta(nk+1)$. Note that if the path is canonical then $r' = r$ and the weight is exactly $\Delta(nk+1)$. Otherwise, if the path is not canonical then $r' > r$ and then the weight is $\Delta(nk+1) + \Delta(r' - r)) \geq \Delta(nk+1) + \Delta$

In the last case suppose $\ell > j$: so clearly the path is not canonical. The minimum weights of any blue edges incident on c_j, d_ℓ are $\Delta(nk - nj + 1), \Delta(n\ell - n + 1)$ respectively. Hence, the sum of weights of these edges is $\Delta(nk - nj + 1) + \Delta(n\ell - n + 1) = \Delta(nk+1) + \Delta + \Delta(n(\ell - j - 1)) \geq \Delta(nk+1) + \Delta$. □

Lemma 3.10. *The weight of blue edges in $\mathcal{X}$ is at least $2k \cdot \Delta(nk+1)$.*

Proof. From Lemma 3.6, 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 $\mathcal{X}$ is at least $\Delta(nk+1)$ for each $i \in [k]$. From Lemma 3.9, we know that the sum of weights of the blue edges in any expensive path is at least $\Delta(nk+1)$. Moreover, Lemma 3.7 implies that there are at least k expensive paths. Since all these edges are clearly disjoint, it follows that the total weight of blue edges is at least $2k \cdot \Delta(nk+1)$. □

Lemma 3.11. *The weight of black edges in $\mathcal{X}$ is at least $2k\left(4(k+1) + 4k(n-1)\right)$, without considering the savings via orange and green edges (see Figure 5).*

Proof. From Lemma 3.5, we know that for each $i \in [k]$ there is an $a_i \rightsquigarrow b_i$ path in $\mathcal{X}$ which does not include any connector edge. Hence, the edges of this $a_i \rightsquigarrow b_i$ paths 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 weight $4(k+1) + 4k(n-1)$. 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 $k\left(4(k+1) + 4k(n-1)\right)$.

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_ℓ. These vertically downward black edges have total weight $4(k+1) + 4k(n-1)$. 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. By Lemma 3.7, there are at least k expensive paths and hence the total weight of the vertically downward black edges is at least $k\Big(4(k+1)+4k(n-1)\Big)$.

Combining the two observations above, it follows that the total weight of black edges (horizontally right and vertically downward) in $\mathcal{X}$ is at least $2k\Big(4(k+1)+4k(n-1)\Big)$, without considering the savings via orange and green edges (see Figure 5). $\square$

Lemma 3.12. *Every expensive path in $\mathcal{X}$ is canonical, i.e., $\mu_j = 0$ for all $j \in [k]$.*

Proof. Suppose an expensive path is not canonical. Lemma 3.9 implies that the contribution of the blue edges of this expensive path is $\geq \Delta(nk+1)+\Delta$. By an argument exactly similar to that of Lemma 3.10, it follows that the contribution of the blue edges to weight of $\mathcal{X}$ is at least $2k \cdot \Delta(nk+1)+\Delta$.

Refer to Figure 5. Note that we can use each shortcut at most $\binom{2k}{2}$ times, once for each pair of paths that will meet at the orange or green vertex (note that there are total $2k$ 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 at most

$$\binom{2k}{2}\Big(2k \cdot n + (n^2 - n)(k^2 - k)\Big) = k(2k-1)\Big(2k \cdot n + (n^2 - n)(k^2 - k)\Big)$$
$$\leq 2n^2 \cdot (2n^2 + n^4) \qquad \text{[Since } k \leq n\text{]}$$
$$\leq 6n^6$$

We now claim that the weight of our solution already exceeds β, even if we allow this maximum possible saving. Recall that we have weight of $W(k-1)$ from the connector edges. Hence, the weight of $\mathcal{X}$ is at least

$$\text{weight of } \mathcal{X} \geq W(k-1) + \Big(2k \cdot \Delta(nk+1)+\Delta\Big) + 2k\Big(4(k+1)+4k(n-1)\Big) - 6n^6$$
$$= W(k-1) + 2k \cdot \Delta(nk+1) + 2k\Big(4(k+1)+4k(n-1)\Big) + \Big(\Delta - 6n^6\Big)$$
$$= W(k-1) + 2k \cdot \Delta(nk+1) + 2k\Big(4(k+1)+4k(n-1)\Big) + n^6 \quad \text{[Since } \Delta = 7n^6\text{]}$$
$$> W(k-1) + 2k \cdot \Delta(nk+1) + 2k\Big(4(k+1)+4k(n-1)\Big)$$
$$> W(k-1) + 2k \cdot \Delta(nk+1) + 2k(4(k+1)+4k(n-1)) - (k^2 - k)$$
$$= \beta \quad \text{[From Equation 2]}$$

Contradiction. $\square$

Lemma 3.13. $\sum_{i=1}^{k} \lambda_i = k$

Proof. From Equation 3 and Lemma 3.12, it follows that $\sum_{i=1}^{k} \lambda_i \geq k$. Suppose $\sum_{i=1}^{k} \lambda_i > k$, i.e., there are at least $k+1$ expensive paths. We follow a line of argument similar to that in proof of Lemma 3.10. Note that the blue edges incident used in an expensive path are not used in the $t \rightsquigarrow s$ path in $\mathcal{X}$, and hence it follows that the total cost of the blue edges from expensive paths is at least $(k+1) \cdot \Delta(nk+1)$. Hence the total weight of the blue edges is at least $k \cdot \Delta(nk+1) + (k+1) \cdot \Delta(nk+1) = 2k \cdot \Delta(nk+1) + \Delta(nk+1) \geq 2k \cdot \Delta(nk+1) + \Delta$. Now an argument exactly similar to that of Lemma 3.12 shows that weight of $\mathcal{X}$ exceeds β, which is a contradiction. $\square$

Note the shortcuts described in Figure 5 again bring the $a_i \rightsquigarrow b_i$ path back to the same horizontal canonical path.

Definition 3.14. *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 5. An almost-canonical path must however end on the same horizontal level on which it began.*

Lemma 3.15. $\mathcal{X}$ *contains exactly one $a_i \rightsquigarrow b_i$ path for each $i \in [k]$. Moreover, this path is almost-canonical.*

Proof. Fix some $i \in [k]$. From Lemma 3.5, we know that $\mathcal{X}$ contains an $a_i \rightsquigarrow b_i$ path which 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 $\mathcal{X}$ is not almost-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 $\mathcal{X}$ that starts with the blue edge from P_i^ℓ and ends with a blue edge from $P_i^{\ell'}$ for some $\ell' > \ell$. Hence, the contribution of these blue edges is $\Delta(nk - ni + n + 1 - \ell) + \Delta(ni - n + \ell') = \Delta(nk + 1) + \Delta(\ell' - \ell) \geq \Delta(nk + 1) + \Delta$. Now, a similar argument as in Lemma 3.12 can be applied to show that the weight of $\mathcal{X}$ is greater than β. Contradiction.

The above paragraph shows that each $a_i \rightsquigarrow b_i$ path in $\mathcal{X}$ is almost-canonical. Suppose there are at least two $a_i \rightsquigarrow b_i$ paths in $\mathcal{X}$. Then the blue edges incident on a_i, b_i must be different (otherwise we get the same almost-canonical path). Therefore, the sum of blue edges incident on a_i and b_i is $\geq 2\Delta(nk + 1)$. A similar argument to Lemma 3.12 can be applied to show that the weight of $\mathcal{X}$ is greater than β. Contradiction. □

Theorem 3.16. *OPT for 2-SCSS-$(k, 1)$ is at most β implies the* GRID TILING* *instance has a solution.*

Proof. By Lemma 3.13, 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 $i \in [k]$. By Lemma 3.12 and Lemma 3.15, we know that $\mathcal{X}$ contains

- (Property 1): Exactly one $a_i \rightsquigarrow b_i$ (almost-canonical) path for every $i \in [k]$.
- (Property 2): Exactly k canonical expensive paths.

In addition, $\mathcal{X}$ contains $(k - 1)$ connector edges. For the moment let us forget the shortcuts we added in Figure 5. The weight of $\mathcal{X}$, without considering the shortcuts from Figure 5, 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 3.15, we know that for each $i \in [k]$ there is exactly one $a_i \rightsquigarrow b_i$ path in $\mathcal{X}$. Moreover, this path is almost-canonical. Recall that only the horizontal edges can save some weight (see Figure 5). Therefore, we can use at most k green edges (one f or each symmetric gadget). Each canonical expensive path can use at most $(k - 1)$ orange edges; once for each of the $(k - 1)$ asymmetric gadget that it encounters along the way. Suppose we use δ green edges. Then Property 1 and Property 2 above imply that $\delta \leq k$. Then the total saving is at most $(k - 1)\big(\sum_{i=1}^k \lambda_i\big) + 2\delta = k(k - 1) + 2\delta$. 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 $G_{i,i}$ which contradicts $\delta = k$. 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 $1 \leq i \neq j \leq k$ we have $(\alpha_i, \alpha_j) \in S_{i,j}$.

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

3.4 Proof of Theorem 1.2

Finally, we are now ready to prove Theorem 1.2 which is restated below:

Theorem 1.2 . *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.*

Proof. Theorem 3.3 implies the W[1]-hardness by giving a reduction which transforms the problem of $k \times k$ GRID TILING* into an instance of 2-SCSS-$(2k-1,1)$ where we want to find $2k-1$ paths from $s \rightsquigarrow t$ and one path from $t \rightsquigarrow s$.

Chen et al. [3] showed for any computable function f, the existence of an $f(k) \cdot n^{o(k)}$ time algorithm for CLIQUE implies ETH fails. Composing the reduction of [11] from CLIQUE to GRID TILING*, along with our reduction from GRID TILING* to 2-SCSS-$(2k-1,1)$, we obtain under ETH there is no $f(k) \cdot n^{o(k)}$ algorithm for 2-SCSS-$(k,1)$ for any computable function f since the parameter blowup is linear. This shows that the $n^{O(k)}$ algorithm for 2-SCSS-$(k,1)$ given in Section 2 is asymptotically optimal. □

4 Conclusions

In this paper, for any $k \geq 1$ we studied the 2-SCSS-$(k,1)$ problem and presented an algorithm which finds an optimum solution in $n^{O(k)}$ time. Moreover, we showed our algorithms is asymptotically optimal: under the ETH, the 2-SCSS-$(k,1)$ problem does not admit an $f(k) \cdot n^{o(k)}$ time algorithm for any computable function f. This algorithm crucially used the fact that there always exists an optimal solution for 2-SCSS-$(k,1)$ that has the reverse-compatibility property. However, we showed in Section 2.4 that the 2-SCSS-(k_1, k_2) problem need not always have an optimal solution which satisfies the general-reverse-compatibility property when $\min\{k_1, k_2\} \geq 2$. Therefore, 2-SCSS-$(k,1)$ is the most general problem that one can attempt to solve with our techniques. 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.

References

[1] Chakrabarty, D., Chekuri, C., Khanna, S., Korula, N.: Approximability of Capacitated Network Design. In: IPCO. pp. 78–91 (2011)

[2] Charikar, M., Chekuri, C., Cheung, T.Y., Dai, Z., Goel, A., Guha, S., Li, M.: Approximation Algorithms for Directed Steiner Problems. J. Algorithms 33(1) (1999)

[3] 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)

[4] Chitnis, R.H., Esfandiari, H., Hajiaghayi, M., Khandekar, R., Kortsarz, G., Seddighin, S.: A Tight Algorithm for Strongly Connected Steiner Subgraph on Two Terminals with Demands (Extended Abstract). In: IPEC. pp. 159–171 (2014)

[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] 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)

[7] 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)

[8] 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

[9] Halperin, E., Krauthgamer, R.: Polylogarithmic inapproximability. In: STOC. pp. 585–594 (2003)

[10] Impagliazzo, R., Paturi, R.: On the Complexity of k-SAT. J. Comput. Syst. Sci. 62(2), 367–375 (2001)

[11] Marx, D.: On the Optimality of Planar and Geometric Approximation Schemes. In: FOCS. pp. 338–348 (2007)

[12] Marx, D.: A Tight Lower Bound for Planar Multiway Cut with Fixed Number of Terminals. In: ICALP (1). pp. 677–688 (2012)

[13] 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)

[14] Marx, D., Pilipczuk, M.: Optimal parameterized algorithms for planar facility location problems using voronoi diagrams. CoRR abs/1504.05476 (2015), `http://arxiv.org/abs/1504.05476`

[15] 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

[16] Ramanathan, S.: Multicast tree generation in networks with asymmetric links. IEEE/ACM Transactions on Networking (TON) 4(4), 558–568 (1996)

[17] Teixeira, R., Marzullo, K., Savage, S., Voelker, G.M.: Characterizing and measuring path diversity of internet topologies. In: SIGMETRICS. pp. 304–305 (2003)

[18] 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)