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

Rajesh Hemant Chitnis† Hossein Esfandiari‡ MohammadTaghi Hajiaghayi†
Rohit Khandekar§ Guy Kortsarz¶ Saeed Seddighin†

May 20, 2015

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 a $t_i\to t_j$ 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 $\phi(e)=\max\Big\{\big|\{i: i\in[k_1],\ e\in F_i\}\big|\ ,\ \big|\{j: j\in[k_2],\ e\in B_j\}\big|\Big\}$. 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 the 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 $\omega: E\to\mathbb{R}^+$ and a

*An extended abstract [4] appeared in IPEC '14

†The Weizmann Institute of Science, Rehovot, Israel. Supported by a postdoctoral fellowship from I-CORE ALGO. Work done in part when at the University of Maryland, College Park. Email: rajesh.chitnis@weizmann.ac.il

‡Department of Computer Science, University of Maryland at College Park, USA. Supported in part by NSF CAREER award 1053605, NSF grant CCF-1161626, ONR YIP award N000141110662, DARPA/AFOSR grant FA9550-12-1-0423. Email: {hossein, hajiagha, saeedrez}@cs.umd.edu

§KCG Holdings Inc., USA. Email: rkhandekar@gmail.com

¶Department of Computer Science, Rutgers University-Camden, USA. Supported by NSF grant 1218620. Email: guyk@camden.rutgers.edu

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 [15] 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 [15] 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 a $t_i \rightarrow t_j$ path for each $1 \leq i \neq j \leq p$. 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.

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 $\omega : E \rightarrow \mathbb{R}_{\geq 0}$, 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. Teixeira et al. [16, 17] 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, 14] 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.*

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 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 [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. Moroever, 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 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]. 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-$(k,1)$. 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 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 $S_{i,i}=\{(j,j):1\leq j\leq n\}$ 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.

In Appendix A we show that the edge-weighted and the vertex-weighted variants of 2-SCSS-(k_1,k_2) are computationally equivalent. 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*

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

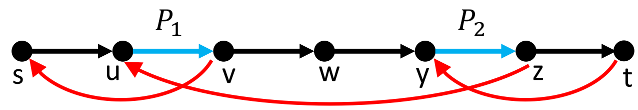

Figure 1: Let F be an $s \rightsquigarrow t$ path given by $s \to u \to v \to w \to y \to z \to t$ and B be an $t \rightsquigarrow s$ path given by $t \to y \to z \to u \to v \to s$. The two paths P_1 and P_2 shown in blue are the maximal common sub-paths between F and B. From Definition 2.1, it follows that F and B are *path-reverse-compatible* since B first sees P_2 and then P_1.

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

For simplicity, we replace each edge e of the input graph G with k copies $e_1, e_2, \dots, e_k$, 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 2.1. (path-reverse-compatible) *Let F be a $s \rightsquigarrow t$ path and B be a $t \rightsquigarrow s$ path. Let $\{P_1, P_2, \dots, P_d\}$ 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 2.2. (reverse-compatible) *Let $\boldsymbol{F} = \{F_1, F_2, \dots, F_r\}$ be a set of $s \rightsquigarrow t$ paths and b be an $t \rightsquigarrow s$ path. We say $(\boldsymbol{F}, B)$ 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 2.3. (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 2.4. (rank) *Let $\boldsymbol{F} = \{F_1, F_2, \dots, F_k\}$ 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 $(\boldsymbol{F}, B)$ is given by*

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

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, i.e., there exists some $F_i \in \mathbf{F}$ such that (F_i, B) is not path-reverse-compatible. From Definition 2.1, 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).

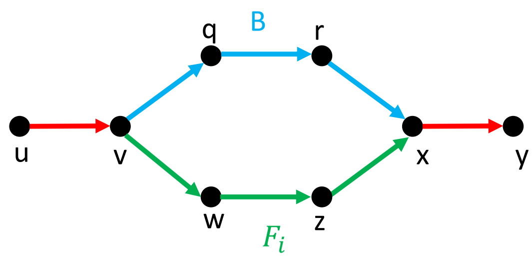

Figure 2: Let the $u \rightsquigarrow y$ sub-path of F_i be a $u \rightsquigarrow v \rightsquigarrow w \rightsquigarrow z \rightsquigarrow x \rightsquigarrow y$ and the $u \rightsquigarrow y$ sub-path of B be $u \rightsquigarrow v \rightsquigarrow q \rightsquigarrow r \rightsquigarrow x \rightsquigarrow y$. From Definition 2.1, it follows that F_i and B are not *path-reverse-compatible* since they both first see $u \rightsquigarrow v$ and then see $x \rightsquigarrow y$.

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. $\square$

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 $1 \le i \le d$ 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 $\mathcal{O}(n^{|\mathcal{T}|} \cdot |\mathcal{M}| \cdot \log(n^{|\mathcal{T}|}))$, where n is the number of the vertices of G.

Lemma 2.5. (algorithm for Token Game) *There exists an algorithm which solves the Token game in time* $\mathcal{O}(n^{|\mathcal{T}|} |\mathcal{M}| \log(n^{|\mathcal{T}|}))$.

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 $\mathcal{O}(n^{|\mathcal{T}|} |\mathcal{M}| \log(n^{|\mathcal{T}|}))$. $\square$

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 $\mathrm{Cor}(I) = \langle G',s',t',\mathcal{T},\mathcal{M},\hat{C}\rangle$ 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 k moves $\{(\mathcal{F}_i,u,v)\}$ to $\mathcal{M}$ for all $1 \le i \le k$. 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 $1 \le i \le k$. Cost of each move is equal to the distance of vertex v from vertex u in G.

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

Lemma 2.6. *For a given instance I of the 2-SCSS-$(k,1)$ we have $OPT(I) \ge OPT(Cor(I))$.*

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.

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

$$opt(I) \le opt(Cor(I))$$

□

Lemma 2.7. *For a given instance I of the 2-SCSS-$(k,1)$ we have $OPT(I) \le OPT(Cor(I))$.*

Proof. In order to prove this lemma we use Lemma 2.3 which states there exists an optimal solution for I which is reverse-compatible; Let it be $(F_1,F_2,\ldots,F_kB)$. We provide a solution for $Cor(I)$ with the total cost equal to $\mathcal{C}(F_1,F_2,\ldots,F_kB)$.

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:

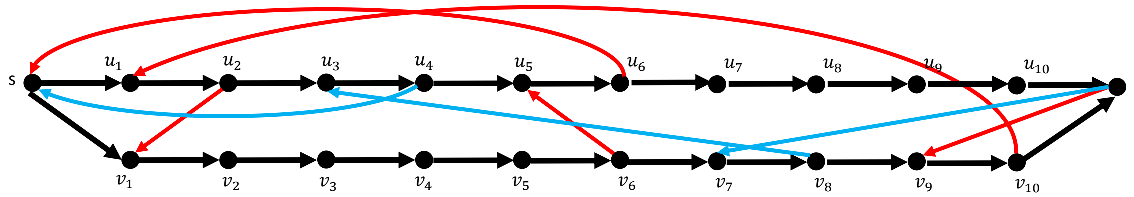

Figure 3: Each black edge has weight 1, each red edge and each blue edge has weight 0.

- 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 begining 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 $opt(I) \geq opt(Cor(I))$. $\square$

Theorem 2.8. *There exists an algorithm that solves the* 2*-SCSS-*$(k,1)$ *in time* $\mathcal{O}(n^{\mathcal{O}(k)})$.

Proof. Let I be an instance of the 2-SCSS-$(k,1)$. According to Lemmas 2.6 and 2.7, we have $opt(I) = opt(Cor(I))$.

Since the number of moves in $\mathcal{M}$ is $\mathcal{O}(n^2 d)$, by Lemma 2.5 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. $\square$

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 2.1 to reverse-compatibility of a set of forward paths and a set of backward paths as follows.

Definition 2.9. (general-reverse-compatible) *Let* $\boldsymbol{F} = \{F_1, F_2, \ldots, F_{k_1}\}$ *be a set of* $s \rightsquigarrow t$ *paths and* $\boldsymbol{B} = \{B_1, B_2, \ldots, B_{k_2}\}$ *be a set of* $t \rightsquigarrow s$ *paths. We say* $(\boldsymbol{F}, \boldsymbol{B})$ *is general-reverse-compatible, if for all* $1 \leq i \leq k_2$, $(\boldsymbol{F}, B_i)$ *is reverse-compatible.*

The following theorem shows that Lemma 2.3 does not hold for the 2-SCSS-(k_1,k_2) problem when $\min\{k_1,k_2\} \geq 2$, i.e., Lemma 2.3 is in its most general form.

Theorem 2.10. *There exists an instance of* 2*-SCSS-*$(2,2)$ *in which no optimum solution is general-reverse-compatible.*

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 2.1).

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 $\mathbf{S}$, of weight exactly 22. We now show that $\mathbf{S}$ must exactly be the solution described in above paragraph. We first show the following lemma:

Lemma 2.11. *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. □

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 2.12. *The two $s \rightsquigarrow t$ paths in $\mathbf{S}$ 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 $\mathbf{S}$.*

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

Lemma 2.13. *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 $\mathbf{S}$ are P_1 and P_2, this shows that the weight of $\mathbf{S}$ is at least 23. □

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 2.13), 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 2.13. 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 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 $\ell \in [n]$

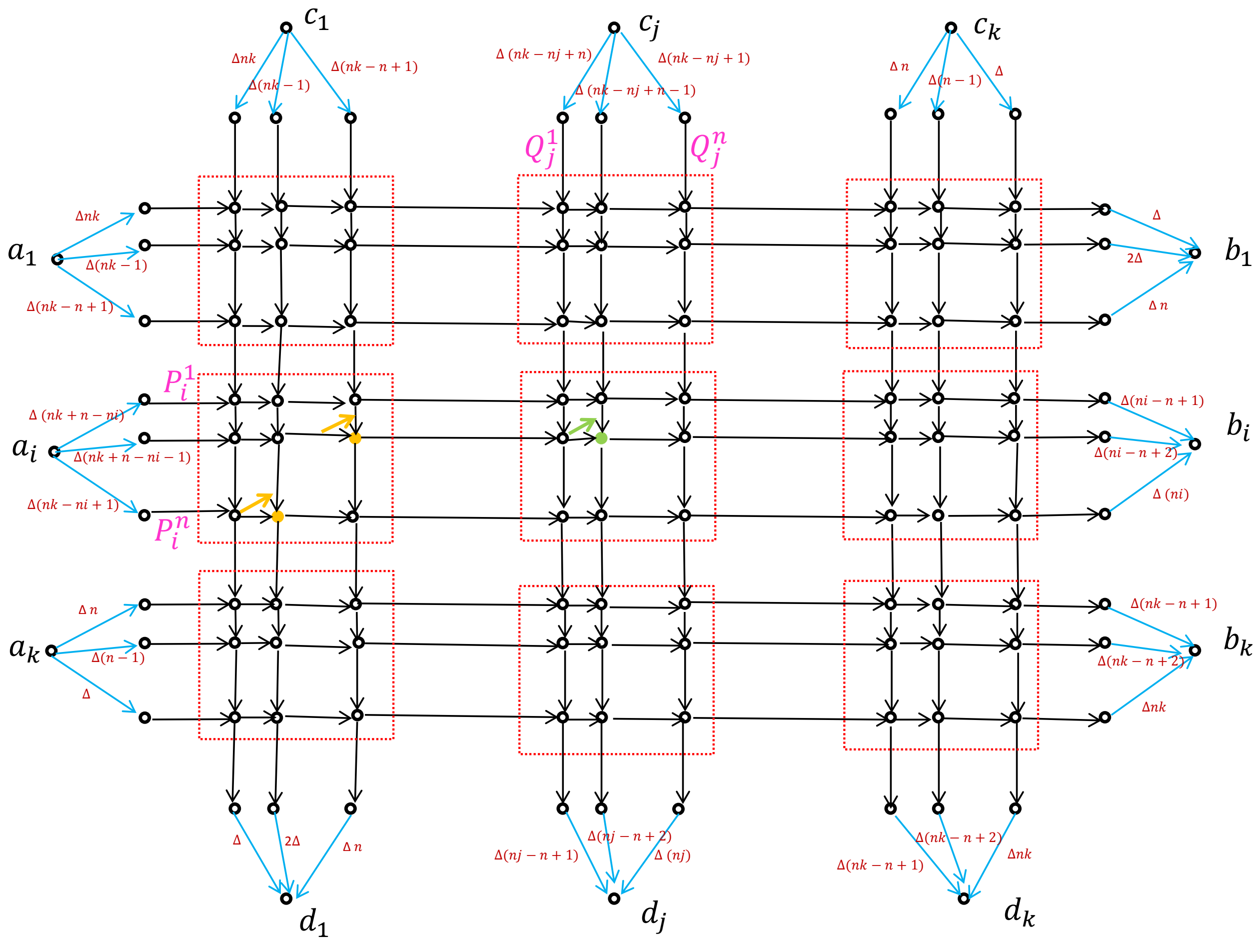

Figure 4: The instance of 2-SCSS-$(k,1)$ created from an instance of Grid Tiling*.

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 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 $\alpha = \Delta(nk+1) + 4(k+1) + 4k(n-1)$.

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 4)

- **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 5. 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

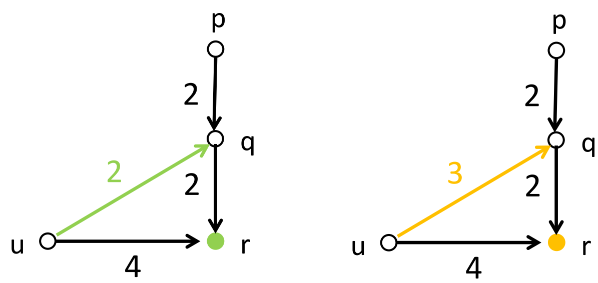

Figure 5: Let u, r be two consecutive vertices on the canonical path say P_i^{ℓ}. Let r be on the canonical path $Q_j^{\ell'}$ and let p be the vertex preceding it on this path. If r is a green (respectively orange) vertex then we subdivide the edge (p, r) by introducing a new vertex q and adding two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

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 5. 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 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. 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: 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. Let

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

We now prove two theorems which together give a reduction from GRID TILING* to 2-SCSS-$(k, 1)$.

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

First we show the easy direction.

Theorem 3.2. GRID TILING* *has a solution implies OPT for* 2-SCSS-$(k,1)$ *is 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 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 δ_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 γ_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 $bi \rightarrow e_i \rightarrow f_i \rightarrow a_{i-1}$. 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 $\alpha \cdot k + W(k-1)$.
- Finally use the edge (b_1, s) of weight 0.

Therefore, with a total weight of $\alpha \cdot k + W(k-1)$ we have a $t \rightsquigarrow s$ 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 $s \rightarrow e_i \rightarrow f_i \rightarrow t$ 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 (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 a $s \rightsquigarrow t$ path whose weight is exactly equal to α. 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 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^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 $\Big(\alpha \cdot k + W(k-1)\Big) + \alpha \cdot k - (k^2+k) = \beta$. $\square$

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

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

Definition 3.3. *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 3.4. *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 $i \in [k]$*

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 $2 \leq i \leq k$.

The above argument also implies that we have a $a_i \rightsquigarrow b_i$ path for each $2 \leq i \leq k$. Since the only coming incoming edge into s is (b_1, s) we must also have a $a_1 \rightsquigarrow b_1$ 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 β.

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

Contradiction. □

Lemma 3.5. *In the optimum solution exactly k of the $s \rightsquigarrow t$ paths cannot use any connector edge.*

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 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 $\{c_1, c_2, \dots, c_k\}$ and the only incoming edges into t are from $\{d_1, d_2, \dots, d_k\}$. Moreover, the $t \rightsquigarrow s$ path in the optimum does not use any blue edge incident on $\{c_1, c_2, \dots, c_k\}$ or $\{d_1, d_2, \dots, d_k\}$ 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 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 $\ge W(k-1) + W = Wk > \beta$, which is a contradiction. □

We call the $s \rightsquigarrow t$ paths described in Lemma 3.5 as **expensive** paths. Note that the only outgoing edges from s are to $\{c_1, c_2, \dots, c_k\}$ and the only incoming edges into t are from $\{d_1, d_2, \dots, d_k\}$. So, we can think of the expensive paths as actually k paths from $\{c_1, c_2, \dots, c_k\}$ to $\{d_1, d_2, \dots, d_k\}$. Since expensive edges do not use any connector edge, the existence of a $c_j \rightsquigarrow d_\ell$ path implies $\ell \ge j$.

Definition 3.6. *For each $i \in [k]$, let λ_i, μ_i denote the number of $c_i \rightsquigarrow d_i, c_i \rightsquigarrow \{d_{i+1}, d_{i+2}, \dots, d_k\}$ expensive paths in the optimum solution.*

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

Lemma 3.7. *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 $\Delta(nk+1)$.*

Proof. From Lemma 3.4, 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 $\ell' \ge \ell$. Hence the sum of weights of the blue edges is given by $\Delta(nk - ni + n + 1 - \ell) + \Delta(ni - n + \ell') = \Delta(nk+1) + (\ell' - \ell) \ge \Delta(nk+1)$. □

Lemma 3.8. *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.*

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_ℓ are $\Delta(nk - nj + 1), \Delta(n\ell - n + 1)$ respectively. Hence, the sum of weights of these edges is $\Delta(nk - nj + 1) + \Delta(n\ell - n + 1) = \Delta(nk+1) + \Delta + \Delta(n(\ell - j - 1)) \geq \Delta(nk+2)$.

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). $\square$

Lemma 3.9. *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* $2k\Big(4(k+1) + 4k(n-1)\Big)$

Proof. From Lemma 3.7, 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 3.8, 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_ℓ for some $k \geq \ell \geq j \geq 1$. Hence, the total weight of blue edges is at least $2k \cdot \Delta(nk+1)$. $\square$

Lemma 3.10. *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 5).*

Proof. From Lemma 3.4, 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 $a_i \rightsquigarrow b_i$ 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 $k\Big(4(k+1) + 4k(n-1)\Big)$. 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_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. 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 $2k\Big(4(k+1) + 4k(n-1)\Big)$. $\square$

Lemma 3.11. *Every expensive path is canonical, i.e.,* $\mu_j = 0$ *for all* $j \in [k]$.

Proof. Suppose an expensive path is not canonical. Hence, from Lemma 3.8, the contribution of the blue edges of this expensive path is $\geq \Delta(nk+2)$. From Lemma 3.9, it follows that the contribution of the blue edges to the optimum is at least $2k \cdot \Delta(nk+1) + \Delta$.

Refer to Figure 5. 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

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

We now claim that the weight of our solution 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 our optimum solution is at least

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

Contradiction. □

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

Definition 3.12. *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 3.13. *For each $i \in [k]$, the optimum solution contains an almost canonical $a_i \rightsquigarrow b_i$ path .*

Proof. Fix some $i \in [k]$. From Lemma 3.4, 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^ℓ and ends with a blue edge from $P_i^{\ell'}$ for some $\ell' > \ell$. Hence, the contribution of these blue edges is $\Delta(nk-ni+n+1-\ell)+\Delta(ni-n+\ell') = \Delta(nk+1)+\Delta(\ell'-1) \ge \Delta(nk+1)+\Delta$. Now, a similar argument as in Lemma 3.11 can be applied to show that the weight of this optimal solution is greater than β. Contradiction. □

Theorem 3.14. *OPT for 2-SCSS-$(k,1)$ is at most β implies the* GRID TILING* *instance has a solution.*

Proof. By Lemma 3.11, we know that $\sum_{i=1}^{k} \lambda_i = k$ and $\lambda_i \ge 0$ 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 3.11 and Lemma 3.13, we know that $\mathcal{X}$ contains

- An $a_i \rightsquigarrow b_i$ almost canonical path for every $1 \le i, j \le k$.
- 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 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 3.13, 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 5). 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 δ 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 (α_i,γ_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. □

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 3.2 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-$(k,1)$ where we want to find k paths from $s\rightsquigarrow t$ and one path from $t\rightsquigarrow s$.

Chen et al. [3] showed for any function f an $f(k)n^{o(k)}$ 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-$(k,1)$, 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. □

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 time $n^{O(k)}$, and that is asymptotically optimal under the ETH. 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, 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.

Acknowledgements: We would like to thank DIMACS for its hospitality where a subset of the authors had fruitful discussions on this 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. STOC '03

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

[11] Marx, D.: On Optimality of Planar & Geometric Approximation Schemes. FOCS'07

[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] 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

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

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

[17] 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)

A Equivalence of Vertex-Weighted and Edge-Weighted Versions

Lemma A.1. *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. □