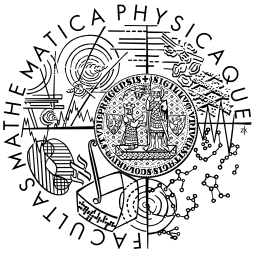

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Tamás Dávid Kemény

Strongly Connected Steiner Subgraphs with Small Number of Steiner Vertices

Department of Applied Mathematics

Supervisor of the bachelor thesis: Dr. Andreas Emil Feldmann
Study programme: Computer Science
Study branch: General Computer Science

Prague 2020

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ..... signature of the author

I would like to thank my supervisor Dr. Andreas Emil Feldmann for his expert guidance and endless patience with me throughout the course of this work, as well as since the very first year of my studies.

Thanks to Dr. Feldmann and prof. RNDr. Jiří Sgall, DrSc., this work received support by project 17-09142S of GAČR - *Modern algorithms: New challenges of complex data sets.*

Title: Strongly Connected Steiner Subgraphs with Small Number of Steiner Vertices

Author: Tamás Dávid Kemény

Department: Department of Applied Mathematics

Supervisor: Dr. Andreas Emil Feldmann, Department of Applied Mathematics

Abstract: Two well-established methods of dealing with hard optimization problems have been to develop approximation and parameterized algorithms. Recent results have shown that for some problems, it is only by combining these two approaches, into so-called *parameterized approximation algorithms*, that we are able to efficiently find solutions that are of reasonable quality. This is the viewpoint from which we study the problem known as the Strongly Connected Steiner Subgraph problem, where a set of *terminal* vertices of an edge-weighted directed graph needs to be strongly-connected in the cheapest way possible.

Keywords: Strongly Connected Steiner Subgraphs, Parameterized Algorithms, Approximation Algorithms, Bidirected Graphs

Contents

1. Introduction

A central focus of computer science has been to develop algorithms since its very conception. It is clear that to keep up with the ever-growing amount of data that needs to be processed today, we must continuously come up with ways to increase the computational power of our devices. However, it should be equally important to ensure the effectiveness of the algorithms that they use, as well as to study computational problems we may face tomorrow.

The aim of this work is to further develop the results of a recent paper by Chitnis et al. [1], regarding a hard optimization problem called the Strongly Connected Steiner Subgraph (SCSS) problem, where a set R of *terminal* vertices of an edge-weighted directed graph needs to be strongly-connected in the cheapest way possible. This problem is known to be NP-hard [2], thus it is widely believed [3, 4] that there is no algorithm with run time polynomial in the size of the input, which finds an optimum solution to every instance.

The problem has also been studied from the perspective of *parameterized algorithms*, whose aim is to isolate any super-polynomial run time to a *parameter* of the input. A parameter in this context is a natural number k describing some property of the input. In practical scenarios, we might be interested in solving only instances where k is relatively small, in which case an algorithm that runs in time $f(k) \cdot n^{\mathcal{O}(1)}$, for some (exponential or even super-exponential) function f, is still considered efficient. If such an algorithm exists, the problem is called *fixed-parameter tractable* (FPT) for the parameter k and belongs to the complexity class FPT. For the SCSS problem we do not believe this to be true when k is the number of terminals, as this is known to be W[1]-hard[1] due to a results by Guo et al. [5].

In spite of this hardness, we still need algorithms that compute solutions to this problem. One way of achieving this is by the use of *approximation algorithms*. By relaxing the requirement that our algorithms must return optimal solutions, we hope to find algorithms that give solutions to otherwise hard problems. It is however crucial for these algorithms to come with a provable guarantee on the quality of their solutions, to differentiate them from heuristics. Such a guarantee might ensure that the solutions have, for example in the case of a minimization problem such as SCSS, cost at most a constant factor c times that of the optimum. We call such an algorithm a *c-approximation algorithm* and the constant c its *approximation factor*. In another paper by Rajesh et al. [6], the authors give an FPT 2-approximation algorithm for the SCSS problem when the parameter is the number of terminals. This combination of the approaches of parameterization and approximation proves to be useful for the SCSS problem, since besides its W[1]-hardness, their algorithm also overcomes another hardness result [2] which states that under reasonable assumptions[2]

[1] although a definiton of W[1]-hardness falls outside the scope of this work, it should be clear to the reader from the context that we simply mean "not in FPT"

[2] this assumption is that $NP \subsetneq ZTIME(n^{polylog(n)})$. Again we will not give a full explanation of the $ZTIME$ complexity class, we merely stress that the statement has an underlying assumption.

there can be no polynomial time $\mathcal{O}(\log^{2-\epsilon} n)$-approximation algorithm.

Although SCSS is not FPT when parameterizing by the number of terminals, it could still be FPT for other choices of the parameter. An alternative parameter to consider is the number of non-terminal vertices, called *Steiner vertices*, in an optimum solution. An FPT algorithm for this parameter means we would be able to efficiently find exact SCSS solutions even if the input has many terminals. Failing this, we hope for a parameterized approximation algorithm which runs efficiently on inputs with few Steiner vertices and computes solutions that are only near-optimal. In Chapter 3 we present our first result, which implies that this selection of the parameter is in fact not a good one.

Theorem 1. *For any constant $c \in \mathbb{N}$, it is* W[1]*-hard to compute a c-approximation to the* Strongly Connected Steiner Subgraph *problem, when parameterizing by the number of Steiner vertices in an optimum solution.*

An example where the combination of the two paradigms does work, is the classical Steiner Tree (ST) problem, which is one of the 21 original NP-hard problems listed by Karp [7]. As their names suggest, the ST and SCSS problems are in fact related. In the ST problem we are also given a set R of terminal vertices, however we are asked to find a tree, called a *Steiner tree*, of least possible weight which connects all terminals in R in an edge-weighted undirected graph. When parameterized by the number of Steiner vertices in an optimum Steiner tree, the problem is known [8] to be W[2]-hard. It is also known [9] that there can be no polynomial approximation algorithm with approximation ratio $\frac{96}{95}$, unless P=NP. In contrast to these hardness results, Dvořak et al. [10] give an FPT approximation algorithm[3] which can in fact achieve this ratio.

The directed variant of ST, called the Directed Steiner Tree (DST) problem, is more closely related to SCSS. In fact the 2-approximate FPT algorithm for SCSS parameterized by the number of terminals, is obtained using an FPT algorithm for DST by Dreyfus et al.[11]. Furthermore, Theorem 1 incorporates a similar proof as to the same hardness result to DST in the paper by Dvořak et al. [10].

We study a special case of the SCSS problem called the bi-SCSS problem, where the set of inputs is restricted to the class of *bidirected* graphs. This family of graphs, despite being directed, has the benefit of a symmetric shortest path function, placing it between undirected and directed graphs. This means the bi-SCSS problem can be seen as a generalization more closely related to the ST than DST. We show that bi-SCSS need not be approximated using ST, instead we compare the structure of their optimum solutions to solutions obtained from cycles.

In the metric Travelling Salesman Problem (TSP) we are given a complete graph and an edge-weight function for which the triangle inequality holds, and are asked to find a cycle visiting each vertex with minimum possible sum of edge-weights. In Chapter 4 we show that solutions to TSP can give bi-SCSS solutions that are near-optimal. Since the TSP is a well-studied problem, this opens up the possibility of solving bi-SCSS using known techniques, giving us Theorems 2 and 3.

[3] technically they present not a single algorithm, but rather an algorithm schema.

Theorem 2. *There is a polynomial time algorithm for* BI-SCSS *that computes a 2-approximation.*

We show that the approximation ratio of our algorithm in Theorem 2 can be improved when parameterized by the number of terminals, using an exponential algorithm for TSP. This allows for a trade-off between runtime and solution quality, resulting in an approximation ratio that is quite unusual.

Theorem 3. *There is an algorithm for* BI-SCSS *that computes a* $\frac{4}{3}$*-approximation in time* $2^{\mathcal{O}(|R|)} \cdot poly(n)$.

The last Steiner problem we will relate to our problem is a generalization of SCSS. It is the Directed Steiner Network (DSN) problem, which takes as input a directed edge-weighted graph and a set of ordered demand pairs. The aim is to compute the cheapest network which for each demand (u, v) guarantees a directed path from u to v. The SCSS problem can be seen as the special case of DSN, where the demand pairs form a graph that is strongly-connected. This means that the SCSS problem can be no harder than DSN. In fact, the SCSS problem is strictly easier to approximate than DSN, since there is an FPT 2-approximation algorithm for SCSS, but not for DSN [1]. Furthermore, the BI-SCSS problem has an FPT exact algorithm, while BI-DSN does not [1]. On the next page, Table 1 shows that our results also imply that the SCSS problem is similarly strictly easier to approximate than DSN, when restricting both problems to bidirected inputs. This reaffirms the previous observation relating the two problems.

problem	algorithm	
	approx. ratio	runtime
BI-DSN	1	$n^{\mathcal{O}(k)}$
BI-DSN	2	$2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$
BI-DSN	4	$n^{\mathcal{O}(1)}$
BI-SCSS	1	$2^{\mathcal{O}(2^{k^2-k})} \cdot n^{\mathcal{O}(1)}$
BI-SCSS	$\frac{4}{3}$	$2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$
BI-SCSS	2	$n^{\mathcal{O}(1)}$

Table 1. *An extract from [1] presenting some known results for the* BI-DSN *and* BI-SCSS *problems. We have added our own Theorems 2 and 3 at the bottom. For* BI-SCSS *the parameter k is the number of terminals, while for* BI-DSN *it is the number of demand pairs.*

2. Preliminaries

We assume that the reader already has an understanding of most of the concepts used in this work. The aim of this work is to apply algorithms, proof techniques and any knowledge gained during undergraduate studies, rather than to provide a summary of already known results. Thus, we will not give definitions that are not specific to our work, but instead recommend related literature. For a textbook covering the core material of graph theory, we refer the reader to Reinhard Diestel's *Graph Theory: Graduate Texts in Mathematics* [12]. Knowledge of parameterized algorithms and the theory surrounding it is also crucial, for this we recommend the recent textbook titled *Parameterized Algorithms*, by Marek Cygan et al. [13]. Furthermore, *The Design of Approximation Algorithms* by David P. Williamson and David B. Shmoys [14] serves as a reference to both students and researchers in the field of approximation algorithms. Finally, a textbook by Bernhard Korte and Jens Vygen titled *Combinatorial Optimization* [16] is an indispensable source of information for algorithmic theory.

2.1 Definitions specific to our work

Definition 1. Given a directed graph $G = (V, A)$ and a set of terminals $R \subseteq V$, a subgraph $N \subseteq G$ is called a *strongly-connected Steiner subgraph* if the following holds:

1. N contains all the terminals i.e. $R \subseteq V(N)$
2. N is strongly-connected i.e. N contains a uv-path for every $u, v \in V(N)$

Strongly Connected Steiner Subgraph (SCSS) problem	
Instances:	directed graphs $G = (V, A)$, with weight function $w : A \to \mathbb{R}^+$, and a set of terminals $R \subseteq V$
Solutions:	strongly-connected Steiner subgraphs $N \subseteq G$
Objective:	minimize $w(N) \coloneqq \sum_{a \in A(N)} w(a)$

Definition 2. A *bidirected* graph is a directed graph $G = (V, A)$ an with edge-weight function $w : A \to \mathbb{R}^+$, such that for every arc $uv \in A$ there exists the reverse arc $vu \in A$ and has the same weight.

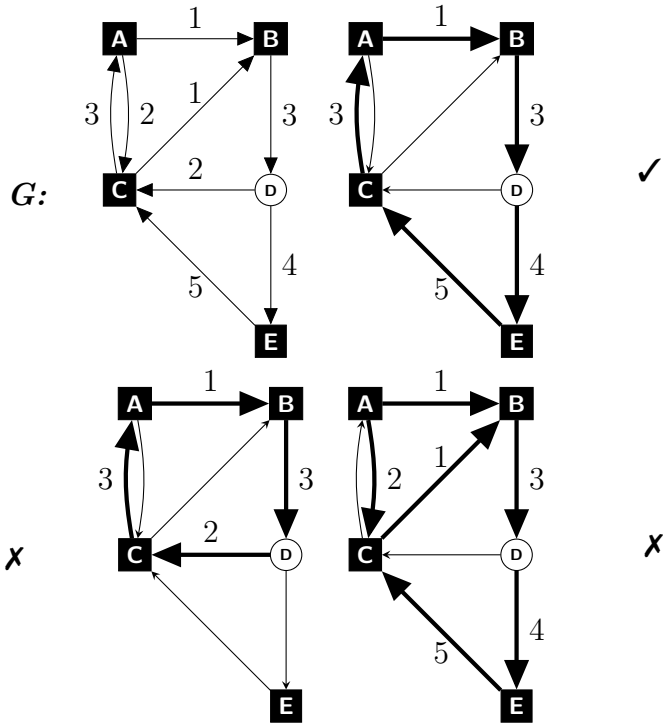

Figure 1. *An instance of the SCSS problem. On the top left is the input graph, while the other three graphs show, using thick edges, possible solutions N - of which only the one on the top right is feasible. On the bottom left, N does not contain the terminal E and on the bottom right, there are no paths in N that end at the terminal A.*

2.2 Useful techniques

In order to avoid cluttered proofs we describe in this section, some techniques that are less common in related literature, but are specifically used during this work.

Metric completion. The metric completion or *metric closure* of an undirected graph is defined as the undirected complete graph on the same vertex set, with its edge set given by the length of the shortest path between pairs of vertices. In an attempt to extend this notion to directed graphs, it becomes obvious that since the length of the shortest path from u to v and from v to u may differ, the shortest path function no longer forms a metric and can only be modelled by a complete graph that is directed. In *bidirected* graphs this symmetry is given - for each arc uv there exists the reverse arc vu having the same weight. With this in mind during Chapter 4, we will consider the metric completion of a bidirected graph as also being undirected.

Shortcutting. Given an Eulerian tour, i.e, a closed walk that uses each edge once, we can construct from it a cycle, that visits each vertex once, in the following way: Suppose the Eulerian tour visits all vertices $v_1, v_2, ..., v_n$ of G in the order given by the vertex sequence $S = (u_1, u_2, ...u_k, u_1)$. After we have visited a vertex v_i, unless it is the first vertex, we are no longer interested in returning to it. This means we may remove subsequent occurrences of v_i in S, i.e., we remove all u_j such that $u_j = v_i$, to obtain a sequence S' which visits v_i exactly once. For the first vertex, we remove all but its first and last occurrences. By applying this process to each vertex, we obtain a cycle that visits each vertex exactly once. Furthermore, by skipping a repeated vertex and directly going to the next unvisited vertex using the shortest path, the total distance traversed by the cycle is no greater than that of the original Eulerian tour. In terms of the underlying graph, this process can be seen as replacing parts of the tour by edges from the metric completion of the graph. We may also be interested in skipping certain vertices entirely, not just their repeated occurrences. We will refer to this as shortcutting a tour to a certain set V', thereby skipping the vertices not in V'.

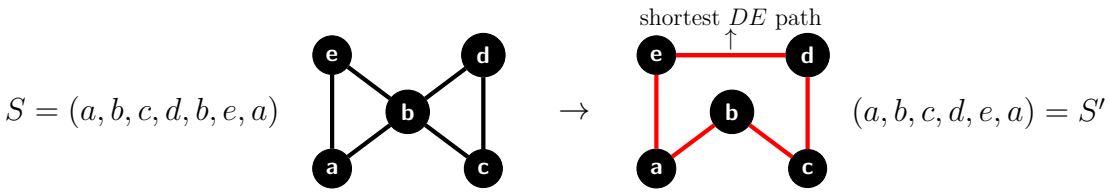

Figure 2. *An Eulerian tour (left) and the cycle (right) obtained by applying the shortcutting technique to remove the repeated occurrence of the vertex b. The edge de on the right is defined as the shortest among all paths from d to e, one of which is the path (d, b, e) used by S.*

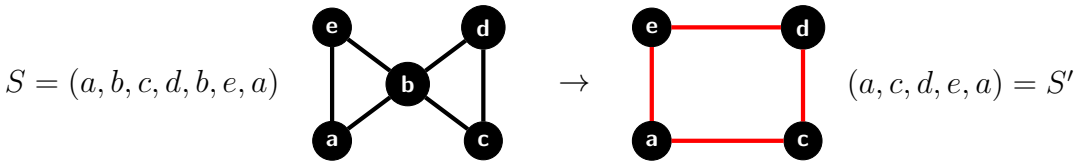

Figure 3. *An Eulerian tour (left) and the cycle (right) obtained by shortcutting to the set $V' = \{a, c, d, e\}$.*

3. Reducing DS to SCSS

In the recent paper by Dvořák et al. [10] the authors present a parameterized reduction from the Dominating Set (DS) problem to the Directed Steiner Tree (DST) problem. The authors combine this reduction with an earlier finding [15] which proves that there is no efficient constant approximation algorithm for DS, even when parameterizing by the size of the optimum solution, to obtain the same inapproximability result for DST. In this section we show that this reduction can be modified to the Strongly Connected Steiner Subgraph (SCSS) problem. More specifically, we give a parameterized reduction from the Dominating Set problem to the Strongly Connected Steiner Subgraph problem, where the parameters considered are the size of the dominating set and the number of Steiner vertices in the solution, respectively.

Definition 3. Given an undirected graph $G = (V, E)$, a set $X \subseteq V$ is called a *dominating set* if every vertex of the graph is in X or has a neighbour in X. More formally, X is a dominating set if and only if $\bigcup_{v \in X} N[v] = V$, where $N[v]$ is the closed neighbourhood of v.

Dominating Set (DS) problem	
Instances:	undirected graphs $G = (V, E)$
Solutions:	dominating sets $X \subseteq V$
Objective:	minimize $\lvert X \rvert$

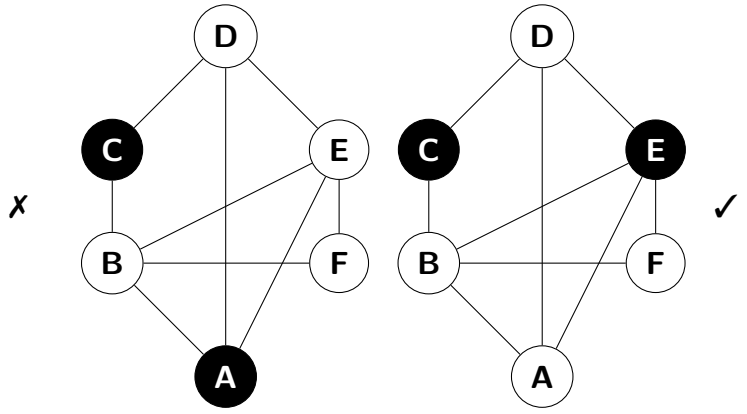

Figure 4. *A graph with two candidates for the set X. The shaded vertices on the right form a dominating set, whereas the vertex F on the left is neither in X nor has any neighbour in X.*

Deciding if a given graph contains a dominating set of size at most k is a notoriously hard problem. It is the most well-known problem complete for the class W[2] [15], i.e., there is no algorithm for solving it in time $f(k) \cdot n^{\mathcal{O}(1)}$, unless FPT = W[2].

Lemma 1. *For any input graph $G = (V, E)$ of* Dominating Set *we can construct an instance (H, R, w) of weighted* Strongly Connected Steiner Subgraph *such that G contains a dominating set X of size k if and only if H contains a strongly-connected Steiner subgraph N with k Steiner vertices.*

For an overview of the constructed instance, see Figure 6 on the next page.

Proof. Let us fix the notation $\{v_1, v_2, ..., v_n\} = V$ and $m = |E|$. We construct the graph $H = (W, A)$ having $2n+1$ vertices and $3n+2m$ arcs from the following sets. Let the vertex set W consist of a Steiner vertex s_i and a terminal t_i for each $v_i \in V$, and an auxiliary terminal r called the *root*. For each $v_i \in V$ we add arcs of the form rs_i into A, which we will call *selector* arcs, whose role will be to incur a cost associated with selecting a Steiner vertex into a solution. For each edge $v_iv_j \in E$, we create arcs $s_it_j, s_jt_i \in A$, that can be viewed as an encoding of neighbourhoods of vertices from V into H. However, since a vertex that is in the dominating set need not be adjacent to any vertex of the dominating set, by paying for a Steiner vertex s_i, we must satisfy the demand created by its corresponding terminal t_i. Thus, we also add for each $v_i \in V$ the arc $s_it_i \in A$, which we will call *self* arcs, to complete the previous set of arcs into an encoding of the *closed* neighbourhood of v_i. The last type of arc that we construct, called *return* arcs, are of the form $t_ir \in A$ and ensure that the terminals can be strongly-connected. Finally, we define the weight function $w : A \to \mathbb{R}^+$ as $w(a) \coloneqq 1$ if a is a selector arc, and $w(a) \coloneqq 0$ otherwise.

Suppose H contains a strongly-connected Steiner subgraph N with k Steiner vertices. Let $X \coloneqq \{v_i \in V | s_i \in V(N)\}$, i.e., we define X to be the set of vertices v_i of G such that N contains the corresponding Steiner vertex s_i. We claim that X is a dominating set of size k. From the initial hypothesis and the definition of X, there are k vertices in X. Now, we want to show that every vertex of G is either in X or has a neighbour in X. Since N is strongly-connected and all of its paths that reach some terminal t_i must first go through a Steiner vertex, the set of k Steiner vertices in N has at least one outgoing arc to each of the terminals. This means, the set of vertices X, which corresponds to exactly these k Steiner vertices, has a closed neighbourhood that contains all terminals when encoded into N, or equivalently, has a closed neighbourhood that contains all vertices in G, since all terminals in H correspond to a vertex in G. Thus, X is a dominating set of size k.

Now for the other direction, suppose we have a dominating set X of size k, we show that we can construct a strongly-connected Steiner subgraph N with k Steiner vertices. Since N must contain all terminals to be a feasible solution, add all $n + 1$ terminals to $V(N)$. For each $v_i \in X$, of which there are k, add s_i into $V(N)$. No other Steiner vertices will be added to N. Since N must contain paths from the root r that reach all terminals, and this is only possible through selector arcs, let $rs_i \in A(N)$ for all $s_i \in V(N)$. Furthermore, all arcs leaving the Steiner vertices $s_i \in V(N)$ have cost 0, thus we may add them for free. Note that since X is a dominating set and since these arcs encode exactly the closed neighbourhood of X, the addition of these arcs will connect the root to all of the terminals.

Finally, we add all return arcs as they are also free of charge, and guarantee the existence of all paths $t_i r \in N$. In conclusion, there exists a path from the root to each terminal, as well as from the terminal back to the root, therefore the solution is strongly-connected. □

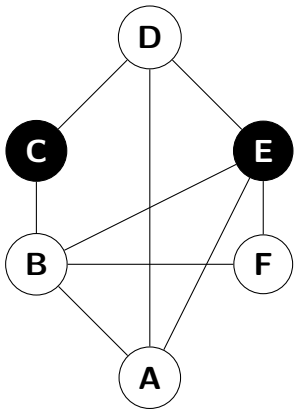

Figure 5. *An instance of the* Dominating set *problem. The highlighted vertices form a dominating set.*

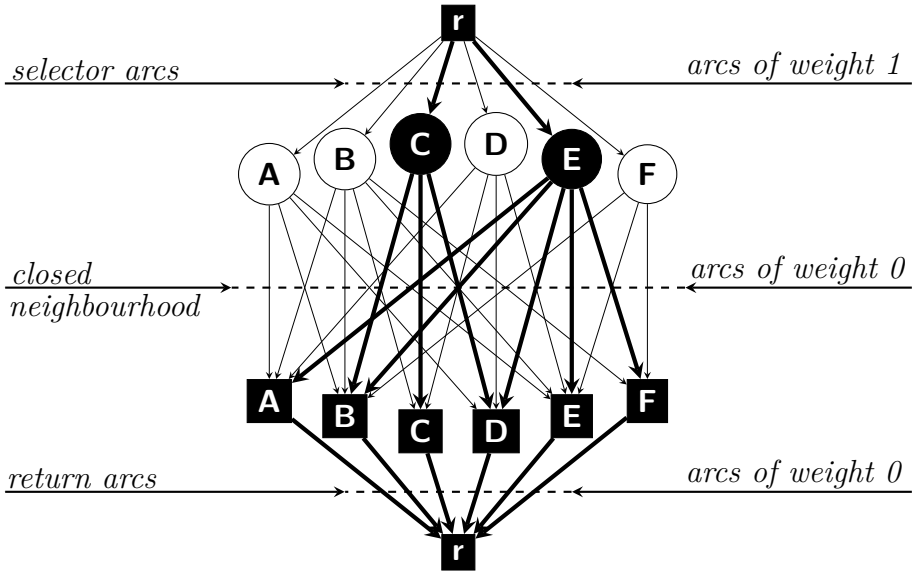

Figure 6. *The instance of weighted* Strongly Connected Steiner Subgraph *obtained by applying the reduction to Figure 5. The thick edges and highlighted vertices form a strongly-connected Steiner subgraph. The closed neighbourhoods of vertices in Figure 5 are encoded by the edges in the middle. Note that we have drawn the root r twice.*

Notation. Let X_{opt} be a dominating set of G of minimum size and N_{OPT} be a strongly-connected Steiner subgraph of H of minimum weight. Furthermore, let $s(N)$ be the number of Steiner vertices in some strongly-connected Steiner subgraph N.

Theorem 4 (Yijia Chen and Bingkai Lin [15])**.** *Unless* FPT = W[1]*, for every constant c and computable function f, there is no algorithm $\mathbb{A}$ which runs in time $f(|X|) \cdot n^{\mathcal{O}(1)}$ and outputs a dominating set X of size at most $c \cdot |X_{\text{OPT}}|$.*

We claim in Theorem 1 that unless FPT = W[1], there is no algorithm $\mathbb{B}$, which for any constant $c \in \mathbb{R}$, outputs a strongly-connected Steiner subgraph N of weight at most $c \cdot w(N_{\text{OPT}})$ and runs in time $f(s(N_{\text{OPT}})) \cdot n^{\mathcal{O}(1)}$.

Proof of Theorem 1. Suppose such an algorithm $\mathbb{B}$ and constant c exists. We show that this implies a contradiction with Theorem 4, as an algorithm $\mathbb{A}$ would be able to perform the following steps:

1. Given an instance G of Dominating Set, we construct in polynomial time an instance (H, R, w) of weighted Strongly Connected Steiner Subgraph, using the reduction in Lemma 1.

2. By our hypothesis, the algorithm $\mathbb{B}$ with input (H, R, w) outputs a strongly-connected Steiner subgraph N, in time $f(s(N_{\text{OPT}})) \cdot n^{\mathcal{O}(1)}$, such that $w(N) \leq c \cdot w(N_{\text{OPT}})$.

3. Using the method described in the proof of Lemma 1, we construct from N, in time polynomial in the size of N, the dominating set X.

4. Return X as the output of $\mathbb{A}$.

According to its construction, the graph H has $2{\cdot}|V(G)|{+}1$ vertices and $3{\cdot}|V(G)|{+}2{\cdot}|E(G)|$ arcs, thus, the size of the solution N, which is at most the size of its supergraph H, is polynomially bounded in the size of G. This means that the time taken to perform step 3 is polynomial in the size of the input. The run time of $\mathbb{A}$ is dominated by the run time of step 2, since the function f is unbounded. Now we would like to show that the guarantee on the quality of solutions of $\mathbb{B}$ translates into the same guarantee for $\mathbb{A}$. By Lemma 1 we have the equalities $|X| = s(N)$ and $|X_{\text{OPT}}| = s(N_{\text{OPT}})$. Since in both N and N_{OPT} only the selector arcs have weight 1 and there is exactly one Steiner vertex for each selector arc, $s(N) = w(N)$ and $s(N_{\text{OPT}}) = w(N_{\text{OPT}})$.
To summarize, we have a solution X which has size:

$$|X| = s(N) = w(N) \leq c \cdot w(N_{\text{OPT}}) = c \cdot s(N_{\text{OPT}}) = c \cdot |X_{\text{OPT}}|$$

and was computed in time:

$$f(s(N_{\text{OPT}})) \cdot n^{\mathcal{O}(1)} \leq f(c \cdot |X_{\text{OPT}}|) \cdot n^{\mathcal{O}(1)} = g(|X_{\text{OPT}}|) \cdot n^{\mathcal{O}(1)} \leq g(|X|) \cdot n^{\mathcal{O}(1)}$$

We have arrived at a contradiction with Theorem 4. □

4. Approximating BISCSS via TSP

In this section we show that TSP tours give near-optimal BI-SCSS solutions that can be computed efficiently when parameterizing by the number of terminals. We describe how to transform BI-SCSS instances into TSP instances and how to output BI-SCSS solutions from TSP solutions. The majority of the work concerns proving the quality of these solutions. We show that the optimal TSP solution is no more than a constant factor worse than the optimal BI-SCSS solution and we give a tight lower bound on this constant. First we note the differences in input graphs, namely that the TSP takes an undirected, complete graph while the BI-SCSS problem takes a bidirected graph as input.

Given a bidirected graph G we consider its underlying undirected graph, obtained by substituting each arc-pair (uv, vu) by a single undirected edge uv. Since we are only interested in connecting a terminal set R, running a TSP algorithm on the whole graph G is not needed. Instead, it is practical to compute the metric completion only on the set R, which will be our input to some TSP algorithm. Let H denote this complete graph on $|R|$ vertices corresponding to the terminals, with edge weights given by the shortest path metric.

Now suppose we have a TSP tour T of H. We would like to extract a feasible BI-SCSS solution from T. We may replace its edges by their corresponding shortest paths in G to obtain the set of paths $\mathcal{P}(T)$, but not before assigning them some direction. A simple way to do this, in order to ensure that the solution N is strongly-connected, is to orient the paths according to a strong orientation of T. Note that multiple copies of the same arc (from a pair of paths with a common arc) are not only redundant in N, but would form a multigraph, which cannot be a subgraph of G. Finally since T visits all vertices of H, N contains all terminals, i.e. N is a feasible BI-SCSS solution.

An algorithm that makes use of this reduction can be summarized using the previous notation in terms of the following steps:

1. Given G and R compute H
2. Run a TSP algorithm on H to get a tour T
3. Compute N from T
4. Output N

We mention that steps 1 and 3 can be done in polynomial time, while step 2 will have run time which is some function of $|R|$. Note that the TSP algorithm need not be exact but may compute an approximate solution, thus we use the subscripted notation T_{ALG} of the tour obtained in step 2 to distinguish from the optimum TSP tour T_{OPT} in the graph H. Similarly, let N_{ALG} reference N from step 3 in contrast to the optimum N_{OPT}.

In addition to the approximation ratio of the TSP algorithm used in step 2, the quality of N_{ALG} is also worsened by the fact that T_{OPT} can have weight strictly great than that of N_{OPT}, as seen in Figure 7. This will be the topic studied in the next section and the determining factor, besides the approximation ratio of the TSP algorithm, of the quality of our solutions.

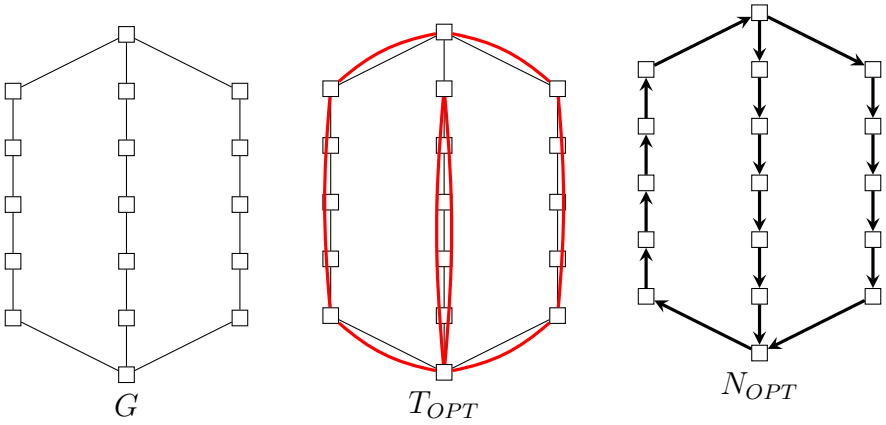

Figure 7. *An instance of* BI-SCSS *such that* $w(T_{OPT}) > w(N_{OPT})$. *Note that all vertices are terminals and for simplicity we have drawn* G *as undirected instead of bidirected. Assuming all arcs of* G *have unit weight,* $w(T_{OPT}) = 22$ *and* $w(N_{OPT}) = 18$.

To show the quality of our output we will prove, for different values of α, that the following inequality holds:

$$w(T_{OPT}) \leq \alpha \cdot w(N_{OPT}) \tag{1}$$

First let us establish that this inequality does in fact yield a guarantee on the quality of our output N_{ALG}.

Lemma 2. *Given a constant $\alpha \in \mathbb{R}$ for which (1) holds and the approximation ratio β of the* TSP *algorithm $\mathcal{B}$ used in step 2, we have that:*

$$w(N_{ALG}) \leq \alpha \cdot \beta \cdot w(N_{OPT})$$

Proof. By definition of the β-approximate tour T_{ALG} obtained from $\mathcal{B}$:

$$w(T_{ALG}) \leq \beta \cdot w(T_{OPT})$$

Multiply both sides of (1) by β and combine with the inequality above:

$$w(T_{ALG}) \leq \beta \cdot w(T_{OPT}) \leq \alpha \cdot \beta \cdot w(N_{OPT})$$

Since in step 3 we remove duplicate arcs, N_{ALG} has weight no greater than that of T_{ALG}:

$$w(N_{ALG}) \leq w(T_{ALG}) \leq \alpha \cdot \beta \cdot w(N_{OPT}) \qquad \square$$

We give a framework which will simplify proving (1) for different values of α. We are interested in making the weights of N_{OPT} and T_{OPT} comparable. To do this we introduce the following problem:

The *Chinese Postman problem* [16] or *Route Inspection problem* is to find a shortest closed tour, called a postman tour, that visits every edge of an undirected graph at least once. If the graph has an Eulerian tour, then that tour is an optimal solution since it visits no edge more than once. Otherwise, we want to find the smallest number of edges (or a set of edges with minimum total weight) that must be repeated so that the tour induces a multigraph that is Eulerian.

Notation. Let $\bar{N}_{OPT}$ denote the underlying undirected multigraph of N_{OPT}, obtained by replacing each arc by an undirected edge.

Lemma 3. *Given a postman tour of $\bar{N}_{OPT}$, i.e., an Eulerian multigraph M containing $\bar{N}_{OPT}$ as a subgraph, we have that (1) holds for $\alpha \coloneqq \frac{w(M)}{w(N_{OPT})}$.*

Proof. Suppose not, i.e., $w(T_{OPT}) > \alpha \cdot w(N_{OPT}) = w(M)$. Using the technique mentioned in section 2.2, the postman tour can be shortcutted to the terminal set R to obtain a TSP tour T with weight $w(T) \leq w(M)$. This contradicts the optimality of T_{OPT}. $\square$

In other words, each postman tour is witness to (1) with its corresponding value of α. This proves useful since now we may focus our efforts on finding the least upperbound on the minimum weight of a postman tour of $\bar{N}_{OPT}$ to get the best possible guarantee on the quality of N_{ALG}. Note, that at no point during the algorithm do we solve the *Chinese Postman problem.*

Definition 4 (T-join)**.** Given a graph $G = (V, E)$, a set of edges $J \subseteq E$ is called a *T-join* for an even set $T \subseteq V$, if for every $v \in V$ we have that $v \in T$ if and only if $\deg_{(V,J)}(v)$ is odd.

It follows from the definition of a $\{v \in V \mid \deg(v) \text{ is odd}\}$-join J that the graph (V, J) has exactly the same set of odd degree vertices. Hence, in the multigraph $(V, E \dot\cup J)$, the degrees of previously odd degree vertices are doubled, i.e., become even, while the rest of the vertices can only have their degrees increased by an even amount. Thus, $(V, E \dot\cup J)$ is an Eulerian multigraph. For brevity, let us call a $\{v \in V \mid \deg(v) \text{ is odd}\}$-join an *odd*-join, as well as to avoid confusion with the TSP tour used in step 2 of the algorithm, which we have already denoted by T.

Remark. Note that the corollary of the handshaking lemma ensures that the set $\{v \in V \mid \deg(v) \text{ is odd}\}$ is necessarily an even set.

Remark. Every graph $G = (V, E)$ has the trivial odd-join $J = E$.

Lemma 4. *(1) holds for $\alpha = 2$*

Proof. The trivial odd-join of $\bar{N}_{OPT}$ gives us an Eulerian multigraph with weight exactly $2 \cdot w(\bar{N}_{OPT})$. Thus, by Lemma 3 , (1) holds for $\alpha = 2$. □

This technique of doubling every edge is a simple way of establishing the following fact: the optimal TSP solution has cost no more that twice that of the optimal bi-SCSS solution. It is worth mentioning that since the metric TSP has a polynomial-time 2-approximation algorithm [16], which uses a similar technique of doubling edges, we already have enough to prove that bi-SCSS admits a polynomial-time 4-approximation. However, we would like to obtain a better approximation factor than this by showing that *(1)* holds for smaller values of α, as well as by using a better algorithm for approximating TSP in step 2.

For $\alpha = \frac{3}{2}$ we will use additional lemmas, aimed at exploiting the strongly-connected property of our bi-SCSS solutions.

Notation. Given a directed graph G, we denote by $\bar{G}$ the underlying undirected multigraph obtained by replacing each arc by an undirected edge.

Remark. Note, that it is since a pair of vertices u and v in a directed graph G may be connected by the reverse arcs uv and vu, that $\bar{G}$ is a multigraph.

Lemma 5. *$\bar{N}_{OPT}$, which is the underlying undirected multigraph of N_{OPT}, is bridgeless.*

Proof. Suppose not, i.e., $\bar{N}_{OPT}$ contains a bridge with endpoints u and v, then in N_{OPT} either u is unreachable from v or v is unreachable from u. This is a contradiction since N_{OPT} is strongly-connected. □

Lemma 6 (Symmetric difference for T-joins [17])**.** *J is a T-join and J' is a T'-join if and only if $J \Delta J'$ is a $(T \Delta T')$-join.*

Proof. Fix $v \in T\Delta T'$. We want to show that $|\delta(v) \cap (J\Delta J')| = \deg_{(V,J\Delta J')}(v)$ is odd.

$$\begin{aligned}
v \in T\Delta T' &\iff v \in T \text{ and } v \notin T', \text{ or, } v \notin T \text{ and } v \in T' \\
&\iff |\delta(v) \cap J| \text{ is odd and } |\delta(v) \cap J'| \text{ is even,} \\
&\quad \text{or, } |\delta(v) \cap J| \text{ is even and } |\delta(v) \cap J'| \text{ is odd} \\
&\iff |\delta(v) \cap J| + |\delta(v) \cap J'| \equiv 1 \pmod 2 \\
&\iff |(\delta(v) \cap J) \cup (\delta(v) \cap J')| + |(\delta(v) \cap J) \cap (\delta(v) \cap J')| \equiv 1 \pmod 2 \\
&\iff |\delta(v) \cap (J \cup J')| + |\delta(v) \cap J \cap J'| \equiv 1 \pmod 2 \\
&\iff |\delta(v) \cap (J \cup J')| - |\delta(v) \cap J \cap J'| \equiv 1 \pmod 2 \\
&\iff |\delta(v) \cap ((J \cup J')\backslash(J \cap J'))| = |\delta(v) \cap (J\Delta J')| \equiv 1 \pmod 2
\end{aligned}$$

Note that the last step follows since $(J \cap J') \subseteq (J \cup J')$. □

We use Lemma 6 in the following statement, which uses the intuition that a minimum weight odd-join will always use the cheaper half of a cycle:

Lemma 7 (Cycle lemma)**.** *Given a cycle C in a graph $G = (V, E)$ with edge weights given by $w : E \to \mathbb{R}^+$ and a minimum weight odd-join J we have that $w(J \cap C) \leq \frac{1}{2} \cdot w(C)$*

Proof. Suppose $w(J \cap C) > \frac{1}{2} \cdot w(C)$. Since all vertices of cycles have degree 2, C is an $\emptyset$-join. Using Lemma 6 we have that $J\Delta C$ is again a $(T\Delta\emptyset = T)$-join and has weight:

$$\begin{aligned}
w(J\Delta C) &= w(J \cup C) - w(J \cap C) \\
&= w(J) + w(C) - 2 \cdot w(J \cap C) \\
&< w(J) + 2 \cdot w(J \cap C) - 2 \cdot w(J \cap C) \\
&= w(J)
\end{aligned}$$

Hence, $w(J\Delta C) < w(J)$, which contradicts the minimality of J. □

Now, we would like to apply Lemma 7 to the cycles of $\bar{N}_{OPT}$, however, since these solutions are not necessarily decomposable into edge-disjoint cycles, we will use so-called *cycle k-covers*.

Definition 5 (Cycle k-cover)**.** A *cycle k-cover* of a graph G is a set of cycles $\mathcal{C}$ which cover every edge of G exactly k times.

Due to the famous Cycle Double Cover Conjecture by Szekeres [18] and Seymour [19], the existence of cycle covers is a well-studied topic in many graph types. Luckily we need not rely on the conjecture, but instead can use a known result:

Theorem 5 (Bermond et al. [20])**.** *Every bridgeless multigraph has a cycle 4-cover.*

Lemma 8. *Every bridgeless multigraph $G = (V, E)$ has an odd-join J with $w(J) \leq \frac{1}{2} \cdot w(E)$.*

Proof. From Theorem 5 we have that G has a cycle 4-cover $\mathcal{C}$. Let J be an odd-join of minimum weight. Then we have:

$$4 \cdot w(J) = \sum_{C \in \mathcal{C}} w(J \cap C) \leq \sum_{C \in \mathcal{C}} \frac{1}{2} \cdot w(C) = \frac{1}{2} \sum_{C \in \mathcal{C}} w(C) = 2 \cdot w(E)$$

Hence, $2 \cdot w(J) \leq w(E)$.
The first and last equalities hold since the cycles of $\mathcal{C}$ cover each edge of G exactly 4 times, while the inequality is obtained by applying Lemma 7 to each $C \in \mathcal{C}$. □

Lemma 9. *(1) holds for* $\alpha = \frac{3}{2}$

Proof. Let J be a minimum weight odd-join of $\bar{N}_{OPT}$. By Lemma 5 $\bar{N}_{OPT}$ is bridgeless and so by the Lemma 8, J has weight $w(J) \leq \frac{1}{2} \cdot w(\bar{N}_{OPT})$. Hence, the edges of $\bar{N}_{OPT}$ and J form an Eulerian multigraph with weight at most $(1+\frac{1}{2}) \cdot w(N_{OPT})$. Apply Lemma 3 to conclude the proof. □

A classical result by Christophides [16], proves that there is an algorithm that computes a $\frac{3}{2}$-approximation in polynomial time to TSP. We mention that the existence of this algorithm and proof of the previous lemma, implies that there exists a polynomial-time $(\frac{3}{2} \cdot \frac{3}{2} = \frac{9}{4})$-approximation by application of Lemma 2 with $\alpha = \beta = \frac{3}{2}$.

Finally, we use a result by Bermond et al. [20] which shows that every bridgeless multigraph contains an Eulerian subgraph with at least $\frac{2}{3}$ of all edges. It is by doubling edges not in this Eulerian subgraph (of which there are at most $\frac{1}{3}$ of all edges) that they obtain an Eulerian multigraph that contains the original graph. Since an edge with weight w can be replaced by w edges of unit weight, their result can be interpreted to say the following equivalent statement regarding weighted graphs:

Theorem 6 (Bermond et al. [20])**.** *Every bridgeless multigraph* $G = (V, E)$ *has a postman tour with weight at most* $\frac{4}{3} \cdot w(E)$.

Lemma 10. *(1) holds for* $\alpha = \frac{4}{3}$

Proof. By Lemma 5 $\bar{N}_{OPT}$ is bridgeless, so Theorem 6 guarantees $\bar{N}_{OPT}$ has a postman tour with weight at most $\frac{4}{3} \cdot w(N_{OPT})$. Again, the rest follows by applying Lemma 3. □

We now show that this is the best possible upperbound, by giving an infinite class of graphs G_k such that $\lim_{k\to\infty} \frac{w(T_{OPT})}{w(N_{OPT})} = \frac{4}{3}$. For an overview, see Figure 8.

Theorem 7. *The inequality $w(T_{OPT}) \leq \frac{4}{3} \cdot w(N_{OPT})$ is tight.*

Proof. Let G_k be the undirected graph on $3k-4$ terminals with $3k$ unit weight edges, consisting of 3 edge-disjoint paths, each of length k, with the same endpoints s and t. The optimum TSP solution must visit one of the paths twice (with the exception of the last edge) in order to return to the starting vertex, thus has weight $4k-2$. The optimum BI-SCSS solution consists of 1 directed copy of each edge, i.e., has weight $3k$. □

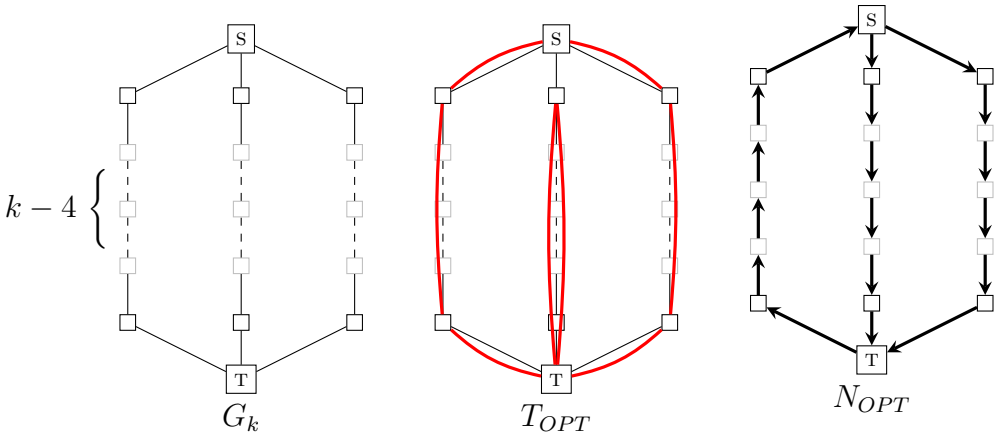

Figure 8. *An instance of* BI-SCSS *showing that the inequality is tight. Again, we have drawn G_k as undirected instead of bidirected for a clearer picture. The dashed edges have variable length, such the total length from s to t along any of the 3 paths is k*

Proof of Theorem 2. Use Christophides' algorithm in step 2 to get a $\frac{3}{2}$-approximate TSP tour T in polynomial time. Apply Lemma 2 with $\alpha = \frac{4}{3}$ and $\beta = \frac{3}{2}$, to get that N_{ALG} is a $(\frac{4}{3} \cdot \frac{3}{2} = 2)$-approximate solution to BI-SCSS . □

Although it is convenient that the two fractional values of α and β cancel out to give the approximation ratio of exactly 2, we may want to spend extra time computing a more accurate TSP tour T in exchange for a solution of better quality. For this, we mention that the *Bellman-Held-Karp* algorithm [21] uses a dynamic programming approach to compute the exact TSP tour with run time exponential in the number of vertices, which in our case is the number of terminals.

Proof of Theorem 3. In step 2, use the *Bellman-Held-Karp* algorithm on H which computes T_{OPT} and has run time $2^{\mathcal{O}(|R|)} \cdot poly(|R|)$, since H has $|R|$ vertices. Apply Lemma 2 with $\alpha = \frac{4}{3}$ and $\beta = 1$, to get a $\frac{4}{3}$-approximate solution. □

5. Conclusion

In Chapter 3, we have shown that, under reasonable assumptions, the weighted SCSS problem does not admit any constant approximation algorithm with FPT runtime when parameterizing by the number of Steiner vertices in the solution. This is in contrast to a result by Chitnis et al. [1], which says that the problem is fixed-parameter tractable when computing a 2-approximation if the parameter is the number of terminals instead. Therefore, in practical scenarios, we are most likely better off using their algorithm, unless the number of terminals far exceeds the number of Steiner vertices. Naturally, the question arises whether for our parameter we can find faster algorithms which compute solutions with a further relaxed requirement on the quality of solutions:

Open question 1. *Can we find a polynomial-time algorithm or a fixed-parameter algorithm where the parameter is the number of Steiner vertices, which for some function f, computes an $f(k)$-approximation to* SCSS*?*

In Chapter 4, we have shown that although the structures of solutions to TSP and BI-SCSS differ, the costs of their optima are comparable. This gives rise to multiple algorithms to approximate BI-SCSS, one for each algorithm that solves TSP. We have also shown that our analysis of the technique is optimal, but this does not mean that this is the best way to approximate BI-SCSS.

Open question 2. *Is there another technique to approximate* BI-SCSS*, which results in a better approximation factor without worsening the runtime, or better runtime without worsening the approximation factor?*

Chitnis et al. [1] have previously shown that BI-SCSS is FPT when parameterizing by the number of terminals, however the dependence on the parameter is doubly exponential. This makes an approximation factor of only $\frac{4}{3}$, in exchange for single exponential dependence on the parameter, again of interest in practice. It remains an open question whether we can do better than double exponential. However, if this dependence on the parameter for exact algorithms is optimal or if we cannot do better than a $\frac{4}{3}$-approximation without going above single exponential dependence, then these non-trivial results would seem to suggest that BI-SCSS is also quite unique from the theoretical standpoint, in both the context of approximation and parameterization.

Bibliography

[1] Rajesh Chitnis, Andreas Emil Feldmann, and Pasin Manurangsi. Parameterized Approximation Algorithms for Bidirected Steiner Network Problems. In Yossi Azar, Hannah Bast, and Grzegorz Herman, editors, *26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:16, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[2] Eran Halperin and Robert Krauthgamer. Polylogarithmic inapproximability. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '03, page 585–594, New York, NY, USA, 2003. Association for Computing Machinery.

[3] Lane A. Hemaspaandra. Sigact news complexity theory column 36. *SIGACT News*, 33(2):34–47, June 2002.

[4] Reasons to believe P!=NP — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/P_versus_NP_problem#Reasons_to_believe_P_%E2%89%A0_NP`.

[5] Jiong Guo, Rolf Niedermeier, and Ondřej Suchý. Parameterized complexity of arc-weighted directed steiner problems. In Yingfei Dong, Ding-Zhu Du, and Oscar Ibarra, editors, *Algorithms and Computation*, pages 544–553, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[6] Rajesh Chitnis, MohammadTaghi Hajiaghayi, and Guy Kortsarz. Fixed-parameter and approximation algorithms: A new look, 2013.

[7] Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

[8] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer New York, 1999.

[9] Miroslav Chlebík and Janka Chlebíková. The steiner tree problem on graphs: Inapproximability results. *Theoretical Computer Science*, 406(3):207 – 214, 2008. Algorithmic Aspects of Global Computing.

[10] Pavel Dvorák, Andreas Emil Feldmann, Dušan Knop, Tomáš Masarík, Tomáš Toufar, and Pavel Veselý. Parameterized Approximation Schemes for Steiner Trees with Small Number of Steiner Vertices. In Rolf Niedermeier and Brigitte Vallée, editors, *35th Symposium on Theoretical Aspects of Computer Science (STACS 2018)*, volume 96 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

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

[12] Reinhard Diestel. *Graph Theory (Graduate Texts in Mathematics)*. Springer, August 2005.

[13] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer International Publishing, 2015.

[14] David P. Williamson and David B. Shmoys. The design of approximation algorithms. 2011.

[15] Yijia Chen and Bingkai Lin. The constant inapproximability of the parameterized dominating set problem. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, October 2016.

[16] Bernhard Korte and Jens Vygen. *Combinatorial Optimization*. Springer-Verlag, 2006.

[17] J. Lee, M.J. Ablowitz, D.G. Crighton, S.H. Davis, E.J. Hinch, A. Iserles, J. Ockendon, and P.J. Olver. *A First Course in Combinatorial Optimization*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2004.

[18] G. Szekeres. Polyhedral decompositions of cubic graphs. *Bulletin of the Australian Mathematical Society*, 8(3):367–387, jun 1973.

[19] P.D. Seymour. Disjoint paths in graphs. *Discrete Mathematics*, 29(3):293–309, 1980.

[20] Jean Claude Bermond, Bill Jackson, and François Jaeger. Shortest coverings of graphs with cycles. *Journal of Combinatorial Theory, Series B*, 35(3):297–308, December 1983.

[21] Held–Karp algorithm — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Held%E2%80%93Karp%20algorithm&oldid=932316993`.