FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Zadání bakalářské práce

Název:	Férová řešení pro problém Target Set Selection
Student:	Bruno Kraus
Vedoucí:	RNDr. Dušan Knop, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Teoretická informatika
Katedra:	Katedra teoretické informatiky
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Vstupem pro problém Target Set Selection (TSS) je graf G s vrcholy ohodnocenými přirozenými čísly pomocí funkce f(v) pro vrchol v. Řešením je množina vrcholů S taková, že pro dynamický proces {\cal S}, nastavující S_0 na S a poté S_{i+1} na S_i sjednocené s množinou těch vrcholů v, které mají v S_i alespoň f(v) sousedů, platí S_n = V(G).
V literatuře je mnoho pojmů vystihujících férový aspekt řešení (motivováno pohledem kupříkladu z multiagentních systémů). Jako jeden z nejběžnějších uveďme řešení, která se snaží minimalizovat maximum vrcholů v S nalézajících se v okolí vrcholu (formálně tedy \min \max_{v \in V(G)} | N(v) \cap S |). Rádi bychom i na základě různých motivací z literatury definovali další aspekty férovosti pro problém TSS a zkoumali jejich vliv na (parametrizovanou) složitost tohoto problému.

Elektronicky schválil/a doc. Ing. Jan Janoušek, Ph.D. dne 17. února 2021 v Praze.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Fair Solutions to the Target Set Selection Problem

Bruno Kraus

Department of Theoretical Computer Science
Supervisor: RNDr. Dušan Knop, Ph.D.

May 13, 2021

Acknowledgements

I would like to express my deepest appreciation and thanks to RNDr. Dušan Knop, Ph.D. for all the invaluable help, guidance, and inspiration during our work on the thesis. He is an inspiring teacher and overall the best supervisor a student could ask for.

Declaration

In Prague on May 13, 2021 . . . .

Citation of this thesis

Kraus, Bruno. *Fair Solutions to the Target Set Selection Problem.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021.

Abstract

TARGET SET SELECTION is an NP-hard problem fundamental in the area of viral marketing. Given a social network, the TARGET SET SELECTION problem asks for the minimum size set of agents, whose influence ultimately affects everyone in the network. We propose the FAIR TARGET SET SELECTION problem, whose solution satisfies a *fair cost* measure rather than a certain solution size. The *fair cost* enforces a fair distribution of a target set by bounding the maximum number of friends of some agent in a solution. In this work, we study the parameterized complexity of FAIR TARGET SET SELECTION and show W[1]-hardness with respect to parameters *treewidth*, *feedback vertex set number*, and *treedepth* and W[2]-hardness for its natural parameter *fair cost*. On the more positive side, we show an FPT algorithm with respect to the combined parameter vertex cover number and fair cost. Our further results arrive when we consider special cases of the threshold functions since we prove NP-hardness when all thresholds are equal to a constant $c \geq 3$. Moreover, in the last chapter we give arguments why we believe that might be a polynomial time algorithm for FAIR TARGET SET SELECTION with majority thresholds on trees.

Keywords fair objective, target set selection, fair target set selection, viral marketing, social networks, parameterized computational complexity

Abstrakt

TARGET SET SELECTION je NP-těžký problém klíčový v oblasti virálního marketingu. Pro danou sociální síť se problém TARGET SET SELECTION zabývá minimální velikostí množiny agentů, jejichž působení nakonec ovlivní každého v dané sociální síti. V této práci představujeme FAIR TARGET SET SELECTION problém, jehož řešení uspokojuje *férovou cenu.* Férová cena vynucuje férové rozdělovení target setu omezením maximalního počtu agentů v okolí nějakého agenta v řešení. V této práci studujeme parametrizovanou složitost FAIR TARGET SET SELECTION a ukazujeme jeho W[1]-těžkost vůči parametrům *treewidth*, *feedback vertex set number* a *treedepth*, W[2]-těžkost pro jeho přirozený parametr *férovou cenu.* Jako pozitivní výsledek ukazujeme FPT algoritmus při parametrizeci kombinovaným paremetrem *vertex cover number* a *férová cena.* Naše další výsledky se dostavují při zohlednění speciálních případů thresholdové funkce. Dokazujeme NP-těžkost pro FAIR TARGET SET SELECTION a to i tehdy, když jsou všechny thresholdy rovny konstantě $c \geq 3$. Na závěr předkládáme argumenty, proč se domníváme, že FAIR TARGET SET SELECTION s majoritními tresholdy lze řešit na stromech v polynomiálním čase.

Klíčová slova férový úkol, výběr cílové množiny, férový výběr cílové množiny, virální marketing, sociální sítě, parametrizovaná výpočetní složitost

Contents

List of Figures

Introduction

Suppose you work in a company with a brand new innovative product and you want the public to adopt it. You are given a graph representing the social network. The vertices in the graph represent people in the social network and the edges between them represent interactions or friendships. Each person is assigned a threshold value representing the number of its friends, that will influence him into buying the product. Now, your task is to select the minimal set of people, who will be offered the free product, so that the cascade of influence and recommendations that they start, ultimately results in the product being adopted by everyone in the network.

The problem described above is informally introduced the NP-Hard Target Set Selection problem. This problem was firstly introduced in the context of viral marketing by Domingos and Richardson [1]. The threshold model we use in our work was proposed by Kempe, Kleinberg, and Tardos [2, 3] and it models the spread of influence, information, or disease in a social network. From the previous studies of the Target Set Selection problem, e.g., [4, 5, 6, 7], it is apparent that this problem is computationally very hard.

Now, let us go back to the company preparing the so-called viral marketing campaign. Suppose it was important for the company that the free samples your company is going to give away must be distributed fairly. Consider a distribution in which many friends of one person receive free samples. It could look as if that person made sure its friends got free samples and other people could feel as if the campaign was not fair. Moreover, you might consider the situation when many of your friends have received the free item but you have not. It might feel unfair to you that you have to pay for the product, which many of your friends received for free.

In our work, we shift from finding the optimal solutions in the sense of the target set size to solutions satisfying some fair aspects. We propose the Fair Target Set Selection problem whose solution satisfies the fair cost measure, inspired by Lin and Sahni [8], rather than a certain solution size. The fair cost bounds the maximum number of friends of each person that can get a free sample. This way, we can prevent the unfair situation described above and thus spread the free samples in the social network more fairly. The related work, where the objective function is changed to a fair measure can be found in, e.g, [8, 9].

We study the Fair Target Set Selection problem from the parameterized complexity point of view, and the results of our work show unexpected computational hardness similar to the hardness of the classical Target Set Selection. In Chapter 2, we show that Fair Target Set Selection is W[2]-hard with respect to its natural parameter fair cost. As we show in Chapter 3, our problem generalizes Fair Vertex Cover, proposed by Knop et. al [9], which implies W[1]-hardness for structural parameters treewidth, feedback vertex cover, and treedepth. In Chapter 4, we show the tractability when Fair Target Set Selection is parameterized with the combined parameter vertex cover number and fair cost. In Chapter 5, we show that the problem is NP-Hard even when all thresholds are equal to a constant $c \geq 3$. In the last chapter, we give arguments, why we think there might be a polynomial time algorithm on trees.

Since the Target Set Selection (TSS) is a heavily studied problem, let us point out some previous results regarding its study. Nichterlein et al. showed that TSS is W[2]-hard with respect to its natural parameter target set size on graphs with diameter 2. Dreyer and Roberts [10] showed that TSS is NP-hard even when all thresholds are constant and equal to $c \geq 3$. Chen [4] later proved NP-hardness of TSS even when all thresholds are constant and equal to two. Chen also showed showed that TSS can be solved on trees in linear time. Ben-Zwi et al. [5] showed that TSS is W[1]-hard with respect to the parameter treewidth and that TSS with input restricted to graphs of bounded treewidth ω can be solved in $\mathcal{O}(n^{\omega})$. Peleg [11] showed that TSS is NP-hard even with majority thresholds. The study of the TSS problem from the parameterized complexity perspective can be found in, e.g., [6, 7, 12].

Chapter 1

Problem statement and preliminaries

In this chapter, we introduce formal graph notations that are used throughout this work, define the Fair Target Set Selection problem, and lastly we introduce the parameterized complexity framework that our work builds on.

1.1 Basic graph notations

An *undirected graph* G is an ordered pair (V, E), where V is a nonempty finite set of elements called *vertices* and $E \subseteq \binom{V}{2}$ is a set of *edges*. An *edge* in an undirected graph G is a two-element subset of V.

A *directed* graph, on the contrary, would have an edge defined as an ordered pair of vertices from V. Unless we explicitly state otherwise, throughout this work we consider graphs to be undirected. It is worth pointing out that our graph definition omits loops and multiple edges.

Let $G = (V, E)$ be a graph. Two vertices $u, v \in V$ are *adjacent*, if there is an edge $\{u, v\} \in E$. The *neighbourhood* of a vertex $v \in V$ is the set of all vertices adjacent to it and we denote it as $N_G(v)$. Formally, $N_G(v) := \{u \in V \mid \{u, v\} \in E\}$. The *degree* of a vertex v in G, denoted $\deg_G(v)$, is defined as $\deg_G(v) := |N_G(v)|$. When the graph G is clear from the context, we omit the subscript. By n we denote the size of the vertex set V and by m we denote the number of edges in E.

A *tree* is a connected acyclic graph. It holds that there is exactly one path between every two vertices. A *rooted tree* $T = (V, E, r)$ is a tree with vertex set V, edge set E, and with a vertex $r \in V$ specified as the *root*. Let $u, v \in V$. If the vertex u is on a path from v to the root, then v is a *descendant* of u and u is an *ancestor* of v. Moreover, if $\{u, v\} \in E(V)$, then v is the *child* of u and u is the *parent* of v. A vertex v is called *leaf* when $deg(v) = 1$. The *level* of the vertex $v \in V$ is the length of the path between v and a root.

1.2 The model

Let us now define our model formally. We are given an undirected graph $G = (V, E)$ and a *threshold function* $\operatorname{thr}\colon V \to \mathbb{N}$.

The default state of every $v \in V$ is *inactive* and a vertex v becomes *active* once it has at least $\operatorname{thr}(v)$ active neighbours. The *activation process* occurs in subsequent *activation rounds*. Let A^i_S denote the active vertices after the i^{th} activation round and let $S \subseteq V$. The activation process is defined as follows:

$$A^0_S = S,$$
$$A^{i+1}_S = A^i_S \cup \Big\{ u \in V \setminus A^i_S \mid \Big| \{N(u) \cap A^i_S\} \Big| \geq \operatorname{thr}(v) \Big\}, \text{ and}$$

the process terminates when $A^\tau_S = A^{\tau+1}_S$ for some $\tau \in \mathbb{N}$. A set $S \subseteq V$ is called the *target set*, denoted *TS*, when $A^\tau_S = V$. We say that $S \subseteq V$ *activates* G, when S is a TS. The *fair cost* of $W \subseteq V$ is defined as $\max_{v \in V} |N(v) \cap W|$.

We now formulate the decision version of the Fair Target Set Selection problem.

Fair Target Set Selection (Fair TSS)	
Input:	An undirected graph $G = (V, E)$, a threshold function $\operatorname{thr}\colon V \to \mathbb{N}$, and a positive integer k.
Question:	Is there a target set $W \subseteq V$ of fair cost at most k?

Optimization version OPT of Fair TSS would ask for a target set with minimal fair cost k. Since k is a positive integer and we can upper bound it by n, solving OPT Fair TSS would introduce only a logarithmic slowdown as we can apply a binary search for the minimal fair cost k, where $(G, \operatorname{thr}, k)$ is an yes-instance.

Special cases of thresholds

Besides unrestricted threshold functions, we consider a few special cases of thresholds. *Degree dependant* threshold functions we work with are *majority* thresholds and *unanimous* thresholds. The majority threshold function assigns each vertex half of its degree and the unanimous threshold function assigns each vertex a threshold value equal to its degree. Another special case of thresholds is when each vertex in a graph has the same threshold value, *e.g.*, 2 or 3. In this work, we call such threshold functions *constant*. Let us define these special cases formally.

- *Unanimous threshold function:* $\operatorname{thr}(v) = \deg(v)$ for all $v \in V$.
- *Majority threshold function:* $\operatorname{thr}(v) = \lceil \deg(v)/2 \rceil$ for all $v \in V$.

- *Constant threshold function:* $\text{thr}(v) = c$ for all $v \in V$, where c is a positive integer.

1.3 Parameterized complexity

Parameterized complexity[13, 14, 15] is a framework in which we study the running times of algorithms not only based on the input size n, but also on the parameters describing various aspects of the input (e.g., size of the solution, treewidth). By studying problems in this manner, we can get a better insight into their computational complexity.

Definition 1. Parameterized problem *is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet. For an instance $(x, k) \in L$, k is called the parameter.*

Definition 2. *A parameterized problem L is* fixed parameter tractable*, if it is possible to correctly decide whether $(x, k) \in L$ in $f(k) \cdot |(x, k)|^{\mathcal{O}(1)}$ time, where $f : \mathbb{N} \to \mathbb{N}$ is a computable function.*

Definition 3. *Instances I and I' of a parameterized problem L are* equivalent *when $I \in L \iff I' \in L$.*

Definition 4 (Parameterized reduction, Cygan et al. [13])**.** *Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. A* parameterized reduction *from A to B is an algorithm that, given an instance (x, k) of A, outputs an instance (x', k') of B such that*

1. *(x, k) is a yes-instance of A if and only if (x', k') is a yes-instance of B,*
2. *$k' \leq g(k)$ for some computable function g, and*
3. *the running time is $f(k) \cdot |x|^{\mathcal{O}(1)}$ for some computable function f.*

Definition 5. *FPT is the complexity class containing all fixed parameter tractable problems.*

In addition to FPT, the complexity classes W[1] and W[2], for which it holds W[1] $\subseteq$ W[2], contain all problems in FPT and problems not believed to be in FPT.

For the following definitions, let $i \in \mathbb{N}$ be a number greater than zero.

Definition 6. *A parameterized problem L is* W[i]-hard*, if we can create a parameterized reduction to L from every problem in W[i].*

Definition 7. *A parameterized problem L is* W[i]-complete*, when it is W[i]-hard and $L \in$ W[i].*

The Clique problem, which asks whether there is a clique of size k in a graph, is W[1]-complete with respect to k. An example of a W[2]-complete problem with respect to the solution size is Dominating Set.

1.4 Parameters

In this section, we formally define the parameters used in our work.

Vertex cover. Let $G = (V, E)$ be a graph and $S \subseteq V$. We say that S is a *vertex cover* of G if for every edge $e \in E$ at least one of its endpoints is in S. Equivalently, it holds that the graph $G - S$ is an edgeless graph.

Definition 8. Vertex cover number $\mathrm{vc}(G)$ *is the size of a minimum cardinality vertex cover in graph G.*

Treewidth

To define the parameter treewidth, we need to define the tree decomposition first.

Definition 9 (Tree decomposition, [16])**.** *A* tree decomposition *of a graph $G = (V, E)$ is a pair (T, β). Here T is a rooted tree whose vertices we call* bags *and β is a function assigning each bag a subset of vertices of G such that the following holds:*

1. $\bigcup_{t \in V(T)} \beta(t) = V(G)$,
2. *for each edge $e \in E(G)$ there exists a bag t in T, such that $e \subseteq \beta(t)$, and*
3. *for each $v \in V(G)$ the set of bags $\{t \in T \mid v \in \beta(t)\}$ induces a connected subtree in T.*

The *width* of a tree decomposition (T, β) is $\max(\{|\beta(t)| - 1 \mid t \in T\})$. The *treewidth* of a graph G, denoted $\mathrm{tw}(G)$, is the minimal possible width of a tree decomposition of G.

Feedback vertex set

Definition 10. Feedback vertex set *of a graph G is the minimum cardinality subset of vertices in G, whose removal leaves G without cycles.*

Treedepth

Definition 11. *Transitive closure of a rooted forest $F = (V, E, r)$ is a graph $G = (V, E')$, where E' contains an edge between two vertices $u, v \in V$ if u is an ancestor of v in F.*

Definition 12 (Nešetřil and de Mendez [17])**.** Treedepth *of a graph G is a minimum height of a rooted forest whose transitive closure contains G.*

CHAPTER 2

First Observations

2.1 First lemmas

In the following lemma, we show that certain vertices do not increase the fair cost of a target set over k.

Lemma 13. *Let $G = (V, E)$ be a graph and suppose S is a target set of G with fair cost k. If there is a vertex $v \in V \setminus S$ with $deg(u) \leq k$ for all $u \in N(v)$, then $S \cup \{u\}$ is a target set for G with fair cost k.*

Proof. Let us assume for a contradiction that adding the vertex v into S increases the fair cost over k. In such case, the vertex v has to have a neighbour with degree at least $k + 1$, which is a contradiction to the assumption that $\forall u \in N(v)\colon \deg(u) \leq k$. $\square$

Now, we present a reduction rule that removes the vertices satisfying the precondition of Lemma 13, which do not increase a fair cost over k.

Definition 14. *We say that a reduction rule is* correct*, when the original instance (G, thr, k) and the new instance (G', thr', k) created by applying the reduction rule are equivalent.*

Reduction Rule 1. Let (G, thr, k) be an instance of FAIR TSS. If $v \in V(G)$ is a vertex such that $\forall u \in N(v)\colon \deg(u) < k$, delete v from G and lower the threshold values of its neighbours by one.

Lemma 15. *Reduction Rule 1 is correct.*

Proof. Let (G, thr, k) be a yes-instance and T be a target set of fair cost $\ell \leq k$ in G. We can create a target set $T' = T \cap V(G)'$ of G' having a fair cost at most ℓ. Now, T' activates G since we have lowered the thresholds in the neighbourhoods of vertices that are in T and not T', making the same effect as in T. Let (G', thr', k) be a yes-instance with target set T'. We can create a target set T in G by adding vertices missing G', since by Lemma 13 those vertices do not increase the fair cost over k. □

In the following lemma, we show what makes a vertex required to be in every target set.

Lemma 16. *Let $G = (V, E)$ be an undirected graph, k is the maximum fair cost, and $v \in V$ a vertex such that $\operatorname{thr}(v) > \deg(v)$. Vertex v must be in every target set of G.*

Proof. Let us assume towards a contradiction that v is not in a target set T. Then it must have been activated by its neighbours. However, if the vertex v has all neighbours active, it still cannot be activated since $\operatorname{thr}(v) > \deg(v)$. Which is a contradiction to T being a target set. □

The following lemma shows a simple yet useful statement that can be used for proving that some set is a target set.

Lemma 17. *Let $G = (V, E)$ be a graph and $T \subseteq V$ be a target set. If $S \subseteq V$ activates T, then S is also a target set.*

Proof. Once all vertices in T are active, they activate G since T is a target set. □

In the following lemma, we state the running time of an algorithm that checks whether a set is a target set. And in the proof of the lemma, we show the algorithm.

Lemma 18. *Given a graph $G = (V, E)$, we can check whether $T \subseteq V$ is a target set in $\mathcal{O}(n + m)$ time.*

Proof. We can check if T is a target set of G by Algorithm 1.

Algorithm 1 is correct since each vertex in T activates each of its neighbours. When a vertex $v \notin T$ gets activated by $\operatorname{thr}(v)$ of its neighbours, it also activates its neighbours.

The algorithm visits each vertex at most twice. Once in the queue and once at the end while checking if all vertices are activated. It also visits each edge $\{u, v\}$ at most twice. Once when u is activating its edges and once when v is activating its edges. This gives us the running time of the algorithm $\mathcal{O}(n + m)$. □

Note 19. You might notice that Algorithm 1 does not work in a different way than the activation process since vertices can be activated in different order. However, the final effect is the same since each vertex in the target set activates all its neighbours and once a vertex is activated, it also activates all its neighbours.

Algorithm 1: Target set check

```
Input  : Graph $G = (V, E)$, Set $T \subseteq V$
Output: True if $T$ is a target set, otherwise False
Queue Q ← $\emptyset$
for $v \in T$ do
    label $v$ as Active
    Q.push($v$)
while Q is not empty do
    v ← Q.front()
    for $u \in N(v)$ do
        $u$.thr ← $u$.thr - 1
        if $u$ is not Active then
            label $u$ as Active
            Q.push($u$)
    Q.pop()
for $v \in V$ do
    if $v$ is not Active then return False
return True
```

2.2 Diameter one graphs

Definition 20. *The* distance *between two vertices is the minimum length of a path between them.*

Definition 21. *The diameter of a graph $G = (V, E)$ is the maximum distance between two vertices $u, v \in V$.*

Definition 22. *A* Complete graph *on n vertices, denoted K_n, is a graph $G = (V, \binom{V}{2})$, where $|V| = n$.*

Note 23. Since in a complete graph every two distinct vertices are at distance 1 from each other, complete graphs are diameter one graphs. Moreover, since in diameter one graphs all vertices are at distance 1 from each other and thus they are adjacent. Diameter one graphs are complete graphs.

Lemma 24. *Let $G = (V, E)$ be a complete graph and T be a target set of G with fair cost k. Suppose that $u \in T$ and $v \notin T$ are vertices for which* $\mathrm{thr}(v) > \mathrm{thr}(u)$ *holds. Then $T' = T \setminus \{u\} \cup \{v\}$ is a target set with the same fair cost as T.*

Proof. Let i denote the number of the activation round when the vertex v gets activated by T. For all previous activation rounds $j \in \{0, \dots, i-1\}$ it holds $|A_T^j| \leq |A_{T'}^j|$. Since $|A_T^0| = |A_{T'}^0|$ and each vertex activated by T in round $\ell \in \{1, \dots, i-1\}$ gets also activated by T' in the same round at latest. The latter holds, since vertices in $V \setminus T$ either neighbour with the same number of vertices in T' as in T or are in T'. It also holds that the vertex u gets activated by T' at latest in the activation round $A^i_{T'}$. Since $|A^{i-1}_T| = |A^{i-1}_{T'}|$ holds and the vertex v has at least $\text{thr}(v)$ active neighbours in A^{i-1}_T, the vertex u with $\text{thr}(u) < \text{thr}(v)$ has at least $\text{thr}(u)$ active neighbours in $A^{i-1}_{T'}$. This leads to T' being a target set as T is a target set and $T \subseteq |A^i_{T'}|$, according to Lemma 17. □

Lemma 25. *If there exists a target set of size k in K_n, we can create it from k vertices with the highest thresholds in K_n.*

Proof. Let T be a target set with size k in K_n. By applying Lemma 24 exhaustively, we end up with a target set T' consisting of k vertices from K_n with the highest thresholds. □

Lemma 26. *Fair TSS is solvable on graphs with diameter one in polynomial time.*

Proof. Let $G = (V, E)$ be a complete graph and k the maximal fair price. If $|V| \leq k + 1$, then we can solve Fair TSS by adding all vertices from V into the target set. Therefore, let us assume $|V| > k + 1$. A target set of K_n can consist of at most k vertices, so that its fair cost is at most k. If there exists a target set in G, we can create a target set T by choosing k vertices with the highest thresholds in G according to Lemma 25. We can choose k vertices with the highest thresholds after sorting vertices in V by their thresholds with heap sort in $\mathcal{O}(n \cdot \log(n))$ time. We can then check whether T is a target set in $\mathcal{O}(n + m)$ time. Since the number of edges in a complete graph is $\mathcal{O}(n^2)$, the total running time of the algorithm solving Fair TSS on diameter one graphs is $\mathcal{O}(n^2)$. □

2.3 W[2]-hardness with respect to the fair cost

In this section, we show that Fair TSS is W[2]-hard with respect to its natural parameter fair cost. We show the hardness by showing a parameterized reduction from Hitting Set.

Hitting Set

Definition 27. *A* set family $\mathcal{F}$ *over an universe* $\mathcal{U}$ *is a set, where each element* $F \in \mathcal{F}$ *is a subset of* $\mathcal{U}$.

The Hitting Set problem is W[2]-hard with respect to the parameter k [18] and is defined as follows.

Hitting Set

Input: An universe $\mathcal{U} = \{u_1, \ldots, u_n\}$, a set family $\mathcal{F} = \{F_1, \ldots, F_m\}$ over an universe $\mathcal{U}$, and a positive integer k.

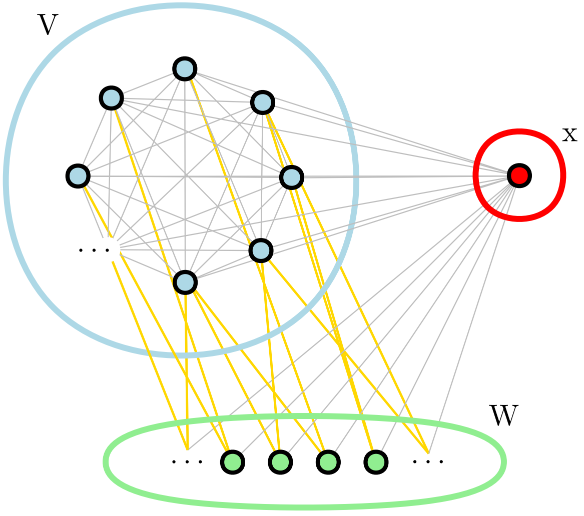
Question: Is there a hitting set $H \subseteq \mathcal{U}$ of size at most k so that $\forall F \in \mathcal{F} : H \cap F \neq \emptyset$?

Parameterized reduction

Let $(\mathcal{U}, \mathcal{F}, k)$ be an instance of Hitting Set. We create Fair TSS instance (G, thr, k) in the following way.

Graph G. Vertices of the graph G consist of *element-vertices* V, *subset-vertices* W, and a special vertex x. In V, there is an *element-vertex* v_u for each $u \in \mathcal{U}$. In W, there is a *subset-vertex* w_F for each $F \in \mathcal{F}$. We add an edge between an element-vertex v_u and a subset-vertex w_F if and only if $u \in F$. To finish the creation of G, we connect every two distinct vertices $u, v \in V \cup \{x\}$ and we connect x to all subset-vertices.

Figure 2.1: **Graph G.** the figure shows the graph G created as a reduction from the Hitting Set problem to Fair TSS. Element-vertices, denoted V, are colored in blue and they form a clique. Subset-vertices, denoted W, are green and they form an independent set. The special vertex x is colored in red, forms a clique with element-vertices and is connected to subset-vertices.



Thresholds. Thresholds in G are set in the following way:

- for the vertex x, it applies that $\text{thr}(x) = k + |W|$,
- each subset-vertex $w \in W$ has $\text{thr}(w) = 1$, and

- each element-vertex $v \in V$ has $\text{thr}(v) = k + 1 + |\{N(v) \cap W\}|$.

Correctness.

Theorem 28. *We claim that $(\mathcal{U}, \mathcal{F}, k)$ is a yes-instance of* Hitting Set *if and only if (G, thr, k) is a yes-instance of* Fair TSS.

Proof. $\Rightarrow$: Suppose that $(\mathcal{U}, \mathcal{F}, k)$ is a yes-instance and $S \subseteq \mathcal{U}$ is its hitting set of size k. Then a subset T of element-vertices in G defined as $T = \{v_u \in V | u \in S\}$ activates all subset-vertices W in the first activation round A_T^1. After A_T^1, all vertices in W are activated and k vertices in V are active. Thus, the vertex x gets activated in A_T^2. Finally, in round A_T^3, all remaining element-vertices get activated, making T a target set.
$\Leftarrow$: Now suppose (G, thr, k) is a yes-instance with the target set T of fair cost at most k.
Claim. We claim that the vertex x cannot be in T.
Proof. For a contradiction, assume that $x \in T$ and thus activates all subset-vertices in the first activation round A_T^1. In order for each element-vertex $v \notin T$ to get activated, it needs $\text{thr}(v) = k + 1 + |\{N(v) \cap W\}|$. Since it has all subset-vertex neighbours active and $x \in T$, it still needs k active element-vertices in T. The only way to activate them is to add k of them into T, which increases the fair cost of T over k. Thus, x cannot be in T. ■
Claim. We claim that a vertex $w \in W$ cannot be in T.
Proof. Assume that $w \in W$ is in T. Even if all vertices in W get activated by T, a vertex $v \in V$ can get activated only when there are k active vertices in V. Since element-vertex cannot get activated by its neighbours when v is not activated, k element-vertices must be in the target set. That would increase the fair cost over k and thus w cannot be in T. ■

Therefore, the target set T consists only of element-vertices. It is not hard to see that the vertex x and the vertices in V, which are not in T, must be activated after the vertices W. Since (G, thr, k) is a yes-instance, k element-vertices from T must activate all subset-vertices W in A_T^1. Making $(\mathcal{U}, \mathcal{F}, k)$ also a yes-instance. □

CHAPTER 3

Fair Vertex Cover reduction

In this chapter, we show how FAIR TSS generalizes the FAIR VERTEX COVER problem. Furthermore, we show the implications this relation has for W[1]-hardness with respect to the structural parameters *treewidth*, *treedepth*, and *feedback vertex set*.

3.1 Fair Vertex Cover

FAIR VERTEX COVER (FAIR VC)	
Input:	An undirected graph G and a positive integer k.
Question:	Is there a vertex cover of G of fair cost at most k?

Lemma 29. *Let (G, k) be an instance of FAIR VC, we can create an equivalent instance of FAIR TSS (G, thr, k), where the threshold function* thr *assigns every vertex in G its degree.*

Proof. Let $G = (V, E)$ be a graph, thr a function which assigns each vertex in V its degree, and k the fair cost.
$\Rightarrow$: Suppose the instance (G, k) is a yes-instance of FAIR VC. Since it is a yes-instance, it has a vertex cover $C \subseteq V$ of fair cost at most k. Let (G, thr, k) be an instance of FAIR TSS. Then C is a target set of G of fair cost at most k, since each vertex $v \in V$ either is in C and thus is active in A_C^0 or has all its $\deg(v)$ neighbours in C and is activated by them in the first activation round A_C^1.
$\Leftarrow$: For the other direction, assume that (G, thr, k) is a yes-instance of FAIR TSS. Then it has a target set $T \subseteq V$ with fair cost at most k. We claim that T is a vertex cover in G. For a contradiction, let $\{u, v\} \in E$ be an edge, where neither u nor v is in T. They cannot be activated in A_T^0, since they are not

in T. Thus, both of them must be activated by all their neighbours. Without loss of generality, let i denote the number of the activation round A_T^i, where u gets activated by $\deg(u)$ of its neighbours. Vertex v has to be active before A_T^i since it is adjacent to u. Now, since v cannot be activated by the vertex u before A_T^i, it must be in T, which contradicts u and v not being in T. □

3.2 Hardness results

Theorem 30. *Fair Target Set Selection is W[1]-hard with respect to the parameters treedepth and feedback vertex number.*

W[1]-hardness with respect to the combined parameter treedepth and fair vertex cover for Fair VC was shown by Knop et al. [9]. These hardness results hold for Fair TSS, since we can create a parameterized reduction from Fair VC to Fair TSS according to Lemma 29. In this section, we show the reduction from W[1]-complete problem ℓ-Multicolored clique to Fair VC, by which these results were proved.

Note 31. Note that the reduction from ℓ-Multicolored clique to Fair TSS can be done in the same manner as to Fair VC with additional setting of the thresholds of vertices to their degrees.

ℓ-Multicolored Clique (ℓ-MCC)

Input:	An undirected ℓ-partite graph $G = (V_1 \cup \cdots \cup V_\ell, E)$, where the partites are pairwise disjoint and each partite is an independent set.
Question:	Is there a clique of size ℓ in G?

The reduction

In this section, we show how an instance (G, ℓ) of ℓ-Multicolored Clique can be reduced to an equivalent instance (G', k) of Fair VC, where the fair cost $k = \max(|E(G)| - 1, 2 \cdot |V(G)|)$.

Introduction. The graph G' consists of three types of parts, which we refer to as gadgets. The types of gadgets in the reduction are *vertex selection gadget*, *edge selection gadget*, and *validation selection gadget.* In this section, we first define the gadgets composing the reduction and then we prove its correctness and the consequent hardness.

Notation. By color we refer to a number $c \in \{1, \ldots, \ell\}$. For each color pair (a, b) in this section, suppose that $a < b$. Let V_c denote a partite of color c in (G, ℓ). Let n denote the number of vertices in G and m denote the number of edges in G. Let $low\colon V(G) \to \{1, \ldots, n\}$ be a bijection and let

$high \colon V(G) \to \{n, \ldots, 2n-1\}$ be a bijection assigning each vertex $v \in V(G)$ a number $n - low(v)$.

Note 32. When constructing G', we can require some vertices to be in every solution. We require a vertex v to be in every solution by adding $k+1$ leaves with threshold 1 adjacent to v. Since $k+1$ neighbours of a vertex v cannot be in a vertex cover with fair cost at most k, the vertex v has to be in every solution of (G', ℓ). Since leaves have their degrees equal to one, they do not increase the fair cost over k.

Note 33. Let v be a vertex and k the maximum fair cost. By the budget of a vertex v, we denote the maximum number of vertices in $N(v)$, whose addition to the solution would not increase the fair cost over k. We can lower the budget of a vertex v, by connecting v to a vertex u, that is required to be in every solution as in Note 32, since such u has to be in every target set.

Vertex selection gadget. Firstly, for each color c, we construct the corresponding vertex selection gadget S_c in the following way. The vertex selection gadget S_c contains $|V_c|$ *choice vertices* and a special vertex called *guard* g_c. It holds that each vertex $v \in V_c$ is represented by one choice vertex in S_c and each choice vertex in S_c represents one vertex in V_c. The guard vertex g_c is adjacent to all choice vertices in S_c and is required to be in every solution. Furthermore, the budget of g_c is lowered to $|V_c| - 1$. Now, for each choice vertex $v \in S_c$, we add n new *connecting vertices* into S_c and connect them to v. Connecting vertices of a choice vertex $v \in S_c$ are divided into $low(u)$ low connecting vertices and $high(u)$ high connecting vertices, where $u \in V_c$ is the vertex that the choice vertex v represents.
Edge selection gadget. Secondly, for every color pair (a, b) in ℓ-MCC, we construct an edge selection gadget $E_{(a,b)}$ in a similar manner to the vertex selection gadget. The edge selection gadget $E_{(a,b)}$ consists of $|\{\{u, v\} \in E \mid u \in V_a \ \wedge v \in V_b\}|$ choice vertices and a guard vertex $g_{(a,b)}$. It holds that each choice vertex in $E_{(a,b)}$ represents one edge $\{u, v\}$ such that $u \in V_a \wedge v \in V_b$ and each edge $\{u, v\}$ such that $u \in V_a \wedge v \in V_b$ is represented by one choice vertex in $E_{(a,b)}$. The guard vertex $g_{(a,b)}$ is adjacent to all choice vertices in $E_{(a,b)}$, is required to be in every solution, and its budget is lowered to $|\{\{u, v\} \in E \mid u \in V_a \ \wedge v \in V_b\}| - 1$. Each choice vertex v in $E_{(a,b)}$ has $2n$ connecting vertices, which are divided into two groups of n *a-connecting vertices* and n *b-connecting vertices.* The a-connecting vertices are further divided into $low(u)$ low a-connecting vertices and $high(u)$ high a-connecting vertices, where $u \in V_a$ is the vertex adjacent to the edge represented by v. The b-connecting vertices are divided in the same manner, only the vertex u must be from V_b.
Validation gadget. For every color pair (a, b), we construct a validation gadget $V_{(a,b)}$. The validation gadget $V_{(a,b)}$ consists of 4 validation vertices. Particularly, we use two *a-validation vertices* to check the connections between S_a and $E_{(a,b)}$ and two *b-validation vertices* to check the connections

between S_b and $E_{(a,b)}$. All validation vertices are required to be in a solution. The two a-validation vertices are divided into the upper a-validation vertex and the lower a-validation vertex. The upper a-validation vertex is connected to high connecting vertices in S_a and to low a-connecting vertices in $E_{(a,b)}$. Similarly, the lower a-validation vertex is connected to low connecting vertices in S_a and high a-connecting vertices in $E_{(a,b)}$. The b-validation vertices are connected in the same manner to connecting vertices in S_b and b-connecting vertices in $E_{(a,b)}$.

Correctness of the reduction. Now, let us prove that the reduction is correct.

Theorem 34. *The presented reduction from ℓ-Multicolored clique to Fair TSS is correct.*

Proof. $\Rightarrow$: Suppose that (G, ℓ) is a yes-instance of ℓ-Multicolored Clique and $K \subseteq V_1, \times \cdots \times, V_\ell$ is its solution. We claim that the instance (G', k) is a yes-instance of Fair VC. We can create a vertex cover C of G' with fair cost at most k as follows:

- all guard and validation vertices, since they are required to be in every vertex cover,
- let $v \in S_c$ be a choice vertex representing a vertex $u \in V_c$ in K, then all connecting vertices of v and all choice vertices in S_c except v are in C, and
- let $v \in E_{(a,b)}$ be a choice vertex representing an edge between $u \in V_a$ and $w \in V_b$, where $u, w \in K$, then all connecting vertices of v and all choice vertices in $E_{(a,b)}$ except v are in C.

From the definition of C, it holds that each vertex is either in C or all its neighbours are. Thus, C is a vertex cover. To analyse the fair cost of C, we need to consider all types of vertices in G':

- each guard vertex g_c neighbours with at most $n - 1$ vertices in C,
- each guard vertex $g_{(a,b)}$ neighbours with at most $m - 1$ vertices in C,
- each choice vertex in S_a neighbours with at most $n + 1$ vertices in C,
- each choice vertex in $E_{(a,b)}$ neighbours with at most $2 \cdot n$ vertices in C,
- each connecting vertex has at most two neighbours in C, and
- finally, each required vertex by its definition has at most k neighbours in C.

Thus, it holds that the fair cost $k \leq \max(2 \cdot n, m-1)$.

$\Leftarrow$: To prove the other direction, suppose that (G', k) is a yes-instance of FAIR VC. Let C be a solution of (G', k). We will show that (G, ℓ) is also a yes-instance.

Claim. Let $E_{(a,b)}$ be an edge selection vertex. There must be $2n$ connection vertices of some choice vertex $v \in E_{(a,b)}$ in C.
Proof. Let h be the number of choice vertices in $E_{(a,b)}$. For a contradiction assume that there are not at least $2n$ connection vertices from $E_{(a,b)}$ in C. Thus, all h choice vertices from $E_{(a,b)}$ must be in C. However, this is a contradiction with C having the fair cost at most k, since the budget of v is lowered to $h-1$. ■

Claim. We claim that there is exactly one choice vertex from each S_a not in C.
Proof. Suppose for a contradiction, that all $|V_a|$ choice vertices from S_a are in C. Since the guard g_a has its budget lowered to $|V_a|-1$, this would increase the fair cost over k. Now, suppose that there are at least two vertices $u, v \in S_a$ not in C. Since C is a vertex cover, it must contain $2n$ connecting vertices of u and v. Let $V_{(a,b)}$ be a validation gadget and let v_1, v_2 be the pair of a-validation vertices in $V_{(a,b)}$. Addition of connecting vertices from u and v into C increases the total number of neighbours of v_1 and v_2 in C to $3n$. Since vertices v_1 and v_2 neighbour with n connecting vertices from $E_{(a,b)}$. Since each connection vertex has lowered budget to n, the maximum total number of neighbours of v_1 and v_2 is $2n$. Making it a contradiction to u and v not being in C. ■

Observe that in the same manner we can prove that in each $E_{(a,b)}$, there is one choice vertex not in C.
Claim. We claim that every pair of choice vertices from vertex selection gadgets that are not in C must represent adjacent vertices in G.
Proof. Towards a contradiction, let $u \in S_a$ and $v \in S_b$, both not in C, represent two vertices in G that are not adjacent. Thus, the choice vertex $w \in E_{(a,b)}$ not in C cannot represent the edge between vertices represented by u and v. Hence, without loss of generality, we assume that the choice vertex $u \in S_a$ does not represent the vertex incident to the edge represented by the choice vertex $w \in E_{(a,b)}$. Let v_1 and v_2 be upper and lower a-validation vertices from $V_{(a,b)}$ connected to the connection vertices of u and w. It holds that v_1 is connected to $low(u) + high(w) < n$ connection vertices in C, since $low(u) + high(w) = n$ if and only if $u = w$. Consequently, since

$$low(u) + high(u) + low(w) + high(w) = 2n \tag{3.1}$$

holds, v_2 is connected to $low(w) + high(u) > n$ connection vertices in C.

Which is a contradiction to the fair cost being at most k and thus u and v must represent vertices adjacent in G. ■

In our claims, we proved that each vertex selection gadget has exactly one choice vertex not in C and all such choice vertices are connected in G. Hence, such choice vertices form a clique of size ℓ in G, making (G, ℓ) a yes-instance of ℓ-MCC. □

Hardness. The total number of validation vertices is $\mathcal{O}(\ell^2)$, since there are 4 validation vertices for each color pair. When we remove the validation vertices from G', we end up with an acyclic graph. Thus, the feedback vertex set number of G' is $\mathcal{O}(\ell^2)$. Furthermore, once we remove the validation vertices, we also end up with trees of height $\mathcal{O}(1)$. Hence, the treedepth of G' is $\mathcal{O}(\ell^2)$. Finally, since the ℓ-Multicolored Clique is W[1]-hard with respect to ℓ [19], W[1]-hardness with respect to the parameters treedepth and feedback vertex set number for the Fair VC problem follows.

Chapter 4

Parameterization by the vertex cover number

In this chapter, we show that Fair TSS parameterized by the *vertex cover number* vc(G) and the *fair cost* k combined is FPT.

4.1 Motivation

In the previous chapter, we have shown W[1]-hardness for the parameter treewidth. Therefore, it is now reasonable to consider another structural parameter, which imposes stronger restrictions on the problem instance than treewidth. The weaker parameter in that manner could help us lower the problem's complexity and make solving it more tractable. One such parameter is the *vertex cover number.*

Lemma 35. *Let $G = (V, E)$ be a graph. It holds that $tw(G) \leq vc(G)$.*

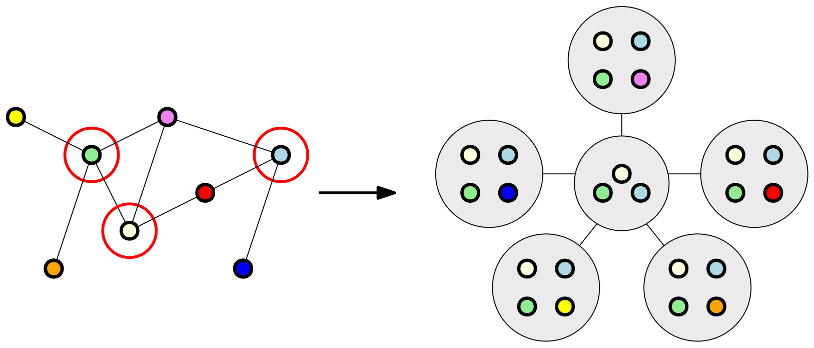
Given a vertex cover Z of G, we can show this relation. Let π denote the cardinality of Z, we can create a tree decomposition of width π in the following way. We create a bag b consisting of vertices in Z. Then we connect to b a bag b_v for each vertex $v \in V \setminus Z$ consisting of $Z \cup \{v\}$. This tree decomposition has the width equal to π. Since we can do this tree decomposition for any vertex cover of G, it holds that $tw(G) \leq vc(G)$.

It is also worth pointing out that a path has treewidth 1 and its vertex cover number depends on the length of the path.

4.2 FTP

In this section, we prove the fixed parameter tractability with respect to the combined parameter vc(G) and fair price k of Fair TSS in a similar

Figure 4.1: **The tree decomposition based on the vertex cover as in the proof of Lemma 35**. On the left side of the figure, there is a graph with vertices of different colors. A vertex cover Z of the graph is marked with red circles around its vertices. On the right side of the figure, there is a tree decomposition of the graph with width equal to $|Z|$.



manner to Niechterlein et al. [6] who gave an FTP algorithm with respect to $\mathrm{vc}(G)$ for TSS.

Let $G = (V, E)$ be a graph and let Z denote its vertex cover. Then $I = G \setminus Z$ is an independent set. Assuming that each vertex $v \in V$ has $\mathrm{thr}(v)$ at most $\deg(v)$, then Z is also a target set in G. Since each vertex $v \in V$ is either $v \in Z$ and thus in A_Z^0 or it holds, $N(v) \subseteq Z$ and thus v is in A_Z^1 since it has $\deg(v)$ active neighbours in A_Z^0. However, note that we do allow the threshold of a vertex $v \in V$ to be higher than v's degree and thus we cannot assume that Z is also a target set. Since a vertex $v \notin Z$ can have $\mathrm{thr}(v) > \deg(v)$ and then it cannot be activated by its neighbours. In addition, note that since v cannot be activated by its neighbours, it holds that v must be in every target set.

Note 36. Note that in this chapter we consider graphs to be connected, since to solve Fair TSS on a disconnected graph, we can solve Fair TSS on each its connected component.

Definition 37. *Vertices $u, v \in V$ are* twins *when $N(v) = N(u)$.*

Note 38. Note that Definition 37 prohibits twin vertices u and v from being adjacent. Twins defined in such a manner are called false twins. On the contrary, true twins would be defined as $u, v \in V$ with $N(v) \cup \{v\} = N(u) \cup \{u\}$.

Lemma 39. *Let $G = (V, E)$, $u, v \in V$ be twins with $thr(u) \leq thr(v)$, and $T \subseteq V$ be a target set such that $u \in T$ and $v \notin T$. Then $U = T \setminus \{u\} \cup \{v\}$ is a target set of the same fair cost.*

Proof. Let i denote the number of an activation round A_T^i in which v gets activated by T. For activation rounds A_T^j, where $0 \leq j < i$, it holds that $A_T^j \setminus \{u\} \subseteq A_U^j \setminus \{v\}$, since u and v have the same neighbours. When $j = i - 1$,

v has at least $\mathrm{thr}(v)$ active neighbours in A^j_T. Since u and v have the same neighbours, then at latest in A^j_U the vertex u has also at least $\mathrm{thr}(u)$ active neighbours, since $\mathrm{thr}(u) \leq \mathrm{thr}(v)$ holds. Now that U activates T, we can say that U is a target set according to Lemma 17.

Now, let us assume towards a contradiction that U does not have the same fair cost as T. Since U has a different fair cost than T, vertex v must neighbour with a vertex $x \in V$, such that $x \notin N(u)$ or vertex u must neighbour with a vertex $y \in V$, such that $y \notin N(v)$. However, both situations are a contradiction with u and v being twins. $\square$

Definition 40 (Critical Independent Set)**.** *Let $G = (V, E)$ be a graph. A set C of vertices is a* critical independent set *if every two distinct vertices from C are twins and C contains every vertex from V that is a twin with some $v \in C$.*

Lemma 41. *Let $G = (V, E)$ be a graph and Z be its vertex cover. Vertices in $I = V \setminus Z$ are contained in at most $2^{|Z|}$ critical independent sets.*

Proof. It holds that each vertex $v \in I$ has all its neighbours in Z and thus each critical independent set $C \subseteq I$ has a unique set of neighbours in Z. Since there are at most $2^{|Z|}$ possible neighbourhoods of vertices from I, there are at most $2^{|Z|}$ critical independent sets the vertices from I are contained in. $\square$

Lemma 42. *Let $G = (V, E)$ be a graph and Z be its vertex cover, and let $C \subseteq V$ be a critical independent set. There can be at most k vertices from C in any target set of fair cost k.*

Proof. Having more vertices than k from the critical set C in a target set would necessarily increase its fair cost to at least $k + 1$. Since all vertices in a critical independent set have the same neighbours. $\square$

Lemma 43. *Let $G = (V, E)$ be a graph, Z be its vertex cover, and $C \subseteq V \setminus Z$ be a critical independent set. Now, suppose that $H \subseteq C$ consists of k vertices with the highest thresholds in C. Given a target set $T \subseteq V$ with fair cost at most k, we can create a target set T' with fair cost at most k such that $C \cap T' \subseteq H$.*

Proof. After an exhaustive application of Lemma 39 on T, we end up with the target set T'. $\square$

Lemma 44. *Let $G = (V, E)$ be a graph, Z be its vertex cover, and let G have a target set with fair cost at most k. We can create a set $S \subseteq V$ of size at most $|Z| + 2^{|Z|} \cdot k$ containing a target set of a fair cost at most k.*

Proof. Let T be a target set of fair cost at most k in G and $I = V \setminus Z$. Let $H \subseteq V$ contain k vertices with the highest thresholds from each critical independent set in I. We claim that $|Z \cup H|$ is the set S.

The vertices in I are contained at most in $2^{|Z|}$ critical independent sets as in Lemma 41. Thus, it holds that $|Z \cup H| \leq |Z| + 2^{|Z|} \cdot k$.

After an exhaustive application of Lemma 43 on T, we end up with a target set T' of the same fair cost as T. Now, let $P \subseteq I$ be a critical independent set and M be a set consisting of k vertices with the highest thresholds in P. It holds that $P \cap T' \subseteq M \subseteq H \subseteq I$. Since this holds for every critical independent set $P \subseteq I$, $T' \subseteq Z \cup H$ follows. □

Theorem 45. *Fair TSS can be solved in $\mathcal{O}(2^{\pi+2^{\pi}\cdot k} \cdot (n+m))$ time, where π is the vertex cover number and k is the maximal fair cost.*

Proof. Let (G, thr, k) be an instance of Fair TSS. The algorithm first checks whether there is a vertex $v \in V$ with more than k neighbours with thresholds higher than their degree. If there is such a vertex v, the algorithm returns no. Since a vertex u with $\mathrm{thr}(u) > \deg(u)$ must be in any TS by Lemma 16. The vertex v with more than k such neighbours would increase the fair cost over k. This check can be done in $\mathcal{O}(n+m)$ time.

Then the algorithm finds a minimal vertex cover Z of G. This can be done by an FPT algorithm. Chen et al. [20] showed a polynomial space algorithm for Vertex Cover that runs in $\mathcal{O}(1.2738^h + h \cdot n)$ time, where h is the size of the VC.

Having computed the vertex cover Z, we can create the set $S \subseteq V$ containing a target set with fair cost at most k if it exists in G as in Lemma 44. The total number of vertices in S is at most $\pi + 2^{\pi} \cdot k$. Note that every vertex v with $\mathrm{thr}(v) \geq \deg(v)$ will be in this set. Since it is either in Z or it is in k vertices with the highest thresholds from some critical set. Let $I = V \subseteq Z$. The vertices in $S \cap I$ can be computed by picking k vertices from each of the 2^{π} critical sets. Since we pick from n vertices and we can check in linear time if v belongs to the critical independent set, we can compute vertices in $S \cap I$ in $\mathcal{O}(2^{\pi} \cdot n \cdot \pi)$ time.

The algorithm then checks each combination from the set of candidates whether it is a target set with fair cost at most k. Since one check can be done in $\mathcal{O}(n+m)$ time, it gives us the total running time of the algorithm $\mathcal{O}(2^{\pi+2^{\pi}\cdot k} \cdot (n+m))$. The algorithm is correct since it checks every combination in S. □

Chapter 5

NP hardness of Constant Fair TSS

In this chapter, we show NP-hardness for Fair TSS.

Theorem 46. *Fair TSS is NP-hard even when all thresholds are equal to a constant $c \geq 3$ and the fair cost $k = c$.*

We prove Theorem 46 by designing a polynomial reduction from NP-hard problem ℓ-Multicolored Clique. In what follows, let I be an instance of Fair TSS with all constant thresholds equal to $c \geq 3$ and fair cost $k = c$.

5.1 Constructions

Before we start constructing our reduction, we first show some building blocks that will be used by our gadgets. In this chapter, let thr be a constant threshold function assigning each vertex a constant c, and let $k = c$ be the maximum fair cost.

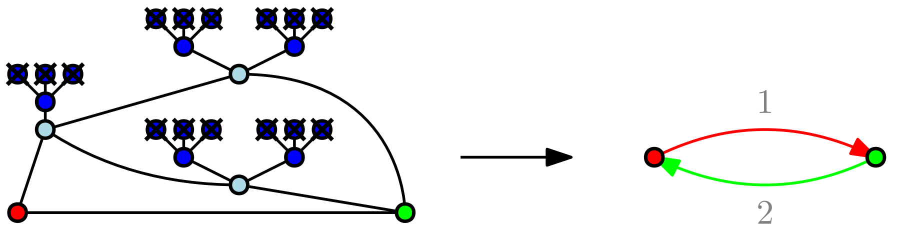
Note 47. Note that for a leaf v, it holds that $\text{thr}(v) \geq 3 > 1 = \deg(v)$. Thus, according to Lemma 16, the leaf v must be in every target set.

Lemma 48. *Let u be a vertex with c leaf neighbours, and let v be a vertex adjacent to u. The vertex v cannot be in any target set.*

Proof. For a contradiction, assume that v is in a target set T. Since the leaves have to be in the target set, u has at least $k + 1$ neighbours in T. Which is a contradiction to v being in a target set T. □

Prohibited vertex. Let v be a vertex in I. The vertex v can be prohibited from being in any target set by connecting v to a vertex u with c leaf neighbours as in Lemma 48. The threshold of each prohibited vertex is lowered by the number of its neighbours with at least c leaf neighbours.

Figure 5.1: **2-to-1 edge**. Let all thresholds be set to 3. On the left side of the figure, we see a green vertex connected to a red vertex by a 2-to-1 edge. Light blue vertices are prohibited from being in any target set since they are adjacent to the dark blue vertices with three leaves. The leaves are crossed in the figure since they must be in every target set as in Note 47. On the right side of the figure we introduce graphic notation of the 2-to-1 edge. A red arrow denotes activation of one neighbour of the green vertex, when red vertex is active. A green arrow denotes activation of two neighbours of the red vertex, when the green vertex is active.



Lemma 49. *Let H be a graph from Figure 5.1. Let u and v be the two vertices connected with 2-to-1 edge. We claim that the following holds.*

- *Active u and inactive u results in activation of two neighbours of v, while*
- *active v and inactive v results in only one active neighbour of u.*

Proof. To describe the graph H, it holds that the vertices u and v are adjacent. The vertex u is connected to two prohibited vertices with thresholds lowered to one. Those two vertices are connected to a prohibited vertex w with a threshold lowered to two. Finally, the vertex w is connected to v. Suppose a set S consisting of all required vertices in H. It holds that S consists of all leaves in H that prohibit some vertices from being in the target set.

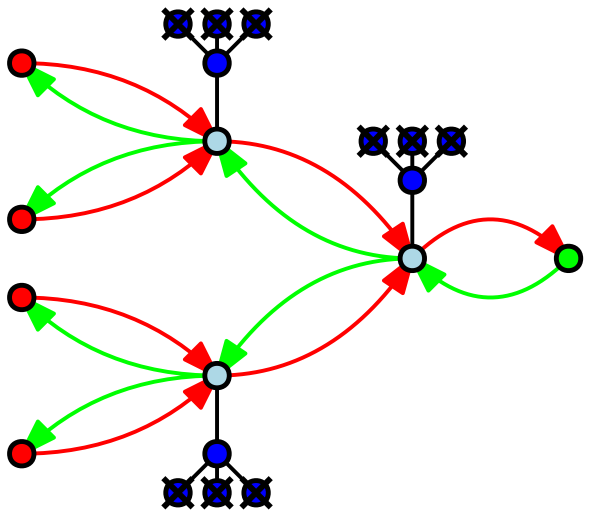
For $T = S \cup \{v\}$, it holds that T does not activate w, since u cannot get activated by T. Thus, the only active neighbour of u is v. Now, suppose that $T = S \cup \{u\}$, it holds that T activates two prohibited neighbours of u in A_T^1 since they have thresholds lowered to one. Those two prohibited neighbours then activate in A_T^2 the prohibited neighbour w of v, resulting in two active neighbours of v. □

2-to-1 edge. When two vertices u and v are connected by the construction described in the proof of Lemma 49 and illustrated in Figure 5.1, we say that they are connected by a 2-to-1 edge. In Lemma 49, we proved that active u results in two neighbours of v getting activated and active v results in one active neighbour of u.

Lemma 50. *Let H be a graph and let $x \in \mathbb{N}$ be a number. Let $u \in V(G)$, we claim that we can add vertices into H so that u can activate x vertices and yet itself is activated once all x vertices are active.*

Proof. Let each prohibited vertex in this proof have a threshold lowered to two. By the last layer, we denote the prohibited vertices not connected by 2-to-1 edge to other prohibited vertices. Let the vertex u be connected by 2-to-1 edges to the two prohibited vertices. Let those vertices in the last layer have two neighbours each. It holds that vertex u can activate 4 vertices while u gets activated once the 4 neighbours of the last layer are activated. (You can see the previously described construction on Figure 5.2.) By adding a layer, we mean connecting two new prohibited vertices by 2-to-1 edges to the prohibited vertices in the last layer. Observe that by adding layers or removing vertices in the last layer, the vertex u can activate any number of vertices and get activated once all those vertices are active. □

Figure 5.2: **Increased threshold construction**. Red and green double edge represents 2-to-1 edges as in Figure 5.1. The green vertex is connected by a chain of 2-to-1 edges to the red vertices, for which it holds that they are activated by the green vertex and that once all red vertices are activated they activate the green vertex. The light blue vertices represent prohibited vertices with thresholds lowered to two. The dark blue vertices prohibit the light blue vertices from being in a target set and the crossed vertices are leaves required to be in a target set as in Note 47.



Increased threshold. The proof of Lemma 50 gives us a construction by which a vertex u can activate x vertices and get activated once all x vertices are active. When we create such a construction, we say that the vertex u is connected by an increased threshold to x vertices. For the illustration of the construction, please refer to Figure 5.2. These newly defined constructions will be useful later in our reduction.

5.2 The reduction

In this section, we define a polynomial reduction from the ℓ-Multicolored Clique problem to Fair TSS.

Notation. Let (G, ℓ) be an instance of ℓ-MCC and (G', thr, k) an instance of Fair TSS. By a color we denote a number in $\{1, \dots, \ell\}$. Let V_c denote a partition of color c in G. For each color pair (a, b) in this section, suppose that $a < b$. Let us define the following two functions for vertices in $V(G)$.

Definition 51. *Let $low\colon V(G) \to \{1, \dots, |V(G)|\}$ be an arbitrary bijection that assigns each vertex in $V(G)$ a number from one to $|V(G)|$.*

Definition 52. *Given a low function, let*

$$high\colon V(G) \to \{|V(G)|, \dots, 2 \cdot |V(G)| - 1\}$$

be a bijection that assigns each vertex $v \in V(G)$ the number $2|V(G)| - low(v)$.

Similarly to the reduction in chapter 3, our reduction consists of *vertex selection*, *edge selection*, and *validation* gadgets. We create one vertex selection gadget for each of the ℓ colors and for each color pair we create one edge selection and one validation gadget.

Vertex selection gadget. The vertex selection gadget S_c of a color $c \in [\ell]$ consists of a special vertex g_c called *guard* and $|V_c|$ vertices called *choice vertices.* Each choice vertex in S_c represents one vertex in V_c and it holds that each vertex in V_c is represented by some choice vertex in S_c. The guard vertex g_c is adjacent to $k-1$ leaves and to all choice vertices in S_c. Each choice vertex in S_c has $2n$ (a, b)*-connecting vertices* for each color pair (a, b). Let (a, b) be a color pair, the (a, b)-connecting vertices of a choice vertex v are divided into $low(u)$ *low* (a, b)*-connecting vertices* and $high(u)$ *high* (a, b)*-connecting* vertices, where $u \in V_c$ is the vertex represented by v. The low (a, b)-connecting vertices are *enumerated* from one to $low(u)$ and the high (a, b)-connecting vertices are enumerated from $low(u) + 1$ to $2n$. A choice vertex v is connected to all its connecting vertices by the increased threshold construction (each vertex in the last layer of the increased threshold construction is connected to two connection vertices). Thus, a choice vertex v from S_c activates all its connecting vertices and gets activated when all its connection vertices are active.

Lemma 53. *Let T be a target set of fair cost at most k. The target set T contains at most one choice vertex v from S_c.*

Proof. For a contradiction, assume that there are two or more choice vertices from S_c in T. Since the guard vertex g_c is adjacent to all choice vertices in S_c

and to $k-1$ leaves which must be in T, the fair cost of T would be greater than k. $\square$

Edge selection gadget. The edge selection gadget $E_{(a,b)}$ of a color pair (a,b) consists of a guard vertex $g_{(a,b)}$ and of h choice vertices, where

$$h = \left|\left\{\{u,v\} \in E(G) \mid u \in V_a \wedge v \in V_b\right\}\right|.$$

It holds that each choice vertex in $E_{(a,b)}$ represents one edge $\{u,v\} \in E(G)$, where $u \in V_a \wedge v \in V_b$, and each such edge is represented by one choice vertex in $E_{(a,b)}$. The guard vertex $g_{(a,b)}$ is connected to all choice vertices in $E_{(a,b)}$ and to $k-1$ leaves. Each choice vertex v in $E_{(a,b)}$ has $4n$ connection vertices, which are split into $2n$ *a-connection vertices* and $2n$ *b-connection vertices.* Let $u \in V_a$ be a vertex incident to the edge represented by the choice vertex v. The connection vertices of v are divided into $low(u)$ *low a-connection vertices* and $high(u)$ *high a-connection vertices.* Then the low a-connection vertices are enumerated from one to $low(u)$ and the high a-connection vertices are enumerated from $low(u)+1$ to $2n$. The b-connection vertices are divided and enumerated in the same manner, except that u is the vertex from V_b incident to the edge represented by the choice vertex v. A choice vertex in $E_{(a,b)}$ is connected to all its connection vertices by an increased threshold construction (each vertex in the last level of the construction is connected to two connection vertices). Thus, active v activates all its connection vertices and v gets activated when all its connection vertices are active.

Lemma 54. *Let T be a target set of fair cost at most k. The target set T contains at most one choice vertex v from $E_{(a,b)}$.*

Proof. For a contradiction, let us assume that there are at least two vertices from $E_{(a,b)}$ in T. It holds that the guard vertex $g_{(a,b)}$ is adjacent to all choice vertices in $E_{(a,b)}$ and to $k-1$ leaves, which must be in T as in Note 47. Therefore, the fair cost of T would be at least $k+1$. $\square$

Validation gadget. The validation gadget $V_{(a,b)}$ for a color pair (a,b) consists of *a-validation group* and *b-validation group.* A validation group consists of the *higher* and the *lower validation row*, where both validation rows consist of $2n$ *validation vertices.* In each validation row, the validation vertices are enumerated from one to $2n$. Each validation vertex is prohibited from being in a target set and its threshold is lowered to one. The a-validation group is connected to (a,b)-connecting vertices of the vertex selection gadget S_a and to a-connecting vertices of the edge selection gadget $E_{(a,b)}$ in the following way:

- let w be a low (a,b)-connecting vertex in S_a, we connect it to the vertex in the lower validation row with the same enumeration,

- let x be a high a-connecting vertex in $E_{(a,b)}$, we connect it to the vertex in the lower validation row with the same enumeration,
- let y be a high (a,b)-connecting vertex in S_a, we connect it to the vertex in the higher validation row with the same enumeration, and
- let z be a low a-connecting vertex in $E_{(a,b)}$, we connect it to the vertex in the higher validation row with the same enumeration.

The b-validation group is connected to the (a,b)-connecting vertices of S_b and to the b-connecting vertices of $E_{(a,b)}$ in the same manner.

Note 55. Intuitively, you might notice that there is a perfect pairing between connecting vertices of a choice vertex $u \in S_a$, a-validation vertices, and a-connecting vertices of a choice vertex $v \in E_{(a,b)}$ if and only if, the vertex represented by u is incident to the edge represented by v in the graph G.

Lemma 56. *Let $v \in S_a$ be a choice vertex. The vertex v gets activated when all a-validation vertices are active.*

Proof. The vertex v can get activated by its connection vertices since it is connected to them by an increased threshold structure. It holds that each connection vertex of v is connected to one a-validation vertex and each a-validation vertex is connected to one connection vertex of v. Therefore, all connection vertices of v get activated when all a-validation vertices are active. □

Lemma 57. *Let $v \in E_{(a,b)}$ be a choice vertex. The vertex v gets activated when all a-validation and b-validation vertices are active.*

Proof. The vertex v can get activated by its connection vertices since it is connected to them by an increased threshold structure. Let $c \in \{a,b\}$. It holds that all c-connecting vertices of v get activated when all c-validation vertices are active. Since each connection vertex of v is connected to one c-validation vertex and each c-validation vertex is connected to one connection vertex of v. □

Lemma 58. *Let v be a choice vertex in S_a and let T be a target set of fair cost at most k. The vertex v gets activated only when $v \in T$ or all a-validation vertices are activated by T.*

Proof. Suppose $v \notin T$ and that some a-validation vertex does not get activated by T. The connection vertices and vertices in the increased threshold construction cannot activate v, since they are prohibited from being in T, and they get activated only once v is active or all a-validation vertices are active. Finally, the guard g_a alone cannot activate the vertex v since v has a threshold greater than one. □

Lemma 59. *Let v be a choice vertex in $E_{(a,b)}$ and let T be a target set of fair cost at most k. The vertex v gets activated only when $v \in T$ or all a-validation and b-validation vertices get activated by T.*

Proof. Towards a contradiction, assume that $v \notin T$ and some vertex in a-validation or b-validation vertices does not get activated by T. The a-connecting vertices of v, b-connecting vertices of v, and the vertices in the increased threshold construction between v cannot activate v, since they are prohibited from being in T and thus they get activated only when v is active or all a-connecting and b-connecting vertices are active. The guard vertex $g_{(a,b)}$ alone cannot activate the vertex v since v has threshold greater than one. □

Lemma 60. *Let $V_{(a,b)}$ be a validation gadget. Let $u \in E_{(a,b)}$ be a vertex representing an edge $\{x, y\} \in E(G)$, where $x \in V_a$ and $y \in V_b$. Let $v \in S_a$ be a choice vertex representing $z \in V_a$. Let $T \subseteq V(G')$ be a set with $u, v \in T$ and fair cost at most k. If $x = z$, the set T activates all vertices in a-validation group of $V_{(a,b)}$.*

Proof. It holds that all connection vertices of u and v get activated by T, since u and v are connected to their connecting vertices with the increased threshold construction. Let A_T^i denote an activation round in which all choice vertices of u and v are activated. Note that every validation vertex has its threshold lowered to one. Let w be a vertex in the upper a-validating row,

- if w is enumerated by a number at most $low(x)$, it gets activated by a low a-connecting vertex from $E_{(a,b)}$ with the same enumeration at the latest in A_T^{i+1}, and
- if w is enumerated by a number at least $low(z) + 1$, it gets activated by a high connecting vertex in S_a with the same enumeration at the latest in A_T^{i+1}.

Let w be a vertex in the lower a-validating row,

- if w is enumerated by a number at most $low(x)$, it gets activated by a low connecting vertex in S_a with the same enumeration at the latest in A_T^{i+1}, and
- if w is enumerated by a number at least $low(z) + 1$, it gets activated by a high a-connecting vertex from $E_{(a,b)}$ with the same enumeration at the latest in A_T^{i+1}.

Since all validation vertices are enumerated by a number from one to $2n$ and it holds, $x = y$. All a-validating vertices in $V_{(a,b)}$ get activated by the connecting vertices of u and v at the latest in $A_T^i + 1$. □

Lemma 61. *Let $T \subseteq V(G')$ be a set of fair cost k. Suppose that T does not contain vertices $v \in S_a$ and $u \in E_{(a,b)}$ for which it holds that v represents a vertex $x \in V_a$ and u represents an edge $\{y, z\} \in E(G)$, where $y \in V_a$, $z \in V_b$ and it holds that $x = y$. The set T cannot be a target set.*

Proof. Since each selection gadget can have at most one choice vertex in T, we can assume that T contains $v \in S_a$ and $u \in E(a, b)$ for which $x \neq y$.

Now, let A^i_T denote the activation round when all a-connecting vertices become active. It holds that the vertex $v \in S_a$ is the only choice vertex from S_a in T, as in Lemma 53, and the vertex u is the only choice vertex from $E_{(a,b)}$ in T, as in Lemma 54. Now, for other choice vertices from S_a and $E_{(a,b)}$ it holds that they cannot get activated before A^i_T according to Lemma 58 and Lemma 59. Since validation vertices are prohibited from being in T, it holds that the only connection vertices activated by T before an activation round i are the ones from u and v.

Towards a contradiction, let us assume that T activates all vertices in a-validation group of $V_{(a,b)}$. It must hold that the upper validation row of a-connecting vertices is connected to the $low(x)+high(x) > 2n$ active connection vertices of u and v, since $low(x)+high(y) = 2n$ applies if and only if $x = y$. Therefore, the lower row is connected to the other $low(y)+high(x) < 2n$ active connection vertices of u and v, since it holds,

$$low(x) + high(y) + low(y) + high(x) = 4n.$$

Since vertices in the lower a-validation row are connected to at most $2n-1$ active connecting vertices and one active connection vertex is connected to at most one vertex in the lower a-validation row. Thus, all vertices in a-validation group of $V_{(a,b)}$ cannot get activated and T cannot be a target set. □

Proof of Theorem 46. Let (G, ℓ) be an instance of ℓ-MCC and (G', thr, k) be an instance of Fair TSS created by the reduction from (G, ℓ).
$\Rightarrow$: Let (G, ℓ) be a yes instance and $K \subseteq V1, \times \cdots \times, V$ is its solution, i.e., a clique in G. We claim that (G', thr, k) is also a yes instance. Let $T \subseteq V(G)$ be created in the following way:

- add all required vertices from G' into T,
- for each color $c \in [\ell]$ add a choice vertex from S_c representing the vertex $v \in V_c \cap K$ into T, and
- for each color pair (a, b) add a choice vertex from $E_{(a,b)}$ representing an edge $\{u, v\} \in E(G)$, where $u \in V_a \wedge v \in V_b \wedge v, u \in K$.

It holds that T is a target set. Since all validation vertices get activated by the choice vertices from vertex selection and edge selection gadgets, according to Lemma 60. And the active validation vertices then activate the remaining inactive choice vertices as in Lemma 56 and Lemma 57. Choice vertices neighbour only with guard vertices and vertices in the increased threshold construction. Vertices in the increased threshold construction neighbour with at most one choice vertex in T and every guard neighbours with one choice vertex in T and $k-1$ required vertices. The required vertices not adjacent to a guard vertex also do not increase the fair cost over k, since there is at most k such required vertices adjacent to a vertex u and it holds that u is not adjacent to a choice vertex. Therefore, the fair cost of T is at most k.

$\Leftarrow$: To prove the other direction, suppose that (G', thr, k) is a yes-instance and T is the target set of fair cost at most k. For each validation gadget $V_{a,b}$, there must be $u, v \in T$ where $u \in S_a$ represents a vertex in G incident to an edge represented by $v \in E_{(a,b)}$ according to Lemma 61. Thus, the choice vertices in T form a clique of size ℓ in G, which makes (G, ℓ) also a yes-instance. $\square$

Chapter **6**

Towards an Algorithm for Trees

In this chapter, we restrict the input of Fair Target Set Selection to trees and we give arguments why we believe a polynomial time algorithm might exist in the majority thresholds setting.

6.1 First claims

Definition 62 (leaves)**.** *Let $G = (V, E)$ be a graph. We define a function* leaves: $V \to \mathbb{N}$*, which assigns each vertex $v \in V$ the number of its neighbours, which are leaves. Formally,* $\text{leaves}(v) = |\{u \in N(v) | \deg(u) = 1\}|$.

In the following lemma, we show that the number of leaf neighbours is a crucial attribute for deciding whether the vertex must be in the target set.

Lemma 63. *Let $G = (V, E)$ be a graph,* thr *a threshold function, and k a fair cost. Let $v \in V$ be a vertex for which it holds*

$$\text{thr}(v) - (\deg(v) - \text{leaves}(v)) > k. \tag{6.1}$$

The vertex v must be in every solution of a Fair TSS instance (G, thr, k).

Note 64. In the equation (6.1) $\deg(v) - \text{leaves}(v)$ represents the number of non-leaf neighbours of v. Assuming that all such neighbours are active and not in the target set, $\text{thr}(v) - (\deg(v) - \text{leaves}(v))$ expresses how many of the leaf neighbours of v we have to add into the target set in order to activate v.

Proof of Lemma 63. Assume towards a contradiction that T is a target set of fair cost at most k and it holds, $v \notin T$. Since $v \notin T$, v has to be activated by its neighbours. Even if we assume that all its non-leaf neighbours are not in T and get activated in the activation process. The vertex v still has to be activated by $\text{thr}(v) - (\deg(v) - \text{leaves}(v)) > k$ neighbours that are leaves, raising the fair cost of T over k. □

In what follows, let thr be the majority threshold function. The following lemma shows us that when a vertex in a rooted tree has all children active, it gets activated in the next round.

Lemma 65. *Let $G = (V, E)$ be a rooted tree,* thr *a majority threshold function, and $v \in V$ with* $\deg(v) \geq 2$*. Let $T \subseteq V$ and let A_T^x be a round of activation process which contains all children of v. We claim that $v \in A_T^{x+1}$.*

Proof of Lemma 65. For a contradiction, let us assume that $v \notin A_T^{x+1}$. Since v has at most one parent, it must have less than one son in A_T^x to not be in A_T^{x+1}. The vertex v having less than one son is a contradiction to its degree. □

Let G be a rooted tree. In the next lemma, we show that the set consisting of all leaves in G is a target set.

Lemma 66. *Let $G = (V, E)$ be a rooted tree and let T be a subset of V defined as $T = \{u \in V \mid \deg(u) = 1\}$. Then S is a target set of G.*

Proof. Let A_T^i contain all vertices of some level k in the rooted tree. It holds that A_T^{i+1} contains all vertices of level $k-1$, since each such vertex v is a leaf and thus in A_T^{i+1} or all its children are in A_T^i and thus $v \in A_T^{i+1}$ according to Lemma 65. Note that A_T^0 contains all vertices from the last level of G, since those vertices are leaves. It follows that T is the target set. □

Let G be a tree and let (G, thr, k) be an instance of Fair TSS with majority thresholds. The previous lemma gives us some intuition that if we manage to activate all vertices in the bottom levels of G, while satisfying the fair cost, we might solve the (G, thr, k).

6.2 Our approach

In this section, we show how we would approach designing the algorithm for Fair TSS with majority thresholds on trees.

By the intuition from Lemma 66. We think that it is worth trying to start by processing all vertices gradually starting from the bottom level of the tree. We know that there might be some vertices that must be in every target set, according to Lemma 63. The following definitions of the vertices in the tree will be helpful later. Let v be a vertex in a rooted tree, the labels are defined as follows.

Definition 67 (mts)**.** *Let k be the fair cost. The vertex v is labeled* mts*, when it cannot be activated by its neighbours without raising the fair cost over k, and thus it must be in every target set.*

Definition 68 (xts)**.** *Let k be the fair cost. Let v be a vertex with a parent vertex p. The vertex v is labeled* xts, *when it holds that v can be activated by its neighbours without raising the fair cost over k only when v's parent p is activated by its neighbours $N(p) \setminus \{v\}$.*

Definition 69 (nts)**.** *Let k be the fair cost. A vertex v is labeled nts when adding v into the target set would certainly increase its child's fair cost over k.*

Note 70. Note that a vertex can have multiple labels.

Note 71. Let I be an instance of Fair TSS and v a vertex in I. Note that v being labeled with both *mts* and *nts* would result in I being a no instance.

Let (G, thr, k) be an instance of Fair TSS with majority thresholds, where $G = (V, E)$ is a rooted tree and k is a positive integer. Let us consider the following partitions of vertices in G:

- V_1 - leaves,
- V_2 - vertices with children, who are only leaves,
- V_3 - vertices with children, who are only non-leaves, and
- V_4 - vertices having both leaves and non-leaves as children.

Note that each vertex in G belongs to exactly one of these sets.
We will be processing the vertices in the rooted tree from the bottom level upwards. Therefore, when the algorithm is processing some vertex, we can assume that all its children were processed and thus are active and labeled.
$\boldsymbol{V_2}$. Let us first consider the vertices in V_2. These vertices have children, who are leaves only. We further divide these vertices in the following cases:

- $V_2^> = \{v \in V_2 \mid \text{thr}(v) > k + 1\}$
- $V_2^= = \{v \in V_2 \mid \text{thr}(v) = k + 1\}$
- $V_2^< = \{v \in V_2 \mid \text{thr}(v) < k + 1\}$

Lemma 72. *Let v be a vertex in V_2. Then either v can be activated by its children, is labeled xts, or is labeled mts. Moreover, this can be decided in $\mathcal{O}(deg(v))$ time.*

Proof. Suppose that the vertex from V_2 does not have a parent. The graph G would be a rooted tree of height one and we could solve (G, thr, k) with a target set consisting of the root vertex only. Therefore, in what follows, we assume that vertices from V_2 do have a parent p.

Claim. We claim that $v \in V_2^<$ can be activated by its neighbours without raising the fair cost over k.

Proof. Let $v \in V_2^{<}$. It holds that the vertex v has one parent and at least one child. Thus, v has at least $\operatorname{thr}(v)$ children, since we have majority thresholds. Now, we can activate v by adding $\operatorname{thr}(v) \leq k$ its children into the target set. If we later add v's parent into the target set and it would increase the fair cost to $k+1$, we can simply remove one child of v from the target set. ■

If the vertex has one parent and all its children are leaves, it holds that $(\deg(v) - \operatorname{leaves}(v)) = 1$. This observation is useful in the proof of the following claim.

Claim. Let $v \in V_2^{>}$. The vertex v is labeled *mts*.
Proof. It holds that $\operatorname{thr}(v) > k+1$, since v is in $V_2^{>}$. Furthermore, the vertex v has one parent and all its children are leaves. Thus,

$$\operatorname{thr}(v) - (\deg(v) - \operatorname{leaves}(v)) = \operatorname{thr}(v) - 1 \geq k + 1 > k \qquad (6.2)$$

holds and by Lemma 63 the vertex v must be in every target set. ■

The last case of vertices in V_2 is $V_2^{=}$. Let v be a vertex in V_2. All its children are leaves and it holds, $\operatorname{thr}(v) = k + 1$.

Claim. Let $v \in V_2^{=}$ be a vertex with a parent p. The vertex v can be activated by its neighbours, only if its parent p is activated by $\operatorname{thr}(p)$ vertices in $N(p) \setminus \{v\}$ (i.e., v is labeled *xts*).
Proof. Since $\operatorname{thr}(v) = k+1$, at least one neighbour of v must be active and not in the target set before v gets activated by its neighbours. Otherwise, v must be in the target set. All children of v are leaves and assuming that they are not in the target set, they can get activated only after v is active. Therefore, v can be activated by its neighbours only when p is activated by $\operatorname{thr}(p)$ vertices in $N(v) \setminus \{v\}$. ■

Let $v \in V_2^{=}$. We label it *xts* and we add k children of v and v itself into the target set. This way, we can assume that v is active. If we later determine, that the parent of v can get activated by $\operatorname{thr}(p)$ vertices in $N(v) \setminus \{v\}$, we can remove v from the target set. Moreover, if the parent p gets added into the target set, we remove one of v's children from the target set to keep the fair cost at most k.

We have shown that for each vertex v in V_2 it can decide based on its degree and fact if v has a parent, whether v either

- must be in the target set (*mts*),
- must be in the target set unless its parent p can be activated by $\operatorname{thr}(p)$ vertices in $N(p) \setminus \{v\}$ or (*xts*), or
- can be activated by its children.

Which concludes our proof. □

***V_4*.** Let us consider vertices in V_4, whose children are leaves and non-leaves. Since we process vertices from the last level in the rooted tree upwards, we assume that it is already decided for the non-leaf children of a vertex in V_2, whether it is labeled *mts*, *xts*, or is activated by its children. Since $v \in V_4$ can have children labeled *xts* and *mts*, the following functions will be useful.

Definition 73. *Let $mTS\colon V \to \mathbb{N}$ be a function that assigns each vertex the number of its children labeled mts.*

Definition 74. *Let $xTS\colon V \to \mathbb{N}$ be a function that assigns each vertex the number of its children labeled xts.*

If $mTS(v) > k$ for some $v \in V(G)$, the instance (G, thr, k) of Fair TSS does not have a solution and the algorithm returns no, since the *mts* labeled vertices increase the fair cost over k. We can check this property at the beginning of the processing of a vertex, thus in what follows we can assume that $k - mTS(v) \geq 0$. Now, we further partition the vertices from V_4 into the following cases.

- $V_4^> = \{v \in V_4 \mid \text{thr}(v) - (\deg(v) - \text{leaves}(v)) > k - mTS(v)\}$
- $V_4^= = \{v \in V_4 \mid \text{thr}(v) - (\deg(v) - \text{leaves}(v)) = k - mTS(v)\}$
- $V_4^< = \{v \in V_4 \mid \text{thr}(v) - (\deg(v) - \text{leaves}(v)) < k - mTS(v)\}$

Firstly, we consider vertices from $V_4^>$. From the definition, for each vertex $v \in V_4^>$ the following equation applies.

$$\text{thr}(v) - (\deg(v) - \text{leaves}(v)) > k - mTS(v) \tag{6.3}$$

Observe that the equation (6.3) extends the inequality from Lemma 63 by subtracting $mTS(v)$ from the right-hand side. In Lemma 63, we assume that all non-leaf neighbours of v are active but not in the target set. In the equation (6.3) we add into consideration the vertices that are active, but have to be in the target set.

Lemma 75. *Let v be a vertex in $V_4^>$. Then either v is labeled mts or the graph is a no-instance. Moreover, this can be decided in $\mathcal{O}(deg(v))$ time.*

Proof. **Claim.** Let $v \in V_4^>$ with $xTS(v) = 0$. The vertex v is labeled *mts*, since it has to be in every target set.
Proof. Let T be a target set of fair cost at most k. Towards a contradiction, let us assume that $v \notin T$. The vertex v must be activated by its neighbours. Suppose that all its $(\deg(v) - \text{leaves}(v))$ non-leaf neighbours are active. It holds that we need to add $\text{thr} - (\deg(v) - \text{leaves}(v))$ leaves into T

to activate v. However, this would increase the fair cost of T over k, since $\text{thr}(v) - (\deg(v) - \text{leaves}(v)) + mTS(v) > k$. ■

Claim. Let $v \in V_4^>$ with $xTS(v) > 0$. The algorithm returns no.
Proof. Let u be an *xts* labeled child of v. The vertex u is in the target set. We can remove it from the target set if and only if v can be activated by its neighbours without u. Since it holds, $\text{thr}(v) - (\deg(v) - \text{leaves}(v)) > k - mTS(v)$, even if we assume that all its non-leaf children except *mts* labeled ones were active and not in the target set, activating v with its neighbours would result in a fair cost greater than k. Therefore, we cannot remove the *xts* labeled vertex from the target set and the fair cost is greater than k, since $mTS(v) + xTS(v) > mTS(v) = k$. ■

We have proved that the vertex in $V_4^>$ is either labeled *mts* or or there isn't a target set of fair cost at most k in the graph. Furthermore, for each vertex $v \in V_4^>$, we can decide which case it is based on the number of its *xts* labeled children. □

Secondly, let us analyse the vertices from $V_4^=$. They have mixed sons and for every vertex $v \in V_4^=$ the following equation holds.

$$\text{thr}(v) - (\deg(v) - \text{leaves}(v)) = k - mTS(v) \tag{6.4}$$

Lemma 76. *Let v be a vertex in $V_4^=$. Then either v is activated by its children, labeled xts, or the graph is a no-instance.*

Proof. We prove this lemma by proving the three following claims.

Claim. Let v be a vertex in $V_4^=$ without a parent and without *xts* labeled children. The vertex v is activated by its neighbours.
Proof. Since we can assume that all its non-leaf children are active and not in the target set, except the *mts* labeled vertices, we can add $\text{thr}(v) - (\deg(v) - \text{leaves}(v))$ leaf children of v into the target set. This activates v and the fair cost is equal to k since $\text{thr}(v) - (\deg(v) - \text{leaves}(v)) + mTS(v) = k$. ■

Claim. Let v be a vertex in $V_4^=$ with a parent p and without any child labeled *xts*. The vertex v is labeled *xts*.
Proof. Since the equation (6.4) holds for v, the vertex v can be activated by its neighbours, assuming that all its non-leaf neighbours, except from the *mts* labeled ones, are active and not in the target set. Since v does not have *xts* labeled children, it holds that v can be activated by its neighbours without raising the fair cost over k, when the parent p can be activated by $\text{thr}(p)$ vertices in $N(p) \setminus \{v\}$. Thus, the vertex v is labeled *xts*. ■

Let v be a vertex satisfying the proposition in the previous claim. The algorithm puts v into the target set. It also puts $\text{thr}(v) - (\deg(v) - \text{leaves}(v))$ of its leaf-children into the target set. Later, if p gets activated by $\text{thr}(p)$ vertices in $N(p) \setminus \{v\}$, we can remove v from the target set. In case that p ends up in the target set, we can remove one leaf-child of v from the target set to keep the fair cost of v at k.

Claim. Let v be a vertex in $V_4^=$ with $xTS(v) > 0$ labeled children. The algorithm returns no.
Proof. The vertex v can be activated by its neighbours, assuming that all its neighbours are active and the only neighbours of v in the target set are leaves and mts labeled vertices. However, v has a non-leaf neighbour in T, which is not labeled mts. Thus, the activation of v by its neighbours would increase the fair cost of T over k. We could remove an xts labeled child from the target set only if v could get activated by its own neighbours excluding u. However, the vertex v would necessarily need more leaf children in the target set, which would also raise the fair cost over k. Thus, since $mTS(v)+xTS(v) > mTS(v) = k$, the algorithm returns no. ■

We have showed, that a vertex v can be activated by its children, labeled xts, or the algorithm returns no. Moreover, we can decide which case holds for v based on the number of its children labeled xts and the fact if v has a parent. □

Now, let us consider $V_4^<$. We believe that similarly to the previous vertices in $V_4^>$, $V_4^=$, and V_2, we can decide based on its processed children, whether the vertex from $V_4^<$ is activated by its children, labeled xts, labeled mts, or the tree is a no-instance. In the following paragraphs we give arguments why that could be the case.

Let v be a vertex in $V_4^<$. The vertex v has mixed children and by its definition the following equation holds.

$$\text{thr}(v) - (\deg(v) - \text{leaves}(v)) < k - mTS(v) \tag{6.5}$$

Since we can only assume that $k - mTS(v) \geq 0$ it is possible that $\text{thr}(v) - (\deg(v) - \text{leaves}(v))$ is a negative number. This would mean that v has more non-leaf neighbours than $\text{thr}(v)$. Let us break down the vertices in $V_4^<$ into the two following cases.

1. $\{v \in V_4^< \mid \text{thr}(v) - (\deg(v) - \text{leaves}(v)) < 0\}$
2. $\{v \in V_4^< \mid \text{thr}(v) - (\deg(v) - \text{leaves}(v)) \geq 0\}$

Let v be a vertex from the second case, it holds that we can add $\text{thr}(v) - (\deg(v) - \text{leaves}(v))$ leaves into the target set, in order to activate v by its neighbour. To make sure that v is activated and has the fair cost at most k, the algorithm would have to consider whether v has a parent or not and the number of children labeled xts and mts.

Let v be a vertex from the first case. For v it holds that

$$\text{thr}(v) - (\deg(v) - \text{leaves}(v)) < 0 \leq k - mTS(v).$$

Since v has more than $\text{thr}(v)$ non-leaf neighbours, it holds that v gets activated by its children without the need to add its leaf-neighbours into the target set. The algorithm would still need to consider the number of vertices labeled xts and mts, to check whether v can be activated, while its children stay activated, without raising the fair cost over k.

***V_1*.** Each vertex in V_1 is a leaf, which has a parent either in V_2 or in V_4. If a vertex $v \in V_1$ has a parent in V_2, the algorithm decides, if v is in the target set or not while processing its parent. Similarly, when its parent is in $V_4^>$ or $V_4^=$. Let us assume that the algorithm could decide whether a vertex $u \in V_4$ can be activated without raising the fair cost over k. Since active u would result in activation of all its leaf children, it would mean that the algorithm decides whether vertices in V_1 are in the target set or not by processing vertices in V_2 and V_4.

***V_3*.** The last category of vertices, the algorithm would need to handle, is V_3. Vertices in V_3 have children, which are only non-leaf children. Since the algorithm processes vertices from the last level gradually upwards, we could assume when processing a vertex from V_3 for all its children, it was decided whether they are activated by its children, are labeled mts, or are labeled xts.

Now, let v be a vertex in V_3. Since we have majority thresholds and v has at most one parent and at least one child, it holds that v gets always activated by its children. Since, we make sure that all vertices are active after the algorithm processes them. However, we will need to check whether v has a fair cost at most k.

Conclusion. In this section, we have shown that when trying to solve Fair TSS with majority thresholds on trees, it is possible to decide for some groups (specifically V_2, $V_4^=$, and $V_4^>$) of vertices in a rooted tree whether they can be activated by their children or neighbours or they must be in the target set.

We believe it might be possible to extend such results to vertices in $V_4^=$. In that case we could create an algorithm, which decides for every vertex in V_1, V_2, and V_4 in polynomial time, whether it can be activated by their neighbours or must be in the target set in order to be active, and keep the fair cost at most k.

The last category of vertices, which would such algorithm need to handle, to solve Fair TSS with majority thresholds on trees, is V_3. With the assumption that all its children are active, these vertices would surely get activated. However, we would still need to check the fair cost and possibly need to remove some xts labeled vertices from the target set. An algorithm working in such way could run in polynomial or even linear time.

Conclusion

We have proposed the Fair Target Set Selection problem and initiated the study of its parameterized complexity. We have obtained W[1]-hardness with respect to the parameters treewidth, feedback vertex cover, and tree-depth. For its natural parameter fair cost, we have shown W[2]-hardness. On the positive side, we have shown an FPT algorithm for the combined parameter vertex cover number and fair cost. Furthermore, we have shown that Fair Target Set Selection is NP-hard even when all thresholds are equal to a constant $c \geq 3$. In the last chapter, we show arguments why we believe that Fair Target Set Selection with majority thresholds might be solvable on trees in polynomial time.

The open question that remains after our work is whether Fair TSS with majority thresholds or even unrestricted thresholds is solvable on trees in polynomial time. An interesting question that could extend our results is whether Fair TSS is NP-hard even when all thresholds are equal to two.

Bibliography

1. RICHARDSON, M.; DOMINGOS, P. Mining Knowledge-Sharing Sites for Viral Marketing. In: Edmonton, Alberta, Canada: Association for Computing Machinery, 2002, pp. 61–70. KDD '02. ISBN 158113567X. Available from DOI: `10.1145/775047.775057`.
2. KEMPE, D.; KLEINBERG, J.; TARDOS, É. Maximizing the Spread of Influence through a Social Network. In: *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Washington, D.C.: Association for Computing Machinery, 2003, pp. 137–146. KDD '03. ISBN 1581137370. Available from DOI: `10.1145/956750.956769`.
3. KEMPE, D.; KLEINBERG, J.; TARDOS, É. Influential Nodes in a Diffusion Model for Social Networks. In: 2005, vol. 3580, pp. 1127–1138. ISBN 978-3-540-27580-0. Available from DOI: `10.1007/11523468_91`.
4. CHEN, N. On the Approximability of Influence in Social Networks. In: *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. San Francisco, California: Society for Industrial and Applied Mathematics, 2008, pp. 1029–1037. SODA '08.
5. BEN-ZWI, O.; HERMELIN, D.; LOKSHTANOV, D.; NEWMAN, I. Treewidth governs the complexity of target set selection. *Discrete Optimization*. 2011, vol. 8, no. 1, pp. 87–96. ISSN 1572-5286. Available from DOI: `https://doi.org/10.1016/j.disopt.2010.09.007`. Parameterized Complexity of Discrete Optimization.
6. NICHTERLEIN, A.; NIEDERMEIER, R.; UHLMANN, J.; WELLER, M. On tractable cases of Target Set Selection. *Social Network Analysis and Mining*. 2012, vol. 3, no. 2, pp. 233–256. Available from DOI: `10.1007/s13278-012-0067-7`.

7. CHOPIN, M.; NICHTERLEIN, A.; NIEDERMEIER, R.; WELLER, M. Constant Thresholds Can Make Target Set Selection Tractable. *Theory of Computing Systems*. 2013, vol. 55, no. 1, pp. 61–83. Available from DOI: `10.1007/s00224-013-9499-3`.
8. LIN, L.; SAHNI, S. Fair Edge Deletion Problems. 1989, vol. 38, no. 5, pp. 756–761. ISSN 0018-9340. Available from DOI: `10.1109/12.24280`.
9. KNOP, D.; MASAŘÍK, T.; TOUFAR, T. Parameterized complexity of fair deletion problems II. *CoRR*. 2018, vol. abs/1803.06878. Available from arXiv: `1803.06878`.
10. DREYER, P. A.; ROBERTS, F. S. Irreversible k-threshold processes: Graph-theoretical threshold models of the spread of disease and of opinion. *Discrete Applied Mathematics*. 2009, vol. 157, no. 7, pp. 1615–1627. ISSN 0166-218X. Available from DOI: `https://doi.org/10.1016/j.dam.2008.09.012`.
11. PELEG, D. Local majorities, coalitions and monopolies in graphs: a review. *Theoretical Computer Science*. 2002, vol. 282, no. 2, pp. 231–257. ISSN 0304-3975. Available from DOI: `https://doi.org/10.1016/S0304-3975(01)00055-X`.
12. DVOŘÁK, P.; KNOP, D.; TOUFAR, T. Target Set Selection in Dense Graph Classes. 2018, vol. 123, 18:1–18:13. ISBN 978-3-95977-094-1. ISSN 1868-8969. Available from DOI: `10.4230/LIPIcs.ISAAC.2018.18`.
13. CYGAN, M.; FOMIN, F. V.; KOWALIK, Ł.; LOKSHTANOV, D.; MARX, D.; PILIPCZUK, M.; PILIPCZUK, M.; SAURABH, S. *Parameterized Algorithms*. 2015. Available from DOI: `10.1007/978-3-319-21275-3`.
14. DOWNEY, R. G.; FELLOWS, M. R. Parameterized Complexity. *Monographs in Computer Science*. 1999. Available from DOI: `10.1007/978-1-4612-0515-9`.
15. NIEDERMEIER, R. *Invitation to fixed-parameter algorithms*. Oxford University Press, 2008.
16. ROBERTSON, N.; SEYMOUR, P.D. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*. 1986, vol. 7, no. 3, pp. 309–322. ISSN 0196-6774. Available from DOI: `https://doi.org/10.1016/0196-6774(86)90023-4`.
17. NEŠETŘIL, J.; DE MENDEZ, P. O. *Sparsity Graphs, Structures, and Algorithms*. Springer Berlin, 2014.
18. CHEN, J.; CHOR, B.; FELLOWS, M.; HUANG, X.; JUEDES, D.; KANJ, I. A.; XIA, G. Tight lower bounds for certain parameterized NP-hard problems. *Proceedings. 19th IEEE Annual Conference on Computational Complexity*. 2004. Available from DOI: `10.1109/ccc.2004.1313826`.

19. PIETRZAK, K. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *J. Comput. Syst. Sci.* 2003, vol. 67, no. 4, pp. 757–771. Available from DOI: `10.1016/S0022-0000(03)00078-3`.

20. CHEN, J.; KANJ, I. A.; XIA, G. Improved upper bounds for vertex cover. *Theoretical Computer Science*. 2010, vol. 411, no. 40, pp. 3736–3756. ISSN 0304-3975. Available from DOI: `https://doi.org/10.1016/j.tcs.2010.06.026`.

Appendix A

Contents of enclosed CD

readme.txt	the file with contents description
thesis.pdf	the thesis text in PDF format