On Tractable Cases of Target Set Selection[⋆]

André Nichterlein, Rolf Niedermeier, Johannes Uhlmann, and Mathias Weller

Institut für Informatik, Friedrich-Schiller-Universität Jena,
Ernst-Abbe-Platz 2, D-07743 Jena, Germany
[andre.nichterlein,rolf.niedermeier,johannes.uhlmann,
mathias.weller]@uni-jena.de

Abstract. We study the NP-complete TARGET SET SELECTION (TSS) problem occurring in social network analysis. Complementing results on its approximability and extending results for its restriction to trees and bounded treewidth graphs, we classify the influence of the parameters "diameter", "cluster edge deletion number", "vertex cover number", and "feedback edge set number" of the underlying graph on the problem's complexity, revealing both tractable and intractable cases. For instance, even for diameter-two split graphs TSS remains very hard. TSS can be efficiently solved on graphs with small feedback edge set number and also turns out to be fixed-parameter tractable when parameterized by the vertex cover number, both results contrasting known parameterized intractability results for the parameter treewidth. While these tractability results are relevant for sparse networks, we also show efficient fixed-parameter algorithms for the parameter cluster edge deletion number, yielding tractability for certain dense networks.

1 Introduction

The NP-complete graph problem TARGET SET SELECTION (TSS) is defined as follows. Given an undirected graph $G = (V, E)$, a threshold function $\mathrm{thr} : V \to \mathbb{N}$, and an integer $k \geq 0$, is there a target set $S \subseteq V$ for G with $|S| \leq k$ activating all vertices in V, that is, for every $v \in V \setminus S$ eventually at least $\mathrm{thr}(v)$ of v's neighbors are activated? Note that *activation* is a dynamic process, where initially only the vertices in S are activated and the remaining vertices may become activated step by step during several rounds (see Section 2 for a formal definition). Roughly speaking, TSS offers a simple model to study the spread of influence, infection, or information in social networks; Kempe et al. [14] referred to it as influence maximization with a linear threshold model. In this work, we extend previous work [5,1] by studying the computational complexity of TSS for several special cases.

Domingos and Richardson [7] introduced TSS, studying it from the viewpoint of viral marketing and solving it heuristically. Next, Kempe et al. [14] formulated the problem using a threshold model (as we use it now), showed its NP-completeness, and presented a constant-factor approximation algorithm for a maximization variant. Next, Chen [5] showed the APX-hardness of a minimization variant, which holds even in case of some restricted threshold functions. He also provided a linear-time algorithm for trees. Most recently, Ben-Zwi et al. [1] generalized Chen's result for trees by showing

⋆ Supported by the DFG, research projects PABI, NI 369/7, and DARE, NI 369/11.

that TSS is polynomial-time solvable for graphs of constant treewidth; however, the degree of the polynomial of the running time depends on the treewidth (in other words, they showed that TSS is in the parameterized complexity class XP when parameterized by treewidth). They also proved that there is no $|V|^{o(\sqrt{w})}$ time algorithm (w denoting the treewidth) unless some unexpected complexity-theoretic collapse occurs. In particular, there is no hope for fixed-parameter tractability of TSS when parameterized by just treewidth.

Motivated by the mostly negative (or impractical) algorithmic results of Ben-Zwi et al. [1], we study further natural parameterizations of TSS. We obtain both hardness and tractability results. Since TSS resembles a dynamic variant of the well-known DOMINATING SET problem, it seems unsurprising that the NP-hardness of DOMINATING SET on graphs with diameter two [15] carries over to TSS. We show that this is indeed the case. In contrast, we also observe that TSS can be solved in linear time for diameter-one graphs (that is, cliques).

The main part of the paper considers three different parameterizations of TSS exploiting different structural graph parameters: First, we look at the "cluster edge deletion number" ξ of the input graph denoting the minimum number of edges whose deletion transforms the graph into a disjoint union of cliques. We show that TSS can be solved in $O(4^{\xi} \cdot |E| + |V|^3)$ time and provide polynomial problem kernel with respect to ξ and the maximum threshold $t_{\max}$.

Following the spirit of previous work considering graph layout and coloring problems [9,10], we study the parameter "vertex cover number" τ of the underlying graph. This parameter imposes a stronger restriction than treewidth does. Indeed, we prove that TSS is fixed-parameter tractable when parameterized by τ. In addition, for constant thresholds we show a problem kernel consisting of $O(2^{\tau})$ vertices and prove that there is little hope for a polynomial-size problem kernel in the general case.

Finally, as a third and (other than the previous parameters) easy-to-compute parameter also measuring tree-likeness, we study the feedback edge set number f (the minimum number of edges to delete to make a graph acyclic).[1] We develop polynomial-time data reduction rules that yield a linear-size problem kernel with respect to the parameter f. Moreover, we show that TSS can be solved in $4^f \cdot n^{O(1)}$ time, which again generalizes Chen's result [5] for trees.

Due to the lack of space, most proofs are deferred to a full version of the paper.

2 Preliminaries, Helpful Facts, and Small Diameter

Basic notation. Let $G = (V, E)$ be a graph and let $n := |V|$ and $m := |E|$ throughout this work. The *(open) neighborhood* of a vertex $v \in V$ in G is $N_G(v) := \{u : \{u, v\} \in E\}$ and the *degree* of v in G is $\deg_G(v) := |N_G(v)|$. The *closed neighborhood* of a vertex $v \in V$ in G is $N_G[v] := N_G(v) \cup \{v\}$. Moreover, for $V' \subseteq V$ let $N_G(V') := \bigcup_{v \in V'} N_G(v) \setminus V'$ and $N_G[V'] := \bigcup_{v \in V'} N_G[v]$. *Bypassing* a vertex $v \in V$ with $N(v) = \{u, w\}$ means to delete v from G and to insert the edge $\{u, w\}$.

[1] Graphs with small feedback edge set number are "almost trees"; such social networks occur in the context of e.g. sexually transmitted infections [19] and extremism propagation [12].

Note that, if u and w are already neighbors in G, then v must not be bypassed. We sometimes write $G - x$ as an abbreviation for the graph that results from G by deleting x, where x may be a vertex, edge, vertex set, or edge set. A *split graph* is a graph in which the vertices can be partitioned into a clique and an independent set. The class of split graphs is contained in the class of chordal graphs [4]. For a vertex set S, let $\mathcal{A}_G^i(S)$ denote the set of vertices of G that are *activated by S in the i'th round*, with $\mathcal{A}_G^0(S) := S$ and $\mathcal{A}_G^{j+1}(S) := \mathcal{A}_G^j(S) \cup \{v \in V : |N(v) \cap \mathcal{A}_G^j(S)| \geq \mathrm{thr}(v)\}$. For $S \subseteq V$, the uniquely determined positive integer r with $\mathcal{A}_G^{r-1}(S) \neq \mathcal{A}_G^r(S) = \mathcal{A}_G^{r+1}(S)$ is called the *number $r_G(S)$ of activation rounds*. Furthermore, we call $\mathcal{A}_G(S) := \mathcal{A}_G^{r_G(S)}(S)$ the set of vertices that are *activated by S*. If $\mathcal{A}_G(S) = V$, then S is called a *target set for G*. Thus, we arrive at the problem TARGET SET SELECTION: given an undirected graph $G = (V, E)$, a threshold function $\mathrm{thr} : V \to \mathbb{N}$ and an integer $k \geq 0$, is there a target set $S \subseteq V$ for G that contains at most k vertices?

Thresholds. Apart from arbitrary threshold functions, we consider two types of different threshold functions in this work.
Constant thresholds: All vertices have the same threshold $t_{\max}$. Since solving instances of TSS with $t_{\max} = 1$ is trivial (just select an arbitrary vertex in each connected component), we assume that all connected components of an input graph contain a vertex with threshold at least two.
Degree-dependent thresholds: The threshold of a vertex v depends on $\deg(v)$. In this context, note that, if $\mathrm{thr}(v) = \deg(v)$ for all vertices v, then TSS is identical to VERTEX COVER [5]. In this work, we particularly consider the "majority" threshold function, defined as $\mathrm{thr}(v) := \lceil \deg(v)/2 \rceil$.

In our work, sometimes thresholds of vertices are being decreased. However, thresholds can never be smaller than zero; further decreasing a threshold of zero has no effect.

Parameterized complexity. This is a two-dimensional framework for the analysis of computational complexity [8,11,17]. One dimension is the input size n, and the other one is the *parameter* (usually a positive integer). A problem is called *fixed-parameter tractable* (fpt) with respect to a parameter k if it can be solved in $f(k) \cdot n^{O(1)}$ time, where f is a computable function only depending on k. A core tool in the development of fixed-parameter algorithms is polynomial-time preprocessing by *data reduction* [2,13]. Here, the goal is to transform a given problem instance I with parameter k in polynomial time into an equivalent instance I' with parameter $k' \leq k$ such that the size of I' is upper-bounded by some function g only depending on k. If this is the case, we call I' a *kernel* of size $g(k)$. Usually, this is achieved by applying data reduction rules. We call a data reduction rule $\mathcal{R}$ *correct* if the new instance I' that results from applying $\mathcal{R}$ to I is a yes-instance if and only if I is a yes-instance. An instance is called *reduced* with respect to some data reduction rule if this rule has no further effect when applied to the instance. The whole process is called *kernelization*. Downey and Fellows [8] developed a parameterized theory of computational complexity to show fixed-parameter intractability by means of *parameterized reductions*. A parameterized reduction from a parameterized problem P to another parameterized problem P' is a function that, given an instance (x, k), computes in $f(k) \cdot n^{O(1)}$ time an instance (x', k') (with k' only depending on k) such that (x, k) is a yes-instance of P if and only if (x', k') is a yes-instance of P'. The basic complexity class for fixed-parameter intractability is called $W[1]$ and there is good reason to believe that $W[1]$-hard problems are not fpt [8,11,17]. Moreover, there is a whole hierarchy of classes $W[t]$, $t \geq 1$, where, intuitively, problems become harder with growing t.

Basic Facts. Based on the following observation, we present a data reduction rule to deal with vertices whose thresholds exceed their degrees.

Observation 1 *Let $G = (V, E)$ be a graph and let $v \in V$. If $\mathrm{thr}(v) > \deg(v)$, then v is contained in all target sets for G. If $\mathrm{thr}(v) = 0$, then v is not contained in any optimal target set for G.*

Reduction Rule 1. *Let $G = (V, E)$ and $v \in V$. If $\mathrm{thr}(v) > \deg(v)$, then delete v, decrease the threshold of all its neighbors by one and decrease k by one. If $\mathrm{thr}(v) = 0$, then delete v and reduce the thresholds of all its neighbors by one.*

Considering that changes propagate over each edge at most once, it is not hard to see that a graph can be reduced with respect to Reduction Rule 1 in $O(n + m)$ time. With Reduction Rule 1, we can define the graph that remains after an activation process.

Definition 1. *Let $(G = (V, E), \mathrm{thr}, k)$ be an instance of* TSS*, let $S \subseteq V$, and let $\mathrm{thr}_S : V \to \mathbb{N}$ with $\mathrm{thr}_S(v) := \infty$ for all $v \in S$ and $\mathrm{thr}_S(v) := \mathrm{thr}(v)$ for all $v \in V \setminus S$. Then we call the instance (G', thr', k') that results from exhaustively applying Reduction Rule 1 to (G, thr_S, k) the* reduced instance *of (G, thr, k) with respect to S.*

Observation 2 *Let $G = (V, E)$ be a graph reduced with respect to Reduction Rule 1. Then there is an optimal target set for G not containing vertices with threshold one.*

Small-Diameter Graphs. Since many real-world (social) networks have small diameter, it is natural to investigate the influence of the diameter on the complexity of TSS. If the diameter of a given graph G is one (that is, G is a clique), then a simple exchange argument allows us to observe that there is a minimum-size target set for G that contains the vertices with the highest thresholds. With Observation 1, we can sort the vertices by their thresholds in linear time using bucket sort and, once sorted, we can determine in linear time whether k vertices suffice to activate the clique. For graphs with diameter at least two, the close relationship to DOMINATING SET suggests that TSS is NP-hard [15]. We confirm this intuition by providing hardness results for several variants of TSS on graphs with constant diameter. Our hardness proofs use reductions from the $W[2]$-hard HITTING SET problem.

Theorem 1. TARGET SET SELECTION *is NP-hard and W[2]-hard for the parameter target set size k, even on*

1. *split graphs with diameter two,*
2. *bipartite graphs with diameter four and constant threshold two, and*
3. *bipartite graphs with diameter four and majority thresholds.*

3 Cluster Edge Deletion Number

Social networks may consist of almost cluster graphs. This motivates to study a parameterization measuring the distance from the input graph to a graph where all connected components are cliques, a so-called “cluster graph” [20]. One such distance measure is the size ξ of a minimum cluster edge deletion set, that is, a smallest set of edges whose removal results in a cluster graph.

For arbitrary thresholds, we show that TSS is fixed-parameter tractable with respect to ξ by providing an algorithm running in $O(4^{\xi} \cdot m + n^3)$ time. For restricted thresholds, we present a linear-time computable kernel whose number of vertices grows linearly with ξ when the thresholds are bounded by some constant. In fact, we can bound the number of vertices in this kernel by $2\xi(t_{\max} + 1)$, where $t_{\max}$ is the maximum threshold occurring in the input. Our elaborations highly depend on the notion of “critical cliques”: a clique K in a graph is a *critical clique* if all its vertices have the same closed neighborhood and K is maximal with respect to this property. Computing all critical cliques of a graph can be done in linear time [16].

Arbitrary Thresholds. To present a solving strategy for TSS in case of arbitrary thresholds, we first introduce a data reduction rule that shrinks clique-like substructures. The key to showing fixed-parameter tractability is the observation that, in a graph reduced with respect to this data reduction rule, there is an optimal target set consisting only of vertices that are incident to the ξ edges of a given optimal cluster edge deletion set. Since we can compute an optimal target set for a clique in linear time (see Section 2), we assume that none of the connected components of the input graph is a clique.

Now, we describe the data reduction rule that shrinks clique-like substructures. Consider a critical clique K in the input graph G and its neighbors $N_G(K)$ and note that $N_G(K)$ separates K from the rest of G. If activating all vertices in $N_G(K)$ is not enough to activate all vertices of K, then every target set of G has to contain some vertices of K. Without loss of generality, we can assume those vertices to have the highest threshold among all vertices of K. These considerations lead to the following.

Reduction Rule 2. *Let $I := (G, \mathrm{thr}, k)$ be an instance of* TSS *and let K be a critical clique in G. Moreover, let (G', thr', k') be the reduced instance of $(G[N_G[K]], \mathrm{thr}, k)$ with respect to $N_G(K)$ and let S denote an optimal target set for (G', thr') (see Definition 1). Then, reduce I with respect to S.*

Herein G' is a clique and, hence, an optimal target set S for (G', thr') can be computed in linear time (see Section 2).

Having reduced the input with respect to Reduction Rule 2, we claim that we can limit our considerations to a small subset of vertices of the remaining graph. To see this, consider a cluster edge deletion set E' of a graph G and let $V(E')$ denote the vertices incident to the edges in E' (we call the vertices in $V(E')$ *affected* by E').

Lemma 1. *Let $I := (G, \mathrm{thr}, k)$ denote a yes-instance reduced with respect to Reduction Rule 2 and let E' denote an optimal cluster edge deletion set of G. Then, there exists an optimal target set S for I with $S \subseteq V(E')$.*

By Lemma 1, an optimal target set can be found by systematically checking every subset of the at most 2ξ vertices affected by an optimal cluster edge deletion set, leading directly to the following theorem.

Theorem 2. TARGET SET SELECTION *can be solved in* $O(4^\xi \cdot m + n^3)$ *time, where* ξ *denotes the cluster edge deletion number of* G.

Restricted Thresholds. In the following, we present a data reduction shrinking large critical cliques until their size can be bounded by the maximum threshold $t_{\max}$. Thereby, we obtain a problem kernel with $2\xi(t_{\max}+1)$ vertices, which is linear in ξ if the thresholds of the input are bounded by a constant.

Consider a critical clique K containing at least $t_{\max}+1$ vertices. Informally speaking, it is sufficient to keep the $t_{\max}$ vertices of K with the smallest thresholds, since if these are activated, then all vertices in $N_G[K]$ are activated.

Reduction Rule 3. *Let* (G, thr, k) *be an instance of* TSS *and let* $t_{\max}$ *denote the maximum threshold of this instance. Furthermore, let* K *be a critical clique in* G *with* $|K| > t_{\max}$ *and let* K^{high} *denote the* $|K| - t_{\max}$ *vertices with highest thresholds of* K. *Then, delete the vertices in* K^{high} *from* G.

In a reduced instance, the number of critical cliques is upper-bounded by 2ξ and each critical clique contains at most $t_{\max}$ vertices. Thus, the number of vertices in a reduced graph is at most $2\xi(t_{\max}+1)$.

Theorem 3. TARGET SET SELECTION *admits a problem kernel with* $2\xi(t_{\max}+1)$ *vertices. The kernelization runs in* $linear$ *time.*

4 Vertex Cover Number

The vertex cover number of a graph G denotes the cardinality of an optimal vertex cover of G (that is, a minimum-size vertex set such that every edge has at least one endpoint in this set). Since deleting all vertices in a vertex cover results in a graph without edges, a vertex cover of a graph G is a feedback vertex set for G as well. Moreover, it is a well-known fact that the feedback vertex set number is an upper bound on the treewidth. Hence, the vertex cover number is an upper bound on the treewidth, too. Ben-Zwi et al. [1] have shown that TSS is W[1]-hard for the combined parameter treewidth and target set size. Indeed, the given reduction shows that TSS is W[1]-hard even for the combined parameter feedback vertex set number and target set size. These hardness results motivate the study of TSS parameterized by parameters larger than the feedback vertex set number as, for example, the vertex cover number [9,10].

Arbitrary thresholds. If the input graph G is reduced with respect to Reduction Rule 1, then the threshold of each vertex is bounded by its degree. Thus, it can be activated by activating all its neighbors. This implies that a vertex cover of G is also a target set for G and, hence, the target set size k is at most the vertex cover number τ of G. In this sense, the vertex cover number τ is a “weaker” parameter than the target set size k and allows for fixed-parameter algorithms with respect to parameter τ.

Let $G = (V, E)$ denote a graph and let Z denote a vertex cover of G. Then, $I := V \setminus Z$ is an independent set. The key to showing fixed-parameter tractability of TSS with respect to τ is to bound the number of vertices in I that are candidates for an optimal target set. To this end, vertices in I with an identical neighborhood are of particular interest. In this section, τ denotes the vertex cover number of G.

Definition 2. *Two pairwise non-adjacent vertices with the same open neighborhood are called* twins*. A set V' of vertices is called* critical independent set *if each two distinct vertices from V' are twins and V' is maximal with respect to this property.*

The vertices of I are contained in at most $2^{|Z|}$ critical independent sets (one for each subset of Z). The next observation allows us to focus on a restricted number of vertices for each critical independent set when looking for an optimal target set.

Observation 3 *Let $G = (V, E)$ denote a graph and let $u, w \in V$ denote two twins of G with $\mathrm{thr}(u) \geq \mathrm{thr}(w)$. In addition, let S denote a target set with $w \in S$ and $u \notin S$. Then, $S' := S \setminus \{w\} \cup \{u\}$ is a target set for G.*

In the following, we assume that the vertices of the input graph are ordered decreasingly by their thresholds. By Observation 3, we can directly conclude that there is an optimal target set that contains for each critical independent set I only vertices from the set of the $k \leq \tau$ vertices with highest threshold of I. Moreover, we can bound the number of critical independent sets by a function of τ, leading to fixed-parameter tractability of TSS with respect to τ.

Theorem 4. TARGET SET SELECTION *can be solved in $O(2^{(2^\tau+1)\cdot\tau} \cdot m)$ time, where τ denotes the vertex cover number of G.*

Proof. (Sketch) Let $(G = (V, E), \mathrm{thr}, k)$ denote the input instance for TSS. In the following, we assume that G is reduced with respect to Reduction Rule 1.

The algorithm works as follows. In a first phase, it computes an optimal vertex cover Z of G (this can be done in $O(1.3^\tau + \tau n)$ time [17]). If $k \geq |Z|$, then the algorithm returns Z. In a second phase it computes a set C of at most $(2^\tau + 1) \cdot \tau$ vertices for which there exists an optimal target set S with $S \subseteq C$. Then, in a third phase, the algorithm determines an optimal target set by systematically checking every subset of C. The set C in the second phase is computed as follows: For each critical independent set, C contains the k vertices with highest threshold (for a critical independent set of cardinality at most k all its vertices are in C). In the third phase, the algorithm systematically checks every size-k subset of C for being a target set. If no such target set is found, then it rejects the instance.

Next, we show that the presented algorithm finds a target set of size at most k for G, if it exists. For the correctness of the first phase note that, since G is reduced with respect to Reduction Rule 1, any vertex cover of G is a target set for G, as well. For the correctness of the other phases note that, by iteratively applying Observation 3, we can directly conclude that there exists an optimal target set S with $S \subseteq C$. Thus, the correctness of the algorithm follows by the fact that it checks every size-k subset of C.

We omit the running time analysis. □

Restricted thresholds. Next, we show that TSS admits a problem kernel with $O(2^\tau \cdot t_{\max})$ vertices. Moreover, we show that TSS parameterized by the vertex cover number presumably does not admit a polynomial problem kernel even for majority thresholds.

First, we present a problem kernel for the combined parameter vertex cover number τ and maximum threshold $t_{\max}$. To bound the number of vertices in a reduced instance, we need (besides Reduction Rule 1) one further rule that reduces large critical independent sets. For every critical independent set of size at least $t_{\max} + 1$, this rule removes all but the $t_{\max}$ vertices with "smallest thresholds".

Reduction Rule 4. *Let (G, thr, k) denote a* TSS*-instance reduced with respect to Reduction Rule 1 and let I denote a critical independent set of G with $|I| > t_{\max} + 1$. Then delete the $|I| - (t_{\max} + 1)$ highest-threshold vertices of I.*

Theorem 5. TARGET SET SELECTION *admits a problem kernel with at most $O(2^\tau \cdot t_{\max})$ vertices, where $t_{\max}$ denotes the maximum threshold and τ denotes the vertex cover number of G. The kernelization runs in $O(n \cdot (n + m))$ time.*

Dom et al. [6] showed that HITTING SET does not admit a problem kernel of size $(|U| + k')^{O(1)}$ unless an unexpected complexity-theoretic collapse occurs. Here, k' denotes the solution size and $|U|$ denotes the size of the universe. Bodlaender et al. [3] introduced a refined concept of parameterized reduction (called polynomial time and parameter transformation) that allows to transfer such hardness results to new problems. By a reduction from HITTING SET to TSS, we can show that, under reasonable complexity-theoretic assumptions, TSS does not admit a problem kernel of size $(\tau + k)^{O(1)}$.

Theorem 6. TARGET SET SELECTION *on bipartite graphs with majority thresholds does not admit a problem kernel of size $(\tau + k)^{O(1)}$ unless coNP $\subseteq$ NP/poly, where k denotes the target set size and τ denotes the vertex cover number of G.*

5 Feedback Edge Set Number

Another structural parameter of a graph G that is lower bounded by its treewidth is the size f of a smallest feedback edge set of G, that is, a smallest set of edges of G whose deletion makes G acyclic. Recall from Section 4 that TSS is $W[1]$-hard with respect to the parameter feedback vertex set. Some real-world social networks are tree-like (for example sexual networks [19], see also [5]) or scale-free (see [12]). This often corresponds to small feedback edge sets and motivates parameterizing TSS with f. Like the vertex cover number τ, the feedback edge set number f is a "weaker" parameter than treewidth, raising hope for faster algorithms for TSS in cases where f is small. A clear advantage over treewidth (and tree decompositions) is that an optimal feedback edge set can be determined efficiently by computing a spanning tree.

First, we show a problem kernel of size $O(f)$ for TSS based on two data reduction rules. Second, we present an algorithm that solves TSS in $4^f \cdot n^{O(1)}$ time.

Kernelization for TSS. In this paragraph, we present two data reduction rules for TSS that can be applied exhaustively in linear time and leave a problem kernel of size $O(f)$. Reduction Rule 5 removes vertices v with $\mathrm{thr}(v) = \deg(v) = 1$ from G. Note that Reduction Rule 1 (see Section 2) removes all other degree-one vertices from G.

Reduction Rule 5. *Let $(G = (V, E), \mathrm{thr}, k)$ be an instance for* TSS *reduced with respect to Reduction Rule 1 and let $v \in V$ with $\mathrm{thr}(v) = \deg(v) = 1$. Then delete v from G.*

The correctness of Reduction Rule 5 follows immediately from Observation 2 and one easily verifies that Reduction Rule 5 can be exhaustively applied in linear time. In a graph reduced with respect to Reduction Rules 1 and 5, we can bound the number of vertices with degree at least three by $2f$.

Lemma 2. *A graph with feedback edge set number f reduced with respect to Reduction Rules 1 and 5 has at most $2f$ vertices with degree at least three.*

With Lemma 2 we bound the number of vertices with degree at least three in graphs that are reduced with respect to Reduction Rules 1 and 5. Since these graphs do not contain degree-one vertices, it remains to bound the degree-two vertices.

Reduction Rule 6. *Let (G, thr, k) be an instance for* TSS *reduced with respect to Reduction Rule 1 and let (u, v, w) be a path in G with $\deg(u) = \deg(v) = \deg(w) = 2$ where u and w are neither neighbors nor twins. If there is a vertex $x \in \{u, v, w\}$ with $\mathrm{thr}(x) = 1$, then bypass x, otherwise, bypass u and w and decrease k by one.*

Theorem 7. TARGET SET SELECTION *admits a problem kernel of size $O(f)$, where f denotes the feedback edge set number. The kernelization runs in linear time.*

Proof. Let (G, thr, k) be an instance of TSS reduced with respect to Reduction Rules 1, 5, and 6. Furthermore, let F denote an optimal feedback edge set for G. Thus, $T := G - F$ is a forest. Consider the graph T^* that results from bypassing all vertices in T having degree two in both G and T. It is easy to see that all vertices in T^* have the same degree in T^* as they have in T. Hence, each vertex v with degree two in T^* is incident to an edge of F in G because otherwise it would have been bypassed. Furthermore, each leaf in T^* not incident to an edge of F is also a leaf in G, but since G is reduced with respect to Reduction Rules 1 and 5, each leaf of T^* is incident to an edge of F. Thus, the number of vertices of T^* with degree at most two is bounded by $2f$. Furthermore, by Lemma 2, the number of vertices of T^* with degree at least three is bounded by $2f$. Thus, the overall number of vertices in T^* is at most $4f$. Finally, since G is reduced with respect to Reduction Rule 6, for each edge $\{u, v\}$ of T^*, there are at most two vertices between u and v in T. Hence, the overall number of vertices of T can be bounded by $12f$ and since T is a forest, so can the overall number of edges of T. By construction of T, we can bound the overall number of vertices of G by $12f$ and the overall number of edges of G by $13f$.

It can be shown that any graph can be reduced with respect to Reduction Rules 1, 5, and 6 in linear time. □

FPT algorithm for TSS. Theorem 7 already implies that TSS is fixed-parameter tractable by applying a brute-force search to the size-$O(f)$ problem kernel. Here, we develop a fixed-parameter algorithm with much better efficiency.

Our algorithm computes a solution S for an input instance (G, thr, k) for TSS if there is one. The algorithm runs in two phases. In Phase 1, we branch on degree-two vertices that are in a cycle of G. In Phase 2, we alternatingly try solutions for a cyclic subgraph and apply previous data reduction rules.

In the algorithm, we use the notion of branching rules. Here, we analyze an instance I and replace it with a set $\mathcal{I}$ of new instances. The creation of new instances is defined in branching rules, which we call *correct* if the original instance I is a yes-instance if and only if there is a yes-instance in $\mathcal{I}$. In analogy to data reduction rules, instances that are not subject to a specific branching rule $\mathcal{R}$ are called *reduced* with respect to $\mathcal{R}$. By considering each application of a branching rule as a vertex with parent I and children $\mathcal{I}$, we can define a *search tree* for each input instance.

In the following, we denote the set of all vertices of a graph G being part of cycles by $V_{\mathcal{C}}(G)$. Let C be some two-edge connected component of G containing at least three vertices, that is, C is a non-singleton connected component that remains connected after removing all bridges from G.

Phase 1. In the first phase of our algorithm we branch on vertices of $V_{\mathcal{C}}(G)$ with degree two and threshold two. To this end, we present a branching rule with branching number two that decreases the feedback edge set number f in every application. This lets us bound the search tree size by 2^f.

Branching Rule 1. *Let $I := (G = (V, E), \mathrm{thr}, k)$ be an instance of* TSS *and let $v \in V \cap V_{\mathcal{C}}(G)$ and $\mathrm{thr}(v) = \deg(v) = 2$. Then, create the instance of I that is reduced with respect to $\{v\}$ and create the instance $(G - v, \mathrm{thr}, k)$.*

It is easy to see that Branching Rule 1 branches into at most two cases. After each application of Branching Rule 1, our algorithm exhaustively applies Reduction Rules 1 and 5 to the created instances. If none of Branching Rule 1 and Reduction Rules 1 and 5 applies to any of the created instances, then Phase 2 of the algorithm begins.

Phase 2. At the beginning of the second phase, no input instance is subject to Branching Rule 1 or Reduction Rules 1 and 5. In Phase 2, we branch on vertices with degree at least three that are contained in cycles, followed by the application of Reduction Rules 1 and 5. This process is iterated for all input instances until all input instances can trivially be determined to be yes- or no-instances.

Let $I_1 := (G_1, \mathrm{thr}_1, k_1)$ denote an input instance for Phase 2. For a two-edge connected component C of G_1, we denote the set of vertices of $V_{\mathcal{C}}(C)$ that have degree two by $V_2(C)$. Note that, since G_1 is reduced with respect to Reduction Rule 1 and Branching Rule 1, all vertices in $V_2(C)$ have threshold one. Hence, by Observation 2, there is an optimal solution for I_1 that does not contain vertices of V_2. Note that, if each two-edge connected component of G_1 is contracted into a single vertex, then the remaining graph is a forest. In the following, we call this forest the *component forest* T_1 of G_1. Since G_1 is reduced with respect to Reduction Rules 1 and 5, every leaf of T_1 corresponds to a contracted non-singleton two-edge connected component of G.

Branching Rule 2. *Let $I_1 := (G_1, \mathrm{thr}_1, k_1)$ be an instance of* TSS *reduced with respect to Reduction Rules 1 and 5 and let T_1 be the component forest of G_1. Furthermore, let C denote a two-edge connected component of G_1 corresponding to a leaf in T_1, let v be the only vertex in $N_{G_1}(V(C))$, and let w be the only vertex in $N(v) \cap V(C)$. Then, for each $V' \subseteq V(C) \setminus V_2$, create a new instance by modifying I_1 in the following way. If V' is a target set for C, then create the instance of I_1 that is reduced with respect to V'. Else, if V' is a target set for C with $\mathrm{thr}_1(w)$ decreased by one, then delete $V(C)$ from G_1 and decrease k_1 by $|V'|$. Otherwise, reduce to a trivial no-instance.*

Note that one application of Branching Rule 2 can be pictured as branching into the cases $v \in S$ and $v \notin S$ for each vertex $v \in V_{\mathcal{C}}(G_1) \setminus V_2$. By Lemma 2, $|V_{\mathcal{C}}| \leq 2f_1$, with f_1 denoting the feedback edge set number of G_1. Since each application of Branching Rule 1 decreases the feedback edge set number of the input, continuing the search tree of Phase 1 with Branching Rule 2 leads to a search tree of depth at most $2f$ and each node in the search tree has at most two children. Therefore, the overall search tree size is at most 4^f. After the exhaustive application of Branching Rule 2 and Reduction Rules 1 and 5, we can decide for each remaining instance whether it is a yes-instance of TSS or not. If some instance is a yes-instance, then the original instance I is also a yes-instance of TSS. Otherwise, I is a no-instance of TSS.

Theorem 8. TARGET SET SELECTION *can be solved in $4^f \cdot n^{O(1)}$ time, where f denotes the feedback edge set number.*

6 Conclusion and Open Problems

Following the spirit of multivariate algorithmics [18], we studied natural parameterizations of TSS. We confirmed the intuition deriving from the hardness of DOMINATING SET that TSS remains hard for the parameter diameter. However, for the structural parameters "cluster edge deletion number", "vertex cover number", and "feedback edge set number" we established tractability results by showing fixed-parameter algorithms and kernelizations. Since for several applications the parameter "number of activation rounds" appears to be relevant and should be small, this motivates to investigate the influence of this parameter on the complexity of TSS. As it turns out, however, TSS is already NP-hard for only two rounds; similarly to diameter, this calls for the combination with other parameters. Finally, note that we focused on activating *all* graph vertices; it is natural to extend our studies to cases when only a specified fraction of the graph vertices shall be activated at minimum cost.

References

1. O. Ben-Zwi, D. Hermelin, D. Lokshtanov, and I. Newman. An exact almost optimal algorithm for target set selection in social networks. In *Proc. 10th ACM EC*, pages 355–362. ACM Press, 2009.
2. H. L. Bodlaender. Kernelization: New upper and lower bound techniques. In *Proc. 4th IWPEC*, volume 5917 of *LNCS*, pages 17–37. Springer, 2009.

3. H. L. Bodlaender, S. Thomassé, and A. Yeo. Analysis of data reduction: Transformations give evidence for non-existence of polynomial kernels. Technical Report UU-CS-2008-030, Department of Information and Computing Sciences, Utrecht University, 2008.
4. A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph Classes: a Survey*, volume 3 of *SIAM Monographs on Discrete Mathematics and Applications*. SIAM, 1999.
5. N. Chen. On the approximability of influence in social networks. *SIAM Journal on Discrete Mathematics*, 23(3):1400–1415, 2009.
6. M. Dom, D. Lokshtanov, and S. Saurabh. Incompressibility through colors and IDs. In *Proc. 36th ICALP*, volume 5555 of *LNCS*, pages 378–389. Springer, 2009.
7. P. Domingos and M. Richardson. Mining the network value of customers. In *Proc. 7th ACM KDD*, pages 57–66. ACM Press, 2001.
8. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.
9. M. R. Fellows, D. Lokshtanov, N. Misra, F. A. Rosamond, and S. Saurabh. Graph layout problems parameterized by vertex cover. In *Proc. 19th ISAAC*, volume 5369 of *LNCS*, pages 294–305. Springer, 2008.
10. J. Fiala, P. A. Golovach, and J. Kratochvíl. Parameterized complexity of coloring problems: Treewidth versus vertex cover. In *Proc. 6th TAMC*, volume 5532 of *LNCS*, pages 221–230. Springer, 2009.
11. J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.
12. D. W. Franks, J. Noble, P. Kaufmann, and S. Stagl. Extremism propagation in social networks with hubs. *Adaptive Behavior*, 16(4):264–274, 2008.
13. J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, 38(1):31–45, 2007.
14. D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proc. 9th ACM KDD*, pages 137–146. ACM Press, 2003.
15. C. McCartin, P. Rossmanith, and M. Fellows. Frontiers of intractability for dominating set, 2007. Manuscript.
16. R. M. McConnell and J. Spinrad. Linear-time modular decomposition and efficient transitive orientation of comparability graphs. In *Proc. 5th SODA*, pages 536–545. ACM/SIAM, 1994.
17. R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.
18. R. Niedermeier. Reflections on multivariate algorithmics and problem parameterization. In *Proc. 27th STACS*, volume 5 of *LIPIcs*, pages 17–32. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010.
19. J. J. Potterat, L. Phillips-Plummer, S. Q. Muth, R. B. Rothenberg, D. E. Woodhouse, T. S. Maldonado-Long, H. P. Zimmerman, and J. B. Muth. Risk network structure in the early epidemic phase of HIV transmission in Colorado Springs. *Sexually Transmitted Infections*, 78:159–163, 2002.
20. R. Shamir, R. Sharan, and D. Tsur. Cluster graph modification problems. *Discrete Applied Mathematics*, 144(1–2):173–182, 2004.