View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Dagstuhl Research Online Publication Server

Solving Target Set Selection with Bounded Thresholds Faster than 2^n

Ivan Bliznets
St. Petersburg Department of Steklov Institute of Mathematics of the Russian Academy of Sciences, St. Petersburg, Russia
iabliznets@gmail.com
https://orcid.org/0000-0003-2291-2556

Danil Sagunov
St. Petersburg Department of Steklov Institute of Mathematics of the Russian Academy of Sciences, St. Petersburg, Russia
danilka.pro@gmail.com
https://orcid.org/0000-0003-3327-9768

Abstract

In this paper we consider the Target Set Selection problem. The problem naturally arises in many fields like economy, sociology, medicine. In the Target Set Selection problem one is given a graph G with a function $\text{thr} : V(G) \to \mathbb{N} \cup \{0\}$ and integers k, ℓ. The goal of the problem is to activate at most k vertices initially so that at the end of the activation process there is at least ℓ activated vertices. The activation process occurs in the following way: (i) once activated, a vertex stays activated forever; (ii) vertex v becomes activated if at least $\text{thr}(v)$ of its neighbours are activated. The problem and its different special cases were extensively studied from approximation and parameterized points of view. For example, parameterizations by the following parameters were studied: treewidth, feedback vertex set, diameter, size of target set, vertex cover, cluster editing number and others.

Despite the extensive study of the problem it is still unknown whether the problem can be solved in $\mathcal{O}^*\left((2-\epsilon)^n\right)$ time for some $\epsilon > 0$. We partially answer this question by presenting several faster-than-trivial algorithms that work in cases of constant thresholds, constant dual thresholds or when the threshold value of each vertex is bounded by one-third of its degree. Also, we show that the problem parameterized by ℓ is W[1]-hard even when all thresholds are constant.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis, Theory of computation → Parameterized complexity and exact algorithms, Theory of computation → Branch-and-bound

Keywords and phrases exact exponential algorithms, target set, vertex thresholds, social influence, irreversible conversions of graphs, bootstrap percolation

Digital Object Identifier 10.4230/LIPIcs.IPEC.2018.22

Related Version Full version permanently available at `https://arxiv.org/abs/1807.10789`.

Funding This research was supported by the Russian Science Foundation (project 16-11-10123)

1 Introduction

In this paper we consider the Target Set Selection problem. In the problem one is given a graph G with a function $\text{thr} : V(G) \to \mathbb{N} \cup \{0\}$ (a *threshold function*), and two integers k, ℓ. The question of the problem is to find a vertex subset $S \subseteq V(G)$ (a *target set*) such that $|S| \leq k$ and if we initially activate S then eventually at least ℓ vertices of G become activated.


© Ivan Bliznets and Danil Sagunov;
licensed under Creative Commons License CC-BY

The activation process is defined by the following two rules: (i) if a vertex becomes activated it stays activated forever; (ii) vertex v becomes activated if either it was activated initially or at some moment there is at least $\text{thr}(v)$ activated vertices in the set of its neighbours $N(v)$. Often in the literature by Target Set Selection people refer to the special case of Target Set Selection where $\ell = |V(G)|$, i.e. where we need to activate all vertices of the graph. We refer to this special case as Perfect Target Set Selection.

Target Set Selection problem naturally arises in such areas as economy, sociology, medicine. Let us give an example of a scenario [24, 6] under which Target Set Selection may arise in the marketing area. Often people start using some product when they find out that some number of their friends are already using it. Keeping this in mind, it is reasonable to start the following advertisement campaign of a product: give out the product for free to some people; these people start using the product, and then some friends of these people start using the product, then some friends of these friends and so on. For a given limited budget for the campaign we would like to give out the product in a way that eventually we get the most users of the product. Or we may be given the desired number of users of the product and we would like to find out what initial budget is sufficient. It is easy to see that this situation is finely modelled by the Target Set Selection problem.

The fact that Target Set Selection naturally arises in many different fields leads to a situation that the problem and its different special cases were studied under different names: Irreversible k-Conversion Set [10, 16], P_3-Hull Number [3], r-Neighbour Bootstrap Percolation [4], (k, ℓ)-Influence [5], monotone dynamic monopolies [27], a generalization of Perfect Target Set Selection on the case of oriented graphs is known as Chain Reaction Closure and t-Threshold Starting Set [1]. In [10], Centeno et al. showed that Perfect Target Set Selection is NP-hard even when all threshold values are equal to two.

There is an extensive list of results on Target Set Selection from parameterized and approximation point of view. Many different parameterizations were studied in the literature such as size of the target set, treewidth, feedback vertex set, diameter, vertex cover, cluster editing number and others (for more details, see Table 1). Most of these studies consider the Perfect Target Set Selection problem, i.e. the case where $\ell = |V(G)|$. However, FPT membership results for parameters treewidth [6] and cliquewidth [23] were given for the general case of Target Set Selection. From approximation point of view, it is known that the minimization version (minimize the number of vertices in a target set for a fixed ℓ) of the problem is very hard and cannot be approximated within $\mathcal{O}(2^{\log^{1-\epsilon} n})$ factor for any $\epsilon > 0$, unless NP $\subseteq$ DTIME$(n^{polylog(n)})$. This inappoximability result holds even for graphs of constant degree with all thresholds being at most two [11]. Also, the maximization version of the problem (maximize the number of activated vertices for a fixed k) is NP-hard to approximate within a factor of $n^{1-\epsilon}$ for any $\epsilon > 0$ [24].

Taking into account many intractability results for the problem, it is natural to ask whether we can beat a trivial brute-force algorithm for this problem or its important subcase Perfect Target Set Selection. In other words, can we construct an algorithm with running time $\mathcal{O}^*((2-\epsilon)^n)$ for some $\epsilon > 0$. Surprisingly, the answer to this question is still unknown. Note that the questions whether we can beat brute-force naturally arise in computer science and have significant theoretic importance. Probably, the most important such question is SETH hypothesis which informally can be stated as:

▶ **Hypothesis 1** (SETH). *There is no algorithm for SAT with running time $\mathcal{O}^*((2-\epsilon)^n)$ for any $\epsilon > 0$.*

Another example of such question is the following hypothesis:

Table 1 Some known results on different parameterizations of Perfect Target Set Selection. In the Thresholds column we indicate restrictions on the threshold function under which the results were obtained. Here t denotes the maximum threshold value.

Parameter	Thresholds	Result	Reference
Bandwidth b	general	$\mathcal{O}^*\left(b^{\mathcal{O}(b \log b)}\right)$	Chopin et al. [12]
Clique Cover Number c	general	NP-hard for $c = 2$	Chopin et al. [12]
Cliquewidth cw	constant	$\mathcal{O}^*\left((\mathrm{cw} \cdot t)^{\mathcal{O}(\mathrm{cw} \cdot t)}\right)$	Hartmann [23]
Cluster Editing Number ζ	general	$\mathcal{O}^*\left(16^{\zeta}\right)$	Nichterlein et al. [26]
Diameter d	general	NP-hard for $d = 2$	Nichterlein et al. [26]
Feedback Edge Set Number f	general	$\mathcal{O}^*\left(4^f\right)$	Nichterlein et al. [26]
Feedback Vertex Set Number	general	W[1]-hard	Ben-Zwi et al. [6]
Neighborhood Diversity nd	majority	$\mathcal{O}^*\left(\mathrm{nd}^{\mathcal{O}(\mathrm{nd})}\right)$	Dvořák et al. [17]
	general	W[1]-hard	Dvořák et al. [17]
Target Set Size k	constant	W[P]-complete	Abrahamson et al. [1], Bazgan et al. [5]
Treewidth w	constant	$\mathcal{O}^*\left(t^{\mathcal{O}(w \log w)}\right)$	Ben-Zwi et al. [6]
	majority	W[1]-hard	Chopin et al. [12]
Vertex Cover Number τ	general	$\mathcal{O}^*\left(2^{(2^{\tau}+1)\cdot\tau}\right)$	Nichterlein et al. [26]

▶ **Hypothesis 2.** *[29] For every hereditary graph class Π that can be recognized in polynomial time, the* Maximum Induced Π-Subgraph *problem can be solved in $\mathcal{O}^*\left((2-\epsilon)^n\right)$ time for some $\epsilon > 0$.*

There is a significant number of papers [9, 28, 19, 18, 14, 30, 22, 13, 15, 7, 20] with the main motivation to present an algorithm faster than the trivial one.

As in the stated hypotheses and mentioned papers, our goal is to come up with an algorithm that works faster than brute-force. We partially answer this question by presenting several $\mathcal{O}^*\left((2-\epsilon)^n\right)$ running time algorithms for Target Set Selection when thresholds, i.e. the values of $\mathrm{thr}(v)$, are bounded by some fixed constant and in case when the values of $\mathrm{thr}(v) - \deg(v)$, so-called *dual thresholds*, are bounded by some fixed constant for every $v \in V(G)$. We think that this result may be interesting mainly because of the following two reasons. Firstly, the result is established for a well-studied problem with many applications and hence can reveal some important combinatorial or algorithmic structure of the problem. Secondly, maybe by resolving the asked question we could make progress in resolving hypotheses 1, 2.

Our results. In this paper, we establish the following algorithmic results.

Perfect Target Set Selection can be solved in

- $\mathcal{O}^*\left(1.90345^n\right)$ if for every $v \in V(G)$ we have $\mathrm{thr}(v) \leq 2$;
- $\mathcal{O}^*\left(1.98577^n\right)$ if for every $v \in V(G)$ we have $\mathrm{thr}(v) \leq 3$;
- $\mathcal{O}^*\left((2-\epsilon_d)^n\right)$ randomized time if for every $v \in V(G)$ we have $\mathrm{thr}(v) \geq \deg(v) - d$.

Target Set Selection can be solved in

- $\mathcal{O}^*\left(1.99001^n\right)$ if for every $v \in V(G)$ we have $\mathrm{thr}(v) \leq \lceil \frac{\deg(v)}{3} \rceil$;
- $\mathcal{O}^*\left((2-\epsilon_t)^n\right)$ if for every $v \in V(G)$ we have $\mathrm{thr}(v) \leq t$.

We also prove the following lower bound.

Target Set Selection parameterized by ℓ is W[1]-hard even if

- $\mathrm{thr}(v) = 2$ for every $v \in V(G)$.

2 Preliminaries

2.1 Notation and problem definition

We use standard graph notation. We consider only simple graphs, i.e. undirected graphs without loops and multiple edges. By $V(G)$ we denote the set of vertices of G and by $E(G)$ we denote the set of its edges. We let $n = |V(G)|$. $N(v)$ denotes the set of neighbours of vertex $v \in V(G)$, and $N[v] = N(v) \cup \{v\}$. $\Delta(G) = \max_{v \in V(G)} \deg(v)$ denotes the maximum degree of G. By $G[F]$ we denote the subgraph of G induced by a set F of its vertices. Define by $\deg_F(v)$ the degree of v in the subgraph $G[F]$.

By $X_1 \sqcup X_2 \sqcup \ldots \sqcup X_m$ we denote the disjoint union of sets $X_1, X_2, \ldots, X_m$, i.e. $X_1 \sqcup X_2 \sqcup \ldots \sqcup X_m = X_1 \cup X_2 \cup \ldots \cup X_m$ with the additional restriction that $X_i \cap X_j = \emptyset$ for any distinct i, j.

For a graph G, threshold function thr and $X \subseteq V(G)$ we put $\mathcal{S}_0(X) = X$ and for every $i > 0$ we define $\mathcal{S}_i(X) = \mathcal{S}_{i-1}(X) \cup \{v \in V(G) : |N(v) \cap \mathcal{S}_{i-1}(X)| \geq \text{thr}(v)\}$. We say that v *becomes activated in the* i^{th} *round*, if $v \in \mathcal{S}_i(X) \setminus \mathcal{S}_{i-1}(X)$, i.e. v is not activated in the $(i-1)^{\text{th}}$ round and is activated in the i^{th} round. By *activation process* yielded by X we mean the sequence $\mathcal{S}_0(X), \mathcal{S}_1(X), \ldots, \mathcal{S}_i(X), \ldots, \mathcal{S}_n(X)$. Note that $\mathcal{S}_n(X) = \mathcal{S}_{n+1}(X)$ as $\mathcal{S}_i(X) \subseteq \mathcal{S}_{i+1}(X)$ and n rounds is always enough for the activation process to converge. By $\mathcal{S}(X)$ we denote the set of vertices that eventually become activated, and we say that X *activates* $\mathcal{S}(X)$ *in* (G, thr). Thus, $\mathcal{S}(X) = \mathcal{S}_n(X)$.

We recall the definition of Target Set Selection.

Target Set Selection

Input: A graph G with thresholds $\text{thr} : V(G) \to \mathbb{N} \cup \{0\}$, integers k, ℓ.

Question: Is there a set $X \subseteq V(G)$ such that $|X| \leq k$ and $|\mathcal{S}(X)| \geq \ell$?

We call a solution X of Target Set Selection a *target set of* (G, thr).

By Perfect Target Set Selection we understand a special case of Target Set Selection with $\ell = n$. We call X a *perfect target set of* (G, thr), if it activates all vertices of G, i.e. $\mathcal{S}(X) = V(G)$.

Most of the algorithms described in this paper are recursive algorithms that use branching technique. Such algorithms are described by *reduction rules*, that are used to simplify a problem instance, and *branching rules*, that are used to solve an instance by recursively solving smaller instances. If a branching rule branches an instance of size n into r instances of size $n - t_1, n - t_2, \ldots, n - t_r$, we call $(t_1, t_2, \ldots, t_r)$ a *branching vector* of this branching rule. By a *branching factor* of a branching rule we understand a constant c that is a solution of a linear reccurence corresponding to some branching vector of this rule; such constants are used to bound the running time of an algorithm following the rule with c^n. Note that a branching rule may have multiple corresponding branching vectors and multiple corresponding branching factors. By the *worst branching factor* of a branching rule (or multiple branching rules, if they are applied within the same algorithm) we understand the largest among its branching factors. We refer to [21] for a more detailed explanation of these aspects.

In our work we also use the following folklore result.

▶ **Lemma 1.** *For any positive integer n and any α such that $0 < \alpha \leq \frac{1}{2}$, we have* $\sum_{i=0}^{\lfloor \alpha n \rfloor} \binom{n}{i} \leq 2^{H(\alpha)n}$, *where* $H(\alpha) = -\alpha \log_2(\alpha) - (1-\alpha)\log_2(1-\alpha)$.

Algorithm: `minimal_pvcs`(G, F, A, Z)

Input: Graph G with $\Delta(G) < t$, vertex subsets F, A, Z such that $F \sqcup A \sqcup Z = V(G)$.
Output: All minimal partial vertex covers S of G such that $S \cap (A \sqcup Z) = A$.

if $\exists v : N[v] \subseteq F$ **then**
  foreach $R \subsetneq N[v]$ **do**
    `minimal_pvcs`$(G, F \setminus N[v], A \sqcup R, Z \sqcup (N[v] \setminus R))$
else
  foreach $R \subseteq F$ **do**
    if $A \sqcup R$ *is a minimal partial vertex cover of* G **then**
      output $A \sqcup R$

Figure 1 Algorithm enumerating all minimal partial vertex covers of a graph.

2.2 Minimal partial vertex covers

▶ **Definition 2.** Let G be a graph. We call a subset $S \subseteq V(G)$ of its vertices a *T-partial vertex cover of G* for some $T \subseteq E(G)$, if the set of edges covered by vertices in S is exactly T, i.e. $T = \{uv : \{u, v\} \cap S \neq \emptyset, uv \in E(G)\}$.

We call a T-partial vertex cover S of G a *minimal partial vertex cover of G* if there is no T-partial vertex cover S' of G with $S' \subsetneq S$. Equivalently, there is no vertex $v \in S$ so that $S \setminus \{v\}$ is a T-partial vertex cover of G.

The following theorem bounds the number of minimal partial vertex covers in graphs of bounded degree. We note that somewhat similar results were proven by Björklund et al. [8].

▶ **Theorem 3.** *For any positive integer t, there is a constant $\omega_t < 1$ and an algorithm that, given an n-vertex graph G with $\Delta(G) < t$ as input, outputs all minimal partial vertex covers of G in $\mathcal{O}^*\left(2^{\omega_t n}\right)$ time.*

Proof. We present a recursive branching algorithm that lists all minimal partial vertex covers of G. Pseudocode of the algorithm is presented in Figure 1. As input, the algorithm takes three sets F, A, Z such that $F \sqcup A \sqcup Z = V(G)$. The purpose of the algorithm is to enumerate all minimal partial vertex covers that contain A as a subset and do not intersect with Z. So the algorithm outputs all minimal partial vertex covers S of G satisfying $S \cap (A \sqcup Z) = A$. It easy to see that then `minimal_pvcs`$(G, V(G), \emptyset, \emptyset)$ enumerates all minimal partial vertex covers of G.

The algorithm uses only the following branching rule. If there is a vertex $v \in F$ such that $N(v) \subseteq F$ then consider $2^{|N[v]|} - 1$ branches. In each branch, take some $R \subsetneq N[v]$ and run `minimal_pvcs`$(G, F \setminus N[v], A \sqcup R, Z \sqcup (N[v] \setminus R))$. In other words, we branch on which vertices in $N[v]$ belong to minimal partial vertex cover and which do not. Note that if S is a minimal partial vertex cover then it cannot contain $N[v]$, since otherwise $S \setminus \{v\}$ is its proper subset and covers the same edges. Hence, above branching consider all possible cases. Since $\Delta(G) < t$, the worst branching factor is $(2^t - 1)^{\frac{1}{t}}$.

If the branching rule cannot be applied then we apply brute-force on all possible variants of the intersection of the minimal partial vertex cover S and the set F. So we consider all $2^{|F|}$ variants of $S \cap F$, and filter out variants that do not correspond to a minimal partial vertex cover. Minimality of a partial vertex cover can be checked in polynomial time, so filtering out adds only a polynomial factor.

Note that we run brute-force only if every vertex in F has at least one neighbour in $A \sqcup Z$, in other words, $A \sqcup Z$ is a dominating set of G. Since $\Delta(G) < t$, any dominating set of G consists of at least $\frac{n}{t}$ vertices. Hence, $|F| \leq \frac{(t-1)n}{t}$. This leads to the following upper bound on the running time of the algorithm:

$$\left(\left(2^t - 1\right)^{\frac{1}{t}}\right)^{\frac{n}{t}} \cdot 2^{\frac{(t-1)n}{t}} \cdot n^{\mathcal{O}(1)}.$$

Hence, we can put $\omega_t = \frac{1}{t^2}\log\left(2^t - 1\right) + \frac{t-1}{t} < 1$. ◀

3 Algorithms for bounded thresholds

3.1 Algorithm for thresholds bounded by fixed constant

In this subsection we prove the following theorem.

▶ **Theorem 4.** *Let t be a fixed constant. For* Target Set Selection *with all thresholds bounded by t there is a $\mathcal{O}^*\left((2-\epsilon_t)^n\right)$-time algorithm, where ϵ_t is a positive constant that depends only on t.*

Our algorithm consists of three main stages. In the first stage we apply some simple reduction and branching rules. If the instance becomes small enough we then apply brute-force and solve the problem. Otherwise, we move to the second stage of the algorithm. In the second stage we perform branching rules that help us describe the activation process. After that we move to the third stage in which we run special dynamic program that finally solves the problem for each branch. Let us start the description of the algorithm.

3.1.1 Stage I

In the first stage our algorithm applies some branching rules. In each branch we maintain the following partition of $V(G)$ into three parts A, Z, F. These parts have the following meaning: A is the set of vertices that are known to be in our target set, Z — the set of vertices that are known to be not in the target set, F — the set of all other vertices (i.e. vertices about that we do not know any information so far). At the beginning, we have $A = Z = \emptyset$ and $F = V(G)$.

We start the first stage with exhaustive application of reduction rule 1 and branching rule 1.

▶ **Reduction rule 1.** *If there is any vertex $v \in \mathcal{S}(A)$, but $v \notin A \sqcup Z$, then assign v to Z.*

Reduction rule 1 is correct as there is no need to put a vertex in a target set if it will become activated eventually by the influence of its neighbours.

▶ **Branching rule 1.** *If there is a vertex $v \in F$ such that $\deg_F(v) \geq \mathrm{thr}(v)$ then arbitrarily choose a subset $T \subseteq N(v) \cap F$ such that $|T| = \mathrm{thr}(v)$ and branch on the following branches:*

1. *For each subset of vertices $S \subseteq T \cup \{v\}$ of size less than $\mathrm{thr}(v)$ consider a branch in which we put S into A and we put other vertices $T \cup \{v\} \setminus S$ into Z;*
2. *Additionally consider the branch in which we assign all vertices from T to A and v is assigned to Z.*

It is enough to consider only above-mentioned branches. All other possible branches assign at least $\mathrm{thr}(v)$ vertices from $T \cup \{v\}$ to A, and we always can replace such branch with the branch assigning T to A, since it leads to the activation of all vertices in $T \cup \{v\}$

and adds at most the same number of vertices into a target set. Branching rule 1 considers $2^{\mathrm{thr}(v)+1} - \mathrm{thr}(v) - 1$ options for $\mathrm{thr}(v)+1$ vertices, thus it gives the biggest branching factor of $(2^{t+1} - t - 1)^{\frac{1}{t+1}}$ (here and below $t = \max_{v \in V(G)} \mathrm{thr}(v)$).

▶ **Branching rule 2.** *If $|F| \leq \gamma n$, where γ is a constant to be chosen later, then simply apply brute-force on how vertices in F should be assigned to A and Z.*

If branching rule 2 is applied in all branches then the running time of the whole algorithm is at most $2^{\gamma n}(2^{t+1} - t - 1)^{\frac{(1-\gamma)n}{t+1}}$ and we do not need to use stages II and III, as the problem is already solved in this case.

3.1.2 Stage II

After exhaustive application of reduction rule 1 and branching rules 1 and 2, in each branch we either know the answer or we have the following properties:

1. $\Delta(G[F]) < t$;
2. $|F| > \gamma n$;
3. $\mathcal{S}(A) \subseteq A \sqcup Z$.

Now, in order to solve the problem it is left to identify the vertices of a target set that belong to F. It is too expensive to consider all $2^{|F|}$ subsets of F as F is too big. Instead of this direct approach (brute-force on all subsets of F) we consider several subbranches. In each such branch we almost completely describe the activation process of the graph. For each branch, knowing this information about the activation process, we find an appropriate target set by solving a special dynamic program in stage III.

Let X be an answer (a target set). X can be expressed as $X = A \sqcup B$ where $B \subseteq F$. At the beginning of the activation process only vertices in $\mathcal{S}_0(X) = X = A \sqcup B$ are activated, after the first round vertices in $\mathcal{S}_1(A \sqcup B)$ are activated, and so on. It is clear that $\mathcal{S}(A \sqcup B) = \mathcal{S}_n(A \sqcup B)$. Unfortunately, we cannot compute the sequence of $\mathcal{S}_i(A \sqcup B)$ as we do not know B. Instead we compute the sequence $P_0, P_1, \ldots, P_n = P$ such that $P_i \setminus B = \mathcal{S}_i(X) \setminus B$ and $P_i \subseteq P_{i+1}$ for any i.

First of all, using Theorem 3 we list all minimal partial vertex covers of the graph $G[F]$. For each minimal partial vertex cover C we create a branch that indicates that $C \subseteq B$ and, moreover, C covers exactly the same edges in $G[F]$ as B does. In other words, any edge in $G[F]$ has at least one endpoint in B if and only if it has at least one endpoint in C. Note that such C exists for any B. One can obtain C by removing vertices from B one by one while it covers the same edges as B. When no vertex can be removed, then, by definition, the remaining vertices form a minimal partial vertex cover.

Put $P_0 = A \sqcup C$. It is correct since $\mathcal{S}_0(X) \setminus B = A = P_0 \setminus B$. We now show how to find P_{i+1} having P_i. Recall that to do such transition from $\mathcal{S}_i(X)$ to $\mathcal{S}_{i+1}(X)$ it is enough to find vertices with the number of neighbours in $\mathcal{S}_i(X)$ being at least the threshold value of that vertex. As for P_i and P_{i+1}, it is sufficient to check that the number of activated neighbours has reached the threshold only for vertices that are not in B. Thus any transition from P_i to P_{i+1} can be done by using a procedure that, given P_i and any vertex $v \notin P_i$, checks whether v becomes activated in the $(i+1)^{\text{th}}$ round or not, under the assumption that $v \notin B$.

Given P_i it is not always possible to find a unique P_{i+1} as we do not know B. That is why in such cases we create several subbranches that indicate potential values of P_{i+1}.

Let us now show how to, for each vertex $v \notin P_i$, figure out whether v is in P_{i+1} (see pseudocode in Figure 2). Since we know P_i and $P_i \subseteq P_{i+1}$, we assume that $v \notin P_i$.

If $|N(v) \cap P_i| \geq \mathrm{thr}(v)$ then we simply include v in P_{i+1}. We claim that this check is enough for $v \in F$.

▶ **Claim 1.** *If $v \in F \setminus B$, then v becomes activated in the i^{th} round if and only if $|N(v) \cap P_i| \geq \mathrm{thr}(v)$.*

Proof. We show that by proving that $\mathcal{S}_i(X) \cap N(v) = P_i \cap N(v)$ for every $v \in F \setminus B$. Note that $\mathcal{S}_i(X) \setminus B = P_i \setminus B$ by definition of P_i. So it is enough to prove that $\mathcal{S}_i(X) \cap N(v) \cap B = P_i \cap N(v) \cap B$, which is equivalent to $N(v) \cap B = P_i \cap N(v) \cap B$, as $B \subseteq \mathcal{S}_i(X)$. Since $v \notin B$, then any $uv \in E(G[F])$ is covered by B if and only if $u \in B$. C covers the same edges in $G[F]$ as B does, and also $v \notin C$, hence $C \cap N(v) = B \cap N(v)$. Thus, since $C \subseteq P_0 \subseteq P_i$, we get $P_i \cap B \cap N(v) = P_i \cap C \cap N(v) = C \cap N(v) = B \cap N(v)$. ◀

If $v \in B$, the decision for v does not matter. Thus if $v \in F$ and $|N(v) \cap P_i| < \mathrm{thr}(v)$, we may simply not include v in P_{i+1}.

If $v \in Z$, at this point, we cannot compute the number of activated neighbours of v exactly as we do not know what neighbours of v are in B. Note that we do not need the exact number of such neighbours if we know that this value is at least $\mathrm{thr}(v)$. Thus we branch into $\mathrm{thr}(v) + 1$ subbranches corresponding to the value of $\min\{|N(v) \cap B|, \mathrm{thr}(v)\}$, from now on we denote this value as $dg(v)$.

On the other hand, we know all activated neighbours of v that are in $V(G) \setminus F$ since $\mathcal{S}_i(X) \cap (V(G) \setminus F) = P_i \cap (V(G) \setminus F)$, as $B \subseteq F$. Let this number be $m = |N(v) \cap (P_i \setminus F)|$. So the number of activated neighbours of v is at least $m + dg(v)$. Also there may be some activated neighbours of v in $N(v) \cap P_i \cap F$. However, we cannot simply add $|N(v) \cap P_i \cap F|$ to $m + dg(v)$ since vertices in $P_i \cap B$ will be computed twice. So we are actually interested in the value of $|(N(v) \cap P_i \cap F) \setminus B|$. That is why for vertices from $N(v) \cap P_i \cap F$ we simply branch whether they are in B or not. After that we compare $m + dg(v) + |(N(v) \cap P_i \cap F) \setminus B|$ with $\mathrm{thr}(v)$ and figure out whether v becomes activated in the current round or not.

Note that once we branch on the value of $\min\{|N(v) \cap B|, \mathrm{thr}(v)\}$, or on whether $v \in B$ or not for some v, we will not branch on the same value or make a decision for the same vertex again as it makes no sense. Once fixed, the decision should not change along the whole branch and all of its subbranches, otherwise the information about B would just become inconsistent.

Let us now bound the number of branches created. There are three types of branchings in the second stage:

1. Branching on the value of the minimal partial vertex cover C. By Theorem 3, there is at most $\mathcal{O}^*\left(2^{\omega_t |F|}\right)$ such branches.
2. Branching on the value of $dg(v) = \min\{|N(v) \cap B|, \mathrm{thr}(v)\}$ with $v \in Z$. There is at most $(t+1)^{|Z|}$ such possibilities since $t \geq \min\{|N(v) \cap B|, \mathrm{thr}(v)\} \geq 0$.
3. Branching on whether vertex u is in B or not. We perform this branching only for vertices in the set $N(v) \cap P_i \cap F$ with $v \in Z$ only when its size is strictly smaller than $\mathrm{thr}(v) \leq t$. Hence we perform a branching of this type on at most $(t-1)|Z|$ vertices.

Hence, the total number of the branches created in stage II is at most

$$2^{\omega_t |F|} \cdot (t+1)^{|Z|} \cdot 2^{(t-1)|Z|} \cdot n^{\mathcal{O}(1)}.$$

3.1.3 Stage III

Now, for each branch our goal is to find the smallest set X which activates at least ℓ vertices and agrees with all information obtained during branching in a particular branch. That is,

- $A \subseteq X, Z \cap X = \emptyset$ (branchings made in stage I);
- $C \subseteq X$ (branching of the first type in stage II);

Algorithm: `is_activated`$(G, \mathrm{thr}, A, Z, F, P_i, v)$

Input: G, thr, A, Z, F as usual, P_i such that $P_i \setminus B = \mathcal{S}_i(A \sqcup B) \setminus B$ for some B, and a vertex $v \notin P_i$.

Output: True, if $v \notin B$ and $v \in \mathcal{S}_{i+1}(A \sqcup B)$;
False, if $v \notin B$ and $v \notin \mathcal{S}_{i+1}(A \sqcup B)$;
any answer, otherwise.

if $|N(v) \cap P_i| \geq \mathrm{thr}(v)$ **then**
  return *True*
else if $v \in F$ **then**
  return *False*
$m \longleftarrow |N(v) \cap (P_i \setminus F)|$
branch on the value of $dg(v) = \min\{|N(v) \cap B|, \mathrm{thr}(v)\}$
$m \longleftarrow m + dg(v)$
foreach $u \in P_i \cap N(v) \cap F$ **do**
  branch on whether $u \in B$
  if $u \notin B$ **then**
    $m \longleftarrow m + 1$
return $m \geq \mathrm{thr}(v)$

Figure 2 Procedure determining whether a vertex becomes activated in the current round.

- information about $\min\{|N(v) \cap B|, \mathrm{thr}(v)\}$ (second type branchings in stage II);
- additional information whether certain vertices belong to X or not (third type branchings in stage II).

From now on we assume that we are considering some particular branching leaf. Let A' be the set of vertices that are known to be in X for a given branch and Z' be the set of vertices known to be not in X (note that $A \subseteq A'$ and $Z \subseteq Z'$). Let $Z = \{v_1, v_2, \ldots, v_z\}$ and $F' = V(G) \setminus A' \setminus Z' = \{u_1, u_2, \ldots, u_{f'}\}$. So actually it is left to find $B' \subseteq F'$ (in these new terms, $B = (A' \setminus A) \sqcup B'$) such that $|A' \sqcup B'| \leq k$, $|P \cup A' \cup B'| \geq \ell$ and for each $i \in \{1, 2, \ldots, z\}$ the value $\min\{\mathrm{thr}(v_i), |N(v_i) \cap B|\}$ equals $dg(v_i)$. This is true since the information obtained during branching completely determines the value of P.

In order to solve the obtained problem we employ dynamic programming. We create a table TS of size $f' \times \ell \times (t+1)^z$. For all B'_1 such that $|(B'_1 \cup P) \cap \{u_1, u_2, \ldots, u_i\}| = p$ and $\min\{\mathrm{thr}(v_j), |N(v_j) \cap ((A' \setminus A) \sqcup B'_1)|\} = d_j$, in the field $TS(i, p, d_1, d_2, \ldots, d_z)$ we store any set B'_2 of minimum size such that $A' \sqcup B'_1 \sqcup B'_2$ is a potential solution, i.e. $|\mathcal{S}(A' \sqcup B'_1 \sqcup B'_2)| = |(P \cup B'_1 \cup B'_2)| = |P \cap (V(G) \setminus F')| + p + |B'_2| \geq \ell$ and for every j we have $\min\{\mathrm{thr}(v_j), |N(v_j) \cap ((A' \setminus A) \sqcup B'_1 \sqcup B'_2)|\} = \min\{\mathrm{thr}(v_j), |N(v_j) \cap B'_2| + d_j\} = dg(v_j)$. Note that the choice of B'_2 depends only on values $i, p, d_1, d_2, \ldots, d_z$, but not on the value of B'_1 directly. In other words, $TS(i, p, d_1, d_2, \ldots, d_z)$ stores one of optimal ways of how the remaining $f' - i$ vertices in F' should be chosen into B' if the first i vertices in F' was chosen correspondingly to the values of p and d_j.

Note that for some fields in the TS table there may be no appropriate value of B'_2 (there is no appropriate solution). In such cases, we put the corresponding element to be equal to $V(G)$. It is a legitimate operation since we are solving a minimization problem. Note that

the desired value of B' will be stored as

$$TS(0, 0, \min\{|N(v_1) \cap (A' \setminus A)|, \mathrm{thr}(v_1)\}, \ldots, \min\{|N(v_z) \cap (A' \setminus A)|, \mathrm{thr}(v_z)\}).$$

We assign $TS(f', p, dg(v_1), dg(v_2), \ldots dg(v_z)) = \emptyset$ for every p such that $p + |P \cap (V(G) \setminus F')| \geq \ell$. We do this since values $p, dg(v_1), dg(v_2), \ldots dg(v_z)$ indicate that $A' \sqcup B'_1$ is already a solution. In all other fields of type $TS(f', \cdot, \cdots, \cdot)$ we put the value of $V(G)$. We now show how to evaluate values $TS(i, p, d_1, d_2, \ldots, d_z)$ for any $i \geq 0$ smaller than f'. We can evaluate any $TS(i, \cdot, \cdot, \ldots, \cdot)$ in polynomial time if we have all values $TS(i+1, \cdot, \cdot, \ldots, \cdot)$ evaluated. For each $j \in \{1, 2, \ldots, z\}$, let $d_j^{i+1} = \min\{\mathrm{thr}(v_j), d_j + |N(v_j) \cap \{u_{i+1}\}|\}$. In order to compute $TS(i, p, d_1, d_2, \ldots, d_z)$, we need to decide whether u_{i+1} is in a target set or not. If u_{i+1} is taken into B' then d_j becomes equal to d_j^{i+1} for each j, if it is not, none of d_j should change. Hence, $TS(i, p, \langle d_j \rangle) = \min\left[TS(i+1, p+1, \langle d_j^{i+1} \rangle) \cup \{u_{i+1}\},\ TS(i+1, p + |P \cap \{u_{i+1}\}|, \langle d_j \rangle)\right]$.

Since $0 \leq d_j \leq dg(v_j)$ for any j, the TS table has $\mathcal{O}^*\left((t+1)^{|Z|}\right)$ fields. Each field of the table is evaluated in polynomial time. So the desired B' is found (hence, the solution is found) in $\mathcal{O}^*\left((t+1)^{|Z|}\right)$ time for any branch fixed in stage II. Stages II and III together run in $2^{\omega_t|F|} \cdot (t+1)^{|Z|} \cdot 2^{(t-1)|Z|} \cdot (t+1)^{|Z|} \cdot n^{\mathcal{O}(1)}$ time for any fixed subbranch of stage I.

Actually, the $(t+1)^{2|Z|}$ multiplier in the upper bound can be improved. Recall that it corresponds to the number of possible variants of $dg(v_j)$ and the number of possible variants of d_j. However, note that $d_j \leq dg(v_j)$. So after each of $dg(v_j)$ is fixed in stage II, for d_j there is only $dg(v_j) + 1$ options in stage III. Hence, each of the pairs $(d_j, dg(v_j))$ can be presented only in $\binom{t+2}{2}$ variants. This gives an improvement of the $(t+1)^{2|Z|}$ multiplier to a $\binom{t+2}{2}^{|Z|}$ multiplier. So, the upper bound on the running time in stages II and III becomes $\mathcal{O}^*\left(2^{\omega_t|F|} \cdot \binom{t+2}{2}^{|Z|} \cdot 2^{(t-1)|Z|}\right)$.

We rewrite this upper bound in terms of n and $|F|$. Since $|Z| \leq n - |F|$, the upper bound is

$$2^{\omega_t|F|} \cdot \binom{t+2}{2}^{n-|F|} \cdot 2^{(t-1)(n-|F|)} \cdot n^{\mathcal{O}(1)}.$$

Now we are ready to choose γ. We set the value of γ so that computation in each branch created at the end of stage I takes at most $\mathcal{O}^*(2^{\gamma n})$ time. Note that the upper bound on the running time required for stages II and III increases while the value of $|F|$ decreases. So we can find γ as the solution of equation $2^{\gamma n} = 2^{\omega_t \gamma n} \cdot \binom{t+2}{2}^{(1-\gamma)n} \cdot 2^{(t-1)(1-\gamma)n}$. Hence, $\gamma = \frac{(t-1)+\log_2\binom{t+2}{2}}{(t-\omega_t)+\log_2\binom{t+2}{2}} < 1$, as $\omega_t < 1$. So the overall running time is

$$2^{\gamma n}(2^{t+1} - t - 1)^{\frac{(1-\gamma)n}{t+1}} \cdot n^{\mathcal{O}(1)},$$

which is $\mathcal{O}^*\left((2 - \epsilon_t)^n\right)$ for some $\epsilon_t > 0$ since $\gamma < 1$.

3.2 Two algorithms for constant thresholds in the perfect case

Here, we present two algorithms for special cases of Perfect Target Set Selection with thresholds being at most two or three. These algorithms use the idea that cannot be used in the general case of Target Set Selection, so the running times of these algorithms are significantly faster than the running time of the algorithm from the previous subsection. We provide their full descriptions in the full version of our paper.

▶ **Theorem 5.** Perfect Target Set Selection *with thresholds being at most two can be solved in* $\mathcal{O}^*(1.90345^n)$ *time.*

▶ **Theorem 6.** Perfect Target Set Selection *with thresholds being at most three can be solved in* $\mathcal{O}^*(1.98577^n)$ *time.*

3.3 Algorithm for thresholds bounded by one-third of degrees

Here, we prove the following.

▶ **Theorem 7.** *Let* G *be a connected graph with at least three vertices. Assume that* $\mathrm{thr}(v) \leq \lceil \frac{\deg(v)}{3} \rceil$ *for every* $v \in V(G)$*. Then there is a perfect target set of* (G, thr) *of size at most* $0.45|V(G)|$.

▶ **Corollary 8.** Target Set Selection *with thresholds bounded by one-third of degree rounded up can be solved in* $\mathcal{O}^*(1.99001^n)$ *time.*

In our proofs we use a combinatorial model proposed by Ackerman et al. in [2]. We provide these proofs in the full version of our paper.

4 Algorithm for bounded dual thresholds

Let (G, thr) be a graph with thresholds. By *dual threshold* of vertex $v \in V(G)$ we understand the value $\overline{\mathrm{thr}}(v) = \deg(v) - \mathrm{thr}(v)$. In terms of dual thresholds, v becomes activated if it has at most $\overline{\mathrm{thr}}(v)$ not activated neighbours. For bounded dual thresholds we prove the following theorem.

▶ **Theorem 9.** *For any non-negative integer* d, Perfect Target Set Selection *with dual thresholds bounded by* d *can be solved in* $\mathcal{O}^*((2-\epsilon_d)^n)$ *randomized time for some* $\epsilon_d > 0$.

This result follows from the result of Pilipczuk and Pilipczuk in [28], where they presented an algorithm for the Maximum d-Degenerate Induced Subgraph problem with the same running time. One may find the detailed proof in the full version of our paper.

5 Lower bounds

5.1 ETH lower bound

First of all, we show a $2^{o(n+m)}$ lower bound for Perfect Target Set Selection, where m denotes the number of edges in the input graph. We have not found any source that claims this result. Thus, for completeness, we state it here. The result follows from the reduction given by Centeno et al. in [10]. They showed a linear reduction from a special case of 3-SAT, where each variable appears at most three times, to Perfect Target Set Selection where thresholds are equal to two and maximum degree of the graph is constant. Note that in their work they refer to the problem as IRR_2-Conversion Set.

▶ **Theorem 10.** Perfect Target Set Selection *cannot be solved in* $2^{o(n+m)}$ *time unless ETH fails, even when thresolds are equal to two and maximum degree of the graph is constant.*

One can find the detailed proof in the full version of our paper.

5.2 Parameterization by ℓ

We now look at Target Set Selection from parameterized point of view. In [5], Bazgan et al. proved that Target Set Selection ℓ is W[1]-hard with respect to parameter ℓ, when all dual thresholds are equal to 0. This result also follows from the proof of W[1]-hardness

of Cutting ℓ Vertices given by Marx in [25], with a somewhat different construction. Inspired by his proof, we show that this result holds even when all thresholds are constant.

▶ **Theorem 11.** Target Set Selection *parameterized by ℓ is W[1]-hard even when all thresholds are equal to two.*

Proof. Let (G, k) be an instance of the Clique problem. In order to provide the reduction, we construct a graph G' in which each vertex corresponds to a vertex or an edge of graph G i.e. $V(G') = V(G) \sqcup E(G)$. We add edges in G' between vertices corresponding to $v \in V(G)$ and $e \in E(G)$ if and only if v and e are incident in G.

We will refer to the vertex in G' corresponding to an edge $e \in E(G)$ as $v_e \in V(G')$. If a vertex from G' corresponds to a vertex $u \in G$ we refer to it as v_u. Slightly abusing notation we will refer to the set of vertices in G' corresponding to the vertices $V(G)$ as V and to the set of vertices corresponding to the edges $E(G)$ as E, $V \sqcup E = V(G')$. Consider now an instance of Target Set Selection for G', with the same k, $\ell = k + \binom{k}{2}$ and all thresholds equal to $t = 2$.

If G has a clique of size k, then selecting corresponding vertices as a target set of G' leads to activation of the vertices corresponding to the edges of the clique. Hence, $k + \binom{k}{2}$ vertices will be activated in total.

Let us now prove that if G' has a target set of size at most k activating at least $\ell = k + \binom{k}{2}$ vertices, then G has a clique on k vertices. Let S be such target set of G'. Denote by $k_v = |S \cap V|$ the number of vertices in S corresponding to the vertices of G and by $k_e = |S \cap E|$ the number of vertices in S corresponding to the edges of G, $k_v + k_e \leq k$.

Now, we show how to convert any target set S of size at most k activating at least $k + \binom{k}{2}$ vertices into a target set S' such that $|S'| \leq k$, $S' \subseteq V$ and S' activates at least $k + \binom{k}{2}$ vertices.

Observe that if there is an edge $u_1u_2 = e \in E(G)$ such that $v_e \in S$ and $v_{u_1} \in S$ then $S' = S \setminus \{v_e\} \cup \{v_{u_2}\}$ also activates at least $k + \binom{k}{2}$ vertices and the size of S' is at most k. Thus we can assume that if $v_{u_1u_2} \in S$, then $v_{u_1}, v_{u_2} \notin S$.

Observe that any initially not activated vertex in E becomes activated only if all two of its neighbours are activated. It means that any such vertex does not influence the activation process in future. Hence, since G' is bipartite, the activation process always finishes within two rounds, and no vertex in V becomes activated in the second round.

Let V_1 be the set of vertices of V that become activated by S in the first round, i.e. $V_1 = \mathcal{S}_1(S) \setminus \mathcal{S}_0(S) \cap V$. Note that these vertices are activated directly by k_e vertices in $S \cap E$. Let $S_{E,i}$ be the set of vertices in $S \cap E$ that have exactly i endpoints in V_1. Denote by $k_{e,i}$ the size of $S_{E,i}$. Then we have $k_{e,0} + k_{e,1} + k_{e,2} = k_e$. Note that if there is a vertex in $S \cap E$ with no endpoints in V_1 then one can replace it with any neighbour and size of S will not change and it will activate at least the same number of vertices in G'. Thus we can assume that $k_{e,0} = 0$.

We show that $|V_1| \leq \frac{k_{e,1}}{2} + k_{e,2}$. Indeed, in order to be activated, any vertex from V_1 requires at least two vertices from E to be in the target set. Each vertex from $S_{E,i}$ contributes to exactly i vertices from V_1, and the total number of contributions is $k_{e,1} + 2k_{e,2}$. This number should be at least $2|V_1|$. Hence, $|V_1| \leq \frac{k_{e,1}}{2} + k_{e,2}$.

Consider $S' = S \setminus E \cup V_1$ i.e. we replace all k_e vertices from E with all vertices from V_1. Note that $|S'| \leq |S| - \frac{k_{e,1}}{2}$. Vertices from $S_{E,2}$ become activated in the first round since all of them have two endpoints in S'. Thus S' is now a target set of size not greater than $k - \frac{k_{e,1}}{2}$ activating at least $\ell - k_{e,1}$ vertices in G'.

Note that any vertex from $S_{E,1}$ can be activated by adding one more vertex to S'. Consider set $H = N(S_{E,1}) \setminus V_1$. If $|H| \leq \frac{k_{e,1}}{2}$ then consider $S_1 = H \cup S'$. S_1 compared to S'

will additionally activate all vertices in $S_{E,1}$. Note that S_1 is a target set S of size at most k activating at least ℓ vertices.

If $|H| > \frac{k_{e,1}}{2}$ then construct S_1 from S' by simply adding $\frac{k_{e,1}}{2}$ arbitrary vertices from H. Each of these vertices will additionally activate at least one vertex corresponding to edge, thus S_1 is a target set of size at most k activating at least ℓ vertices.

We have shown how to transform any target set S activating at least $k + \binom{k}{2}$ vertices in G' into a target set S_1 such that $S_1 \subseteq V$ and S_1 activates at least the same number of vertices in G'. As we have shown earlier, no vertex in $E \setminus S_1$ influence the activation process after becoming activated. Then, since $S_1 \cap E = \emptyset$, S_1 activates only vertices in E in the first round and the process finishes. Hence, if the instance for G' has a solution, then G has a clique of size k. ◀

References

1. Karl A. Abrahamson, Rodney G. Downey, and Michael R. Fellows. Fixed-parameter tractability and completeness IV: On completeness for W[P] and PSPACE analogues. *Annals of Pure and Applied Logic*, 73(3):235–276, June 1995. doi:10.1016/0168-0072(94)00034-z.
2. Eyal Ackerman, Oren Ben-Zwi, and Guy Wolfovitz. Combinatorial model and bounds for target set selection. *Theoretical Computer Science*, 411(44-46):4017–4022, October 2010. doi:10.1016/j.tcs.2010.08.021.
3. R.T. Araújo, R.M. Sampaio, and J.L. Szwarcfiter. The convexity of induced paths of order three. *Electronic Notes in Discrete Mathematics*, 44:109–114, November 2013. doi:10.1016/j.endm.2013.10.017.
4. József Balogh, Béla Bollobás, and Robert Morris. Bootstrap percolation in high dimensions. *Combinatorics, Probability and Computing*, 19(5-6):643–692, 2010.
5. Cristina Bazgan, Morgan Chopin, André Nichterlein, and Florian Sikora. Parameterized approximability of maximizing the spread of influence in networks. *Journal of Discrete Algorithms*, 27:54–65, July 2014. doi:10.1016/j.jda.2014.05.001.
6. Oren Ben-Zwi, Danny Hermelin, Daniel Lokshtanov, and Ilan Newman. Treewidth governs the complexity of target set selection. *Discrete Optimization*, 8(1):87–96, February 2011. doi:10.1016/j.disopt.2010.09.007.
7. Daniel Binkele-Raible, Ljiljana Brankovic, Marek Cygan, Henning Fernau, Joachim Kneis, Dieter Kratsch, Alexander Langer, Mathieu Liedloff, Marcin Pilipczuk, Peter Rossmanith, and Jakub Onufry Wojtaszczyk. Breaking the 2^n-barrier for Irredundance: Two lines of attack. *Journal of Discrete Algorithms*, 9(3):214–230, September 2011. doi:10.1016/j.jda.2011.03.002.
8. Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. The traveling salesman problem in bounded degree graphs. *ACM Transactions on Algorithms*, 8(2):1–13, April 2012. doi:10.1145/2151171.2151181.
9. Ivan Bliznets, Fedor V. Fomin, Michał Pilipczuk, and Yngve Villanger. Largest Chordal and Interval Subgraphs Faster Than 2^n. In *Lecture Notes in Computer Science*, pages 193–204. Springer Berlin Heidelberg, 2013. doi:10.1007/978-3-642-40450-4_17.
10. Carmen C. Centeno, Mitre C. Dourado, Lucia Draque Penso, Dieter Rautenbach, and Jayme L. Szwarcfiter. Irreversible conversion of graphs. *Theoretical Computer Science*, 412(29):3693–3700, July 2011. doi:10.1016/j.tcs.2011.03.029.
11. Ning Chen. On the Approximability of Influence in Social Networks. *SIAM Journal on Discrete Mathematics*, 23(3):1400–1415, January 2009. doi:10.1137/08073617x.
12. Morgan Chopin, André Nichterlein, Rolf Niedermeier, and Mathias Weller. Constant Thresholds Can Make Target Set Selection Tractable. *Theory of Computing Systems*, 55(1):61–83, September 2013. doi:10.1007/s00224-013-9499-3.

13 Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. Scheduling Partially Ordered Jobs Faster than 2^n. *Algorithmica*, 68(3):692–714, October 2012. doi:10.1007/s00453-012-9694-7.

14 Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. Solving the 2-Disjoint Connected Subgraphs Problem Faster than 2^n. *Algorithmica*, 70(2):195–207, May 2013. doi:10.1007/s00453-013-9796-x.

15 Marek Cygan, Marcin Pilipczuk, and Jakub Onufry Wojtaszczyk. Capacitated Domination Faster Than $O(2^n)$. In *Lecture Notes in Computer Science*, pages 74–80. Springer Berlin Heidelberg, 2010. doi:10.1007/978-3-642-13731-0_8.

16 Paul A. Dreyer Jr. and Fred S. Roberts. Irreversible k-threshold processes: Graph-theoretical threshold models of the spread of disease and of opinion. *Discrete Applied Mathematics*, 157(7):1615–1627, 2009.

17 Pavel Dvořák, Dušan Knop, and Tomáš Toufar. Target Set Selection in Dense Graph Classes. *arXiv preprint arXiv:1610.07530*, 2016.

18 Fedor V. Fomin, Serge Gaspers, Artem V. Pyatkin, and Igor Razgon. On the Minimum Feedback Vertex Set Problem: Exact and Enumeration Algorithms. *Algorithmica*, 52(2):293–307, December 2007. doi:10.1007/s00453-007-9152-0.

19 Fedor V. Fomin, Fabrizio Grandoni, and Dieter Kratsch. Solving Connected Dominating Set Faster than 2^n. *Algorithmica*, 52(2):153–166, December 2007. doi:10.1007/s00453-007-9145-z.

20 Fedor V. Fomin, Pinar Heggernes, Dieter Kratsch, Charis Papadopoulos, and Yngve Villanger. Enumerating Minimal Subset Feedback Vertex Sets. *Algorithmica*, 69(1):216–231, December 2012. doi:10.1007/s00453-012-9731-6.

21 Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010.

22 Fedor V. Fomin, Ioan Todinca, and Yngve Villanger. Exact Algorithm for the Maximum Induced Planar Subgraph Problem. In *Algorithms – ESA 2011*, pages 287–298. Springer Berlin Heidelberg, 2011. doi:10.1007/978-3-642-23719-5_25.

23 Tim A. Hartmann. Target Set Selection Parameterized by Clique-Width and Maximum Threshold. In *SOFSEM 2018: Theory and Practice of Computer Science*, pages 137–149. Springer International Publishing, December 2017. doi:10.1007/978-3-319-73117-9_10.

24 David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '03*. ACM Press, 2003. doi:10.1145/956750.956769.

25 Dániel Marx. Parameterized graph separation problems. *Theoretical Computer Science*, 351(3):394–406, February 2006. doi:10.1016/j.tcs.2005.10.007.

26 André Nichterlein, Rolf Niedermeier, Johannes Uhlmann, and Mathias Weller. On tractable cases of Target Set Selection. *Social Network Analysis and Mining*, 3(2):233–256, May 2012. doi:10.1007/s13278-012-0067-7.

27 D. Peleg. Size bounds for dynamic monopolies. *Discrete Applied Mathematics*, 86(2-3):263–273, September 1998. doi:10.1016/s0166-218x(98)00043-2.

28 Marcin Pilipczuk and Michał Pilipczuk. Finding a Maximum Induced Degenerate Subgraph Faster Than 2^n. In *Parameterized and Exact Computation*, pages 3–12. Springer Berlin Heidelberg, 2012. doi:10.1007/978-3-642-33293-7_3.

29 Michał Pilipczuk. Exact Algorithms for Induced Subgraph Problems. In *Encyclopedia of Algorithms*, pages 1–5. Springer US, 2015. doi:10.1007/978-3-642-27848-8_520-1.

30 Igor Razgon. Computing Minimum Directed Feedback Vertex Set in $O^*(1.9977^n)$. In *Theoretical Computer Science*, pages 70–81. World Scientific, 2007.