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

Ivan Bliznets*   Danil Sagunov†

May 14, 2018

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 or all dual thresholds are constant.

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 $|S| \le k$ and if we initially activate S then eventually at least ℓ vertices of G becomes activated. The activation process is defined by the following two rules: (i) if a vertex becomes activated it

*St. Petersburg Department of Steklov Institute of Mathematics of the Russian Academy of Sciences

†Saint Petersburg Academic University of the Russian Academy of Sciences

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 problem.

Target Set Selection problem naturally arises in such areas as economy, sociology, medicine. Let us give an example of a scenario [1, 2] 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 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 [3, 4], P_3-Hull Number [5], r-Neighbour Bootstrap Percolation [6], monotone dynamic monopolies [7], a generalization of Perfect Target Set Selection on the case of oriented graphs is known as Chain Reaction Closure and t-Threshold Starting Set [8]. 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 [2] and cliquewidth [9] 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 [10]. 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 $\epsilon > 0$ [1].

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 $\mathcal{O}^*\left((2-\epsilon)^n\right)$ for some $\epsilon >$

Parameter	Thresholds	Result	Reference
Bandwidth b	general	$\mathcal{O}^*\left(b^{\mathcal{O}(b\log b)}\right)$	Chopin et al. [11]
Clique Cover Number c	general	NP-hard for $c = 2$	Chopin et al. [11]
Cliquewidth cw	constant	$\mathcal{O}^*\left((\mathrm{cw}\cdot t)^{\mathcal{O}(\mathrm{cw}\cdot t)}\right)$	Hartmann [9]
Cluster Editing Number ζ	general	$\mathcal{O}^*\left(16^{\zeta}\right)$	Nichterlein et al. [12]
Diameter d	general	NP-hard for $d = 2$	Nichterlein et al. [12]
Feedback Edge Set Number f	general	$\mathcal{O}^*\left(4^{f}\right)$	Nichterlein et al. [12]
Feedback Vertex Set Number	general	$W[1]$-hard	Ben-Zwi et al. [2]
Neighborhood Diversity nd	majority	$\mathcal{O}^*\left(\mathrm{nd}^{\mathcal{O}(\mathrm{nd})}\right)$	Dvořák et al. [13]
	general	$W[1]$-hard	Dvořák et al. [13]
Target Set Size k	general	$W[2]$-hard	Nichterlein et al. [12]
Treewidth w	constant	$\mathcal{O}^*\left(t^{\mathcal{O}(w\log w)}\right)$	Ben-Zwi et al. [2]
	majority	$W[1]$-hard	Chopin et al. [11]
Vertex Cover Number τ	general	$\mathcal{O}^*\left(2^{(2^{\tau}+1)\cdot\tau}\right)$	Nichterlein et al. [12]

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.

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 $\mathcal{O}^*\left((2-\epsilon)^n\right)$ for any $\epsilon > 0$.*

Another example of such question is the following hypothesis:

Hypothesis 2. *[14] 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 [15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25] 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 $\text{thr}(v)$, are bounded by some fixed constant and in case when the values of $\text{thr}(v) - \deg(v)$, so-called *dual thresholds*, are bounded by some fixed constant for any v. 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 all threshold values are at most two;
- $\mathcal{O}^*\left(1.98577^n\right)$ if all threshold values are at most three;
- $\mathcal{O}^*\left((2-\epsilon_d)^n\right)$ randomized time if for any $v \in V(G)$ we have $\text{thr}(v) \geq \deg(v) - d$.

Target Set Selection can be solved in

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

We also prove the following lower bounds.

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

- $\text{thr}(v) = 2$ for any $v \in V(G)$;
- $\text{thr}(v) - \deg(v) = 0$ for any $v \in V(G)$.

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 set of vertices of graph G and by $E(G)$ — the set of its edges. We let $n = |V(G)|, m = |E(G)|$. $N(v)$ denote the set of neighbours of vertex $v \in V(G)$, and $N[v] = N(v) \cup \{v\}$. $\Delta(G) = \max_{v \in V(G)} \deg(v)$ denote the maximum degree of G. By $G[F]$ we denote the subgraph of G induced by a set of its vertices F. Define by $\deg_F(v)$ the degree of v in subgraph $G[F]$.

For a graph G, threshold function thr and $X \subseteq V(G)$ we put $\mathcal{S}_0(X) = X$ and for any $i > 0$ 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 $(i-1)^{th}$ 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 are 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 call X a *perfect target set of* (G, thr), if it activates all vertices of G, i.e. $\mathcal{S}(X) = V(G)$.

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 $X \subset V(G)$ such that $|X| \leq k$ and $|\mathcal{S}(X)| \geq \ell$?

Solution X of Target Set Selection problem (G, thr) we call a *target set*.

By Perfect Target Set Selection we understand a special case of Target Set Selection with $\ell = n$.

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* $H(\alpha) = -\alpha \log_2(\alpha) - (1-\alpha)\log_2(1-\alpha)$.

2.2 Minimal partial vertex covers

Definition 1. *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. $T = \{uv \in E(G) : \{u, v\} \cap S \neq \emptyset\}$.*

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.

Theorem 1. *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 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 $\texttt{minimal\_pvcs}(G, V(G), \emptyset, \emptyset)$ 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 $\texttt{minimal\_pvcs}(G, F \setminus N[v], A \sqcup R, Z \sqcup (N[v] \setminus R))$. 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

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 ∃v : N[v] ⊆ F then
    foreach R ⊊ N[v] do
        minimal_pvcs(G, F \ N[v], A ⊔ R, Z ⊔ (N[v] \ R))
else
    foreach R ⊆ F do
        if A ⊔ R is a minimal partial vertex cover of G then
            output A ⊔ R
```

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

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 brute-force all possible variants of the intersection of minimal partial vertex cover S and 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 any 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:

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

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

3 Algorithms for bounded thresholds

3.1 Algorithm for thresholds bounded by fixed constant

In this subsection we prove the following theorem.

Theorem 2. *Let t be fixed constant. For* Target Set Selection *with all thresholds bounded by t there is a $\mathcal{O}^*((2 - \epsilon_t)^n)$-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 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 a set of vertices which are known to be in our target set, Z — vertices which are known to be not in the target set, F — the set of all other vertices (i.e. vertices about which 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 $\deg_F(v) \geq \operatorname{thr}(v)$ then arbitrarily choose a subset $T \subseteq N(v)$ such that $|T| = \operatorname{thr}(v)$ and branch on the following branches:*

1. *For each subset of vertices $S \subseteq T \cup \{v\}$ of size less than $\operatorname{thr}(v)$ 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 $\operatorname{thr}(v)$ 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 2. *If $|F| \leq \gamma n$, where γ is a constant to be chosen later, then simply brute-force over all possibilities of 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 $2^{\gamma n}(2^{t+1} - t - 1)^{\frac{(1-\gamma)n}{t+1}}$ (here and later $t = \max_{v \in V(G)} \operatorname{thr}(v)$) 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 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 brute-force over all $2^{|F|}$ subsets of F as F is too big. Instead of this direct approach (brute-force over all subsets of F) we consider several sub-branches. 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 $P_0, P_1, \ldots, P_n$ 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 1 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, just take C as a minimal T-partial vertex cover when B is a T-partial vertex cover. One can obtain C by removing vertices from B one by one while it remains a T-partial vertex cover. When no vertex can be removed, then, by definition, the vertices left form a minimal T-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 sub-branches 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).

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

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

We show that by proving that $\mathcal{S}_i(X) \cap N(v) = P_i \cap N(v)$. Note that $\mathcal{S}_i(X) \setminus B = P_i \setminus B$ by definition of P_i. It is enough to prove that $\mathcal{S}_i(X) \cap N(v) \cap B = N(v) \cap B = P_i \cap N(v) \cap B$. Since $v \notin B$, for any $uv \in E(G[F])$, uv covered by B is equivalent to $u \in B$. C covers the same edges in $G[F]$ as B does, and also $v \notin C$, hence $C \cap N(v) = B \cap N(v)$. Using this and $C \subseteq P_0 \subseteq P_i$, 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 $|N(v) \cap P_i| < \text{thr}(v)$, 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 $\text{thr}(v)$. Thus we branch into $\text{thr}(v) + 1$ sub-branches corresponding to the value of $\min\{|N(v) \cap B|, \text{thr}(v)\}$, 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 $\text{thr}(v)$ and figure out whether v becomes activated in the current round or not.

Note that once we branch on the value of $\min\{|N(v) \cap B|, \text{thr}(v)\}$ 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 sub-branches, 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 our algorithm:

1. Branching on the value of the minimal partial vertex cover C. By Theorem 1 there is at most $\mathcal{O}^*\left(2^{\omega_t|F|}\right)$ such branches.

2. Branching on the value of $dg(v) = \min\{|N(v) \cap B|, \text{thr}(v)\}$ with $v \in Z$. There is at most $(t+1)^{|Z|}$ such possibilities since $t \geq \min\{|N(v) \cap B|, \text{thr}(v)\} \geq 0$.

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 $\text{thr}(v) \leq t$. Hence we perform a branching of this type on at most $(t-1)|Z|$ vertices.

Hence, the total number of the branchings made, i.e. branches created, in stage II is at most $\mathcal{O}^*\left(2^{\omega_t|F|} \cdot (t+1)^{|Z|} \cdot 2^{(t-1)|Z|}\right)$.

```
Algorithm: is_activated(G, thr, A, Z, F, P_i, v)
Input: G, thr, A, Z, F as usual, P_i such that P_i \ B = S_i(A ⊔ B) \ B for
       some B, and a vertex v ∉ P_i.
Output: True, if v ∉ B and v ∈ S_{i+1}(A ⊔ B);
        False, if v ∉ B and v ∉ S_{i+1}(A ⊔ B);
        any answer, otherwise.

if |N(v) ∩ P_i| ≥ thr(v) then
  | return True
else if v ∈ F then
  | return False
m ⟵ |N(v) ∩ (P_i \ F)|
branch on the value of dg(v) = min {|N(v) ∩ B|, thr(v)}
m ⟵ m + dg(v)
foreach u ∈ P_i ∩ N(v) ∩ F do
  | branch on whether u ∈ B
  | if u ∉ B then
  |   | m ⟵ m + 1
return m ≥ thr(v)
```

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

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);
- 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 $|A' \sqcup B'| \le k$, $|P \cup A' \cup B'| \ge \ell$ and for each $i \in \{1, 2, \ldots, z\}$ the value $\min\{\mathrm{thr}(v_i), |N(v_i) \cap B|\}$ equals $dg(v_i)$.

In order to solve the obtained problem we employ dynamic programming. We create a table TS of size $f' \times \ell \times (t+1)^z$. $TS(i, p, d_1, d_2, \ldots, d_z)$ stores the smallest set $B'_2 \subseteq \{u_{i+1}, u_{i+2}, \ldots, u_{f'}\}$ such that $|\mathcal{S}(A' \sqcup B'_1 \sqcup B'_2)| \ge \ell$, where B'_1 is any subset of $\{u_1, u_2, \ldots u_i\}$ such that $|(B'_1 \cup P) \cap \{u_1, u_2, \ldots, u_i\}| = p$ (i.e. $A \sqcup B'_1 \sqcup B'_2$ activates exactly p vertices in $\{u_1, u_2, \ldots, u_i\}$, since $\mathcal{S}(A \sqcup B'_1 \sqcup B'_2) \cap \{u_1, u_2, \ldots, u_i\} \subseteq B'_1 \cup P$), and $\min\{\mathrm{thr}(v_j), |N(v_j) \cap ((A' \setminus A) \sqcup B'_1)|\} = d_j$ for any j. In other words, $TS(i, p, d_1, d_2, \ldots, d_z)$ stores one of the 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. One can easily show that the choice of B'_2 depends only on values $i, p, d_1, d_2, \ldots, d_z$.

Note that for some values in the TS table there is no appropriate value (there is no 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 any p with $p + |P \cap (V(G) \setminus F')| \ge \ell$ since it corresponds to that all vertices in F' were chosen so that they activate at least ℓ vertices and all dg constraints are satisfied. 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 with $f' - 1 \ge i \ge 0$. We can evaluate any $TS(i, \cdot, \cdot, \ldots, \cdot)$ if we have all $TS(i+1, \cdot, \cdot, \ldots, \cdot)$ evaluated, in polynomial time. 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,

$$\begin{aligned} TS(i, p, \langle d_j \rangle) = \min \big[& TS(i+1, p+1, \langle d_j^{i+1} \rangle) \cup \{u_{i+1}\}, \\ & TS(i+1, p + |P \cap \{u_{i+1}\}|, \langle d_j \rangle) \big]. \end{aligned} \quad (1)$$

Since $0 \le d_j \le dg(v_j)$ for any j, the TS table has $\mathcal{O}^*\left((t+1)^{|Z|}\right)$ fields. Each of the fields of the table is evaluated in polynomial time, so the desired B', hence B, is found in $\mathcal{O}^*\left((t+1)^{|Z|}\right)$ time for any branch fixed in stage II. Thus in stages II and III the algorithm solves the problem for any appropriate branch fixed in stage I, and these two stages together take

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

running time.

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, each single of that can be presented in $t+1$ variants. Recall that $0 \le d_j \le dg(v_j) \le t$, in other words, after each of $dg(v_j)$ is fixed in stage II, each of d_j can be presented in $dg(v_j)+1$ variants in stage III. That leads to an observation that 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. Hence, 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 $|Z| \le n - |F|$, the upper bound becomes

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

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 when 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

Theorem 3. Perfect Target Set Selection *with thresholds equal to two can be solved in* $\mathcal{O}^*(1.90345^n)$ *time.*

Proof. Let (G, thr) be a graph with thresholds, with $|V(G)| = n$ and all thresholds equal to two. For this case, we present an algorithm with $\mathcal{O}^*(1.90345^n)$ running time that finds a perfect target set of (G, thr) of minimum possible size.

We set $\gamma = 0.655984$. The algorithm consists of two parts. In the first part, the algorithm brute-forces over all possible subsets $X \subseteq V(G)$ of size at most $(1-\gamma)n$, in ascending order of their size. If the algorithm meets X that is a perfect target set, i.e. $\mathcal{S}(X) = V(G)$, then it outputs the set and stops. Otherwise, the algorithm runs its second part.

The second part of the algorithm is a recursive branching algorithm that maintains sets A, Z, F similarly to the algorithm in section 3.1. The branching algorithm consists of two reduction and two branching rules. Here, we reuse reduction rule 1 and branching rule 1 from the previous section. Additionally, we introduce the following rules.

Reduction rule 2. *If there is a vertex* $v \in F$ *with* $\deg_G(v) < 2$*, assign* v *to* A*.*

Reduction rule 2 is correct since such vertex cannot be activated other than being put in a target set.

Branching rule 3. *If there are two vertices $u, v \in F$ with $uv \in E(G)$ and $\deg_G(u) = \deg_G(v) = 2$, then consider three branches:*

- $u \in Z$, $v \in A$;
- $u \in A$, $v \in Z$;
- $u, v \in A$.

Branching rule 3 is correct since if none of u, v is in a target set, none of them will eventually have two activated neighbours and thus the target set is not perfect.

If none of the rules can be applied, the algorithm brute-forces all $2^{|F|}$ possibilities of how vertices in F should be assigned to A and Z. This finishes the description of the second part and the whole algorithm. We now give a bound on its running time.

By Lemma 1, the first part of the algorithm runs in $\mathcal{O}^*\left(2^{H(1-\gamma)n}\right) = \mathcal{O}^*\left(1.90345^n\right)$ time. If the algorithm does not stop in this part, then any perfect target set of G consists of at least $(1-\gamma)n$ vertices and the second part is performed.

Branching rules 1 and 3 give branching vectors $(3,3,3,3,3)$ (five variants are considered for three vertices) and $(2,2,2)$ (three variants are considered for two vertices) respectively, and the second vector gives bigger exponential branching factor equal to $\sqrt{3}$.

Observe that if branching rules 1, 3 and reduction rules 1, 2 cannot be applied, then $A \sqcup Z$ is in fact a perfect target set of G. Indeed, in that case $G[F]$ consists only of isolated vertices and isolated edges, as if there was a vertex $v \in F$ with $\deg_F(v) \geq 2$, branching rule 1 would be applied. Note that if some vertex $v \in F$ is isolated in $G[F]$, then it has at least $\deg(v) \geq \operatorname{thr}(v) = 2$ neighbours in $A \sqcup Z$, hence it becomes activated in the first round. Consider an isolated edge $uv \in G[F]$. Note that u and v cannot simultaneously have degree two in G, since branching rule 3 excludes this case. It means that either u or v has degree at least three and thus has at least two neighbours in $A \sqcup Z$. Hence, it becomes activated in the first round. Since the other vertex has at least one neighbour in $A \sqcup Z$, at the end of the first round it will have at least two activated neighbours, thus it becomes activated no later than the second round.

We conclude that if we need to brute-force over $2^{|F|}$ variants, then $A \sqcup Z$ is a perfect target set of G. Hence, $|A \sqcup Z| \geq (1-\gamma)n$ and $|F| \leq \gamma n$. It follows that the second part running time is at most $\sqrt{3}^{(1-\gamma)n} 2^{\gamma n} \cdot n^{\mathcal{O}(1)} = \mathcal{O}^*\left(1.90345^n\right)$. So, the running time of the whole algorithm is $\max\{2^{H(1-\gamma)n} \cdot n^{\mathcal{O}(1)}, \sqrt{3}^{(1-\gamma)n} 2^{\gamma n} \cdot n^{\mathcal{O}(1)}\} = \mathcal{O}^*\left(1.90345^n\right)$. □

Theorem 4. Perfect Target Set Selection *with thresholds equal to three can be solved in $\mathcal{O}^*\left(1.98577^n\right)$ time.*

Proof. Here, we adapt the algorithm working for thresholds equal to two to the case when all thresholds equal three. Let $\gamma = 0.839533$. At first, algorithm brute-forces over all subsets of size at most $(1 - \frac{2}{3}\gamma)n$ and stops if it finds a perfect target set among them. If the algorithm has not found a perfect target set on this step then we run a special branching algorithm.

As with thresholds equal to two we use branching rules 1, 3 and reduction rules 1, 2. The only difference is that now in reduction rule 2 and in branching rule 3 we use constant 3 instead of 2. We also introduce a new branching rule for this algorithm.

Branching rule 4. *Let $v \in F$, $u, w \in N(v) \cap F$ and $\deg_G(v) = 4, \deg_G(u) = \deg_G(w) = 3$. Consider all branches that split u, v, w between A and Z and assign at least one vertex to A.*

The rule is correct as we omit only one branch that put all three vertices u, v, w into Z. Note that if none of the vertices u, v, w is activated initially then none of them will become activated. Hence, this branch cannot generate any perfect target set.

We apply the above-stated rules exhaustively. When none of the rules can be applied we simply brute-force over all possible subsets of F. That is the whole algorithm. Now, it is left to bound the running time of the algorithm.

The first part runs in $\mathcal{O}^*\left(2^{H(1-\frac{2}{3}\gamma)n}\right) = \mathcal{O}^*\left(1.98577^n\right)$ time. If the algorithm does not stop after the first part then any perfect target set of G contains at least $(1 - \frac{2}{3}\gamma)n$ vertices. Branching rules 1, 3, 4 give the following branching factors respectively: $12^{\frac{1}{4}}$ (since 12 variants are considered for 4 vertices), $3^{\frac{1}{2}}$ (3 variants for 2 vertices) and $7^{\frac{1}{3}}$ (7 variants for 3 vertices). The biggest branching factor among them is $7^{\frac{1}{3}}$.

Now, we bound the size of F after exhaustive application of all rules.

Lemma 2. *After exhaustive application of all rules F consists of at most γn vertices.*

Proof. Consider values of A, Z, F when none of the branching rules can be applied. In this case we have that $\Delta(G[F]) < 3$.

Note that our graph does not contain perfect target sets of size at most $(1 - \frac{2}{3}\gamma)n$. Otherwise algorithm would have finished working on the first step when it was brute-forcing over all subsets of size at most $(1 - \frac{2}{3}\gamma)n$. Now, we start constructing a new perfect target set P based on that the rules cannot be applied to A, F, Z. Then, from the fact that $|P| > (1 - \frac{2}{3}\gamma)n$, we obtain that $|F| \leq \gamma n$.

Put $A \sqcup Z$ into a new perfect target set P. Let us show that degrees of vertices in the set $F' = F \setminus \mathcal{S}(A \sqcup Z)$ can only be three or four. If $v \in F$ and $\deg_G(v) \geq 5$, then v has at most two neighbours in F. Hence, it has at least three neighbours in $A \sqcup Z$ and so v is in $\mathcal{S}(A \sqcup Z)$.

Since $\Delta(G[F]) < 3$, $\Delta(G[F']) < 3$ also. Hence, any vertex $v \in F'$ with $\deg_G(v) = 4$ requires one more activated neighbour to become activated. Also, $G[F']$ consists only of isolated paths and cycles. Consider any isolated path in

$G[F']$. Observe that any of its endpoints cannot have degree four in G, since otherwise it would have at least three neighbours in $\mathcal{S}(A \sqcup Z)$ and would be activated. Hence, all endpoints of all isolated paths are vertices of degree three. Note that any endpoint has two activated neighbours. Since branching rule 2 cannot be applied, any two endpoints cannot be adjacent. Thus any isolated path in $G[F']$ consists of at least three vertices.

It means that the vertices that require two more activated neighbours to become activated are vertices of degree three that are not endpoints in any isolated path in $G[F']$. Note that if $u, v \in F'$ with $\deg_G(u) = \deg_G(v) = 3$ and u, v lie in the same isolated path or cycle Q in $G[F']$, then there is at least two vertices of degree four in Q between u and v, or otherwise one of branching rules 3 or 4 would be applied. Thus in any isolated path or cycle Q in $G[F']$ the number of vertices that require at least two activated neighbours to become activated constitute at most one-third of the length of Q. We put all such vertices in the set P. There may be isolated paths or cycles left in $G[F']$ from which we have not put any vertex into P. For each such path or cycle we choose an arbitrary vertex from it and put it into P. Note that from each isolated path or cycle in $G[F']$ we put no more than one-third of its vertices into P. Construction of P is finished.

From each isolated cycle or path we picked at least one vertex into P and vertices that left require only one activated neighbour to become activated. Hence, P activates the whole graph. The size of P is at most $|A \sqcup Z| + \frac{1}{3}|F'| \leq n - |F| + \frac{1}{3}|F| = n - \frac{2}{3}|F|$. It means that $n - \frac{2}{3}|F| \geq (1 - \frac{2}{3}\gamma)n$. Hence, we proved $|F| \leq \gamma n$. □

Using this lemma, we can bound the running time of the second part. The largest branching factor in the rules is $7^{\frac{1}{3}}$. Hence, the running time is at most $7^{\frac{1}{3}(1-\gamma)n} 2^{\gamma n} \cdot n^{\mathcal{O}(1)}$. Combining it with the running time of the first part we get that the overall running time is $\mathcal{O}^*(1.98577^n)$. □

3.3 Algorithm for thresholds bounded by one-third of degrees

Theorem 5. *Let G be a connected graph with at least three vertices. Assume that* $\mathrm{thr}(v) \leq \lceil \frac{\deg(v)}{3} \rceil$ *for any $v \in V(G)$. Then there is a perfect target set of* (G, thr) *of size at most* $0.45|V(G)|$.

Proof. We prove this fact by induction on the number of vertices n in G.

If G is connected and $|V(G)| = 3$ then any single vertex in G forms a perfect target set. This is true since $\Delta(G) \leq 2$ and thus the threshold value of any vertex of G does not exceed 1.

From now on G is a connected graph on n vertices with $n > 3$. Let n_1 be the number of vertices in G of degree one and $n_{\geq 2}$ be the number of vertices in G of degree at least two, $n_1 + n_{\geq 2} = n$.

If $n_1 > n_{\geq 2}$, then there exist vertices $v, u_1, u_2 \in V(G)$ such that $vu_1, vu_2 \in E(G)$, $\deg(u_1) = \deg(u_2) = 1$. Let $\rho(G, \mathrm{thr})$ be the size of minimum perfect target set of (G, thr). Then $\rho(G, \mathrm{thr}) \leq 1 + \rho(G', \mathrm{thr}')$, where $G' = G \setminus v$ and $\mathrm{thr}'(u) = \mathrm{thr}(u) - |N(u) \cap \{v\}|$ for any $u \in V(G')$. Note that $\mathrm{thr}'(u) \leq \lceil \frac{\deg_{G'}(u)}{3} \rceil$.

Let G' consist of k connected components $C_1, C_2, \ldots, C_k$, where $k \geq 3$, $C_1 = \{u_1\}$, $C_2 = \{u_2\}$ and $|C_i| \leq |C_{i+1}|$ for any $i \in \{1, 2, \ldots, k-1\}$. Then $\rho(G', \mathrm{thr}') = \sum_{i=1}^{k} \rho(G'[C_i], \mathrm{thr}')$. Observe that if $|C_i| \leq 2$, then $\rho(G'[C_i], \mathrm{thr}') = 0$. Indeed, if $C_i = \{u\}$, then $\mathrm{thr}'(u) \leq \deg_{G'}(u) = 0$, and u becomes activated in the first round. If $C_i = \{u, w\}$, then either $uv \in E(G)$ or $vw \in E(G)$, without loss of generality, say that $uv \in E(G)$. Also, $\deg_G(u), \deg_G(w) \leq 2$, thus $\mathrm{thr}(u)$ and $\mathrm{thr}(w)$ are not greater than one. Since $uv \in E(G)$, we have that $\mathrm{thr}'(u) = \mathrm{thr}(u) - 1 \leq 0$. Thus u becomes activated in the first round, and, as $\mathrm{thr}'(w) \leq \mathrm{thr}(w) \leq 1$, then w becomes activated no later than the second round. If $|C_i| \geq 3$, then, by induction, $\rho(G'[C_i], \mathrm{thr}') \leq 0.45|C_i|$.

Hence, $\rho(G', \mathrm{thr}') \leq \sum_{i=m+1}^{k} \rho(G'[C_i], \mathrm{thr}') \leq 0.45 \sum_{i=m+1}^{k} |C_i|$, where m is such that $|C_m| \leq 2$ and $|C_{m+1}| \geq 3$. Since $m \geq 2$, we have $\rho(G', \mathrm{thr}') \leq 0.45(|V(G')| - 2)$. This implies that $\rho(G, \mathrm{thr}) \leq 1 + 0.45(|V(G')| - 2) = 1 + 0.45(|V(G)| - 1 - 2) < 0.45|V(G)|$.

To handle the case $n_1 \leq n_{\geq 2}$ (equivalent to $2n_1 \leq n$) we use a combinatorial model proposed by Ackerman et al. in [26]. For each permutation σ of vertices $V(G)$ we construct a perfect target set in the following way. We put vertex v into the perfect target set if the number of neighbours to the left of v in the permutation σ is less than $\mathrm{thr}(v)$. It is easy to see that after such construction we get a perfect target set P_σ, as vertices will become activated from the left to the right. If we take a random permutation σ among all permutations then probability that a particular vertex v ends up in P_σ equals $\frac{\max\{0, \mathrm{thr}(v)\}}{\deg(v)+1}$. Since $\mathrm{thr}(v) \leq \lceil \frac{\deg(v)}{3} \rceil$, for a vertex of degree one the probability is bounded by $\frac{1}{2}$, for a vertex of degree two — by $\frac{1}{3}$, for a vertex of degree three — by $\frac{1}{4}$, for a vertex of degree four — by $\frac{2}{5}$, etc. Observe that the highest probability bounds are for vertices of degree one and four, thus the expected value of the perfect target set size of (G', thr') is bounded by

$$\frac{1}{2}n_1 + \frac{2}{5}n_{\geq 2} = \frac{1}{2}n_1 + \frac{2}{5}(n - n_1) = \frac{2}{5}n + \frac{1}{10}n_1 \leq \frac{2}{5}n + \frac{1}{10} \cdot \frac{1}{2}n = \frac{9}{20}n.$$

Hence, there is at least one perfect target set of (G, thr) of size at most $0.45n$. □

Corollary 1. Target Set Selection *with thresholds bounded by one-third of degree rounded up can be solved in* $\mathcal{O}^*(1.99001^n)$ *time.*

Proof. Let (G, thr) and k, ℓ be an instance of Target Set Selection with $|V(G)| = n$ and $\mathrm{thr}(v) \leq \lceil \frac{\deg(v)}{3} \rceil$ for any $v \in V(G)$. We are looking for $X \subseteq V(G)$ with $|X| \leq k$ and $|\mathcal{S}(X)| \geq \ell$.

Consider subgraph G' of G consisting of all connected components of G of size at least three. By Theorem 5, (G', thr) has a perfect target set of size at most

$0.45|V(G')| \leq 0.45n$, hence it is enough to consider such X that $|X \cap V(G')| \leq 0.45n$. We brute-force over all such variants of $|X \cap V(G')|$. By Lemma 1, it takes $\mathcal{O}^*\left(2^{H(0.45)n}\right) = \mathcal{O}^*\left(1.99001^n\right)$ time.

When $|X \cap V(G')|$ is fixed, it is left to consider connected components of G of size less than three. Note that if we already have $|X \cap V(G')| \leq k$ and $|\mathcal{S}(X \cap V(G'))| \geq \ell$, we may set $X = X \cap V(G')$ and stop. Otherwise, we should consider adding vertices from connected components of size one or two to X. Adding a vertex from a connected component of size one, i.e. isolated vertex, increases the number of activated vertices by one, and adding a vertex from a component of size two increases this number by two. Thus we greedily assign a single vertex from each component of size two to X, but no more than $k - |X \cap V(G')|$ in total. If after that the size of X is still less than k, we assign as many isolated vertices of G to X as we can. Then we finally check whether $|\mathcal{S}(X)| \geq \ell$.

The greedy part of the algorithm runs in polynomial time for each variant of $|X \cap V(G')|$. Hence, the whole algorithm runs in $\mathcal{O}^*\left(1.99001^n\right)$ time. $\square$

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 $\overline{\text{thr}}(v) = \deg(v) - \text{thr}(v)$. In terms of dual thresholds, v becomes activated if it has at most $\overline{\text{thr}}(v)$ not activated neighbours. For bounded dual thresholds we prove the following theorem.

Theorem 6. *For any non-negative integer d,* Perfect Target Set Selection *with dual thresholds bounded by d can be solved in $(2-\epsilon_d)^n \cdot n^{\mathcal{O}(1)}$ randomized time for some $\epsilon_d > 0$.*

Proof. In terms of dual thresholds, we can consider the activation process as a vertex deletion process, where activated vertices are deleted from the graph. With this consideration, activation process goes in the following way. Firstly, the target set is deleted from the graph. Then, in each consecutive round, a vertex v is deleted from the remaining graph if it has at most $\overline{\text{thr}}(v)$ neighbours remaining. When the process converges, vertices in the remaining graph are the vertices that are not activated. Thus the target set is perfect if and only if the remaining graph is empty.

If $\overline{\text{thr}}(v) = d$ for each $v \in V(G)$. Then, a vertex is deleted from the remaining graph if it has at most d neighbours remaining. By definition of d-degeneracy, a graph becomes empty after such process if and only if it is d-degenerate. Thus, a target set X is perfect if and only if $G \setminus X$ is d-degenerate. Hence, if all dual thresholds are equal to d, finding a maximum d-degenerate induced subgraph of G is equivalent to finding a minimum perfect target set of G.

In [16], Pilipczuk and Pilipczuk present an algorithm that solves Maximum Induced d-Degenerate Subgraph problem in randomized $(2-\epsilon_d)^n \cdot n^{\mathcal{O}(1)}$ time for some $\epsilon_d > 0$ for any fixed d. Hence, instances of Perfect Target Set Selection where all dual thresholds are equal to d can be solved in the same running time. Furthermore, one can show that this algorithm can be adjusted to work when all dual thresholds are not necessarily equal, but do not exceed d. □

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. 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 [3]. 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. Note that in their work they refer to the problem as IRR_2-CONVERSION SET.

Theorem 7. PERFECT TARGET SET SELECTION *cannot be solved in* $2^{o(n+m)}$ *time unless ETH fails.*

Proof. 3-BOUNDED-3-SAT is a version of 3-SAT with a restriction that each variable appears at most three times in a formula. It is a well-known fact that an instance of 3-SAT with n variables and m clauses can be transformed into an instance of 3-BOUNDED-3-SAT with $\mathcal{O}(m)$ variables and $\mathcal{O}(m)$ clauses, in polynomial time. Then, according to the Exponential-Time Hypothesis with Sparsification Lemma, it follows that 3-BOUNDED-3-SAT cannot be solved in $2^{o(n+m)}$ time.

In Theorem 2 in [3] Centeno et al. have shown how to reduce an instance of 3-BOUNDED-3-SAT to an instance of PTSS with thresholds equal to two in polynomial time. In this reduction, the number of vertices and edges of a resulting graph remain linear over the length of an initial formula. In other words, an instance of 3-BOUNDED-3-SAT with $\mathcal{O}(n)$ variables and $\mathcal{O}(m)$ clauses can be reduced to an instance of PTSS with $\mathcal{O}(n+m)$ vertices and thresholds equal to two in polynomial time. This implies that PTSS cannot be solved in $\mathcal{O}^*\left(2^{o(n+m)}\right)$ time. □

5.2 Parameterization by the required number of activated vertices

We now look at TARGET SET SELECTION as at a parameterized problem. In [12], Nichterlein et al. showed that PERFECT TARGET SET SELECTION is $W[2]$-hard when parameterized by the size of target set k. We consider the general case of TARGET SET SELECTION instead and take the required number of activated vertices ℓ as a parameter. It is natural to ask whether the problem remains intractable with this parameter as well. Note that ℓ may be significantly large than k. We show that the TARGET SET SELECTION is $W[1]$-hard even parameterized by ℓ even when all thresholds equal two or all dual thresholds are zero. We show the result by reduction from CLIQUE problem.

Theorem 8. TARGET SET SELECTION *parameterized by* ℓ *is* $W[1]$*-hard even if all dual thresholds are equal to 0.*

Proof. This problem is similar to the CUTTING ℓ VERTICES problem, where one is asked to delete at most k vertices from a graph so exactly ℓ vertices become separated from the remaining graph. In our problem we have an additional restriction that these vertices must form an independent set. Fortunately, the proof of $W[1]$-hardness of CUTTING ℓ VERTICES given by Marx [27] works even in this setting under a small adjustment. For completeness we provide the proof here again.

Let (G, k) be an instance of the CLIQUE problem. Construct a graph G' in which each vertex corresponds to a vertex or an edge of graph G i.e. $V(G') = V(G) \sqcup E(G)$. Construct a clique in G' on vertices corresponding to the vertices of G. Moreover, we add edges in G' between vertices corresponding to $v_i \in V(G)$ and $e_j \in E(G)$ if and only if v_i and e_j are incident in G. Consider an instance $(G', k, k+\binom{k}{2}, 0)$ of TARGET SET SELECTION with the same k, ℓ equal to $k+\binom{k}{2}$ and with all dual thresholds equal to $d = 0$.

Observe that if G has a clique of size k, then one can select vertices of this clique as a target set of size k in G'. Any of the $\binom{k}{2}$ vertices corresponding to the edges of the clique are connected only with the vertices in the target set. Hence, vertices corresponding to them become activated, thus giving at least $k + \binom{k}{2}$ activated vertices in total.

Let us show this in the other direction. We show that if G' has a target set of size k activating at least $k + \binom{k}{2}$ vertices than G contains a clique on k vertices. Note that any of $v_i \in V(G')$ can become activated only by being selected into a target set. Indeed, all $v_i \in V(G')$ form a clique and thus any of them requires at least $|V(G)| - 1$ vertices in G' to be selected into a target set to become activated. This implies that the $\binom{k}{2}$ vertices in G' that becomes activated correspond to some edges in G. Note that an edge requires both of its endpoints to be selected in the target set in order to be activated. The only way to activate $\binom{k}{2}$ edges by selecting no more than k vertices is to select exactly k vertices that form a clique in G. □

The following theorem provides a similar result with the usage of thresholds instead of dual thresholds.

Theorem 9. TARGET SET SELECTION *parameterized by* ℓ *is* $W[1]$*-hard even if maximum maximum threshold is two.*

Proof. Similarly, to Theorem 8 we construct a reduction from CLIQUE problem.

Let (G, k) be an instance of CLIQUE. We construct graph G' as in the proof of Theorem 8 with the only difference that vertices corresponding to vertices of G now form an independent set. We will refer to the vertex in G' corresponding to the edge $e \in E(G)$ as $v_e \in V(G')$. Similarly, 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 $(G', k, k + \binom{k}{2}, 2)$ of Target Set Selection with all thresholds equal to $t = 2$.

Again, 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 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 vertices of G and by $k_e = |S \cap E|$ the number of vertices in S corresponding to edges of E, $k_v + k_e = k$.

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 from 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 consider 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 build 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 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 $(G', k, k + \binom{k}{2}, 2)$ has a solution, then G has a clique of size k. $\square$

References

[1] 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.

[2] Oren Ben-Zwi, Danny Hermelin, Daniel Lokshtanov, and Ilan Newman. Treewidth governs the complexity of target set selection. *Discrete Optimization*, 8(1):87–96, 2011.

[3] 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, 2011.

[4] 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.

[5] 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, 2013.

[6] József Balogh, Béla Bollobás, and Robert Morris. Bootstrap percolation in high dimensions. *Combinatorics, Probability and Computing*, 19(5-6):643–692, 2010.

[7] D. Peleg. Size bounds for dynamic monopolies. *Discrete Applied Mathematics*, 86(2-3):263–273, 1998.

[8] 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, 1995.

[9] 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, 2017.

[10] Erika M.M. Coelho, Mitre C. Dourado, and Rudini M. Sampaio. Inapproximability results for graph convexity parameters. *Theoretical Computer Science*, 600:49–58, 2015.

[11] 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, 2013.

[12] 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, 2012.

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

[14] Michał Pilipczuk. Exact Algorithms for Induced Subgraph Problems. In *Encyclopedia of Algorithms*, pages 1–5. Springer US, 2015.

[15] 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.

[16] 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.

[17] Fedor V. Fomin, Fabrizio Grandoni, and Dieter Kratsch. Solving Connected Dominating Set Faster than 2^n. *Algorithmica*, 52(2):153–166, 2007.

[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, 2007.

[19] 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, 2013.

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

[21] 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.

[22] Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. Scheduling Partially Ordered Jobs Faster than 2^n. *Algorithmica*, 68(3):692–714, 2012.

[23] 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.

[24] 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, 2011.

[25] Fedor V. Fomin, Pinar Heggernes, Dieter Kratsch, Charis Papadopoulos, and Yngve Villanger. Enumerating Minimal Subset Feedback Vertex Sets. *Algorithmica*, 69(1):216–231, 2012.

[26] Eyal Ackerman, Oren Ben-Zwi, and Guy Wolfovitz. Combinatorial model and bounds for target set selection. *Theoretical Computer Science*, 411(44-46):4017–4022, 2010.

[27] Dániel Marx. Parameterized graph separation problems. *Theoretical Computer Science*, 351(3):394–406, 2006.