An Exact Almost Optimal Algorithm for Target Set Selection in Social Networks

[Extended Abstract]

Oren Ben-Zwi
Department of Computer Science
University of Haifa
Mount Carmel, Haifa 31905
Israel
nbenzv03@cs.haifa.ac.il

Danny Hermelin[*]
Department of Computer Science
University of Haifa
Mount Carmel, Haifa 31905
Israel
danny@cri.haifa.ac.il

Daniel Lokshtanov
Institutt for Informatikk
University of Bergen
PB 7803, N-5020 Bergen
Norway
daniello@ii.uib.no

Ilan Newman
Department of Computer Science
University of Haifa
Mount Carmel, Haifa 31905
Israel
ilan@cs.haifa.ac.il

ABSTRACT

The TARGET SET SELECTION problem proposed by Kempe, Kleinberg, and Tardos, gives a nice clean combinatorial formulation for many problems arising in economy, sociology, and medicine. Its input is a graph with vertex thresholds, the *social network*, and the goal is to find a subset of vertices, the *target set*, that "activates" a prespecified number of vertices in the graph. Activation of a vertex is defined via a so-called activation process as follows: Initially, all vertices in the target set become active. Then at each step i of the process, each vertex gets activated if the number of its active neighbors at iteration $i - 1$ exceeds its threshold. The activation process is "monotone" in the sense that once a vertex is activated, it remains active for the entire process.

Unsurprisingly perhaps, TARGET SET SELECTION is *NPC*. More surprising is the fact that both of its maximization and minimization variants turn out to be extremely hard to approximate, even for very restrictive special cases. The only known case for which the problem is known to have some sort of acceptable worst-case solution is the case where the given social network is a tree and the problem becomes polynomial-time solvable. In this paper, we attempt at extending this sparse landscape of tractable instances by considering the treewidth parameter of graphs. This parameter roughly measures the degree of tree-likeness of a given graph, *e.g.* the treewidth of a tree is 1, and has previously been used to tackle many classical *NPhard* problems in the literature.

Our contribution is twofold: First, we present an algorithm for TARGET SET SELECTION running in $n^{O(w)}$ time, for graphs with n vertices and treewidth bounded by w. The algorithm utilizes various combinatorial properties of the problem; drifting somewhat from standard dynamic-programming algorithms for small treewidth graphs. Also, it can be adopted to much more general settings, including the case of directed graphs, weighted edges, and weighted vertices. On the other hand, we also show that it is highly unlikely to find an $n^{o(\sqrt{w})}$ time algorithm for TARGET SET SELECTION, as this would imply a sub-exponential algorithm for all problems in *SNPclass*. Together with our upper bound result, this shows that the treewidth parameter determines the complexity of TARGET SET SELECTION to a large extent, and should be taken into consideration when tackling this problem in any scenario.

Categories and Subject Descriptors

F.2.m [**Theory of Computation**]: Anallysis of Algorithms and Problem Complexity—*Miscellaneous*; G.2.2 [**Mathematics of Computing**]: Discrete Mathmatics—*Graph Theory, Graph algorithms*

General Terms

Theory, Algorithms

Keywords

1. INTRODUCTION

Consider the following scenario: You are a marketing executive of a huge clothing company given the task of marketing a new line of summer wear. You have at hand a description of the relationship network formed among a sample of teenagers from the district. After some heavy thinking you come up with the following idea: You will identify, or *target*, key social figures of the network and persuade them into adopting the new summer line, by say, handing out substantial amounts of free samples. You then hope that by peer-pressure laws, the friends of those targeted individuals would be persuaded into buying the new products, which in turn will also cause their friends to be persuaded, and so forth, creating a domino-like effect in the network. But how do you find a good set of individuals to target?

Research in the area of *viral marketing* [8, 12, 17] studies questions similar to the one raised above. The key objects under research are *social networks* which are often modeled by graphs with individuals or organizations as vertices, and relationships or interactions as edges. Social networks play a leading role in many scientific fields, including most social sciences [16, 23, 24], life sciences [11, 32] and medicine [11, 23, 27]. In viral marketing, one attempts to take advantage of social network properties, in order to enhance revenue in various commercial applications. This is based on the premise that targeting a few key individuals may lead to strong Şword-of-mouthŤ effects, which in turn will cause a cascade of influence in the network. Viral marketing has recently become a widespread technique for promoting novel ideas, marketing new products, or spreading innovation [31, 19]. Today, in the age of the Internet, the huge amount of available data poses new challenges for this area which are both daunting and extremely profitable at the same time. As an example, MySpace, FaceBook, and Orkut, are just three of many social networking websites boasting more than a hundred million users world-wide; endlessly engaged in the exchange of news, opinions, gossip, and almost any other thinkable type of information.

One simple way to model the cascade of influence in viral marketing is given by the *threshold model*, see for example [18, 21] and references within. The main idea is to associate with each vertex v of the network two states, *active* and *inactive*, which indicate whether v is persuaded into adopting the idea or product that is marketed. Moreover, v is also assigned a *threshold value* $t(v)$, specifying how many neighboring vertices of v need to get persuaded before v itself is persuaded. A cascade of influence, or *activation process*, proceeds in the network as follows: Initially, all vertices are inactive. In phase 0 of the process, we select k initial vertices, the *target set*, that instantly become active. Then, at every phase $i > 0$, a vertex v becomes active if at least $t(v)$ of its neighbors were active in phase $i - 1$. Once a vertex becomes active, it remains active for the entire process. The process ends in phase $i_{\text{end}} < n$, where n is the number of vertices in the network, when no more vertices can get activated. Given the rules of this activation process, and knowledge of the thresholds in our network, which individuals should we target so as to persuade as many individuals in the network as possible?

The first to study this question from an algorithmic point of view were Kempe, Kleinberg, and Tardos in their seminal paper [20]. They investigated the following maximization problem: Given a social network G with vertex-thresholds, find a target set of size at most k that activates as many vertices in G as possible. This models the situation where there is a prespecified budget for targeting. We note that Kempe *et al.* focused mostly on the case where the thresholds of the graph are random. This work was extended in the work of [21, 26]. Chen [10] studied the following analogous minimization problem: Given a social network G, find a target set of smallest possible size that activates at least ℓ vertices of G. This models the case where we have a minimum limit for the number of persuaded individuals overall. The decision versions of these two problems coincide, and are the main focus of this paper. We refer to this decision problem throughout as the TARGET SET SELECTION problem.

Unsurprisingly perhaps, TARGET SET SELECTION is NP-complete. More surprising is the fact that both of its optimization variants turn out to be extremely hard to approximate, even for very restrictive special cases. Kempe, Kleinberg, and Tardos show that the maximization problem they introduced cannot be approximated within any non-trivial factor, unless P = NP, even when the given social network is bipartite with bounded degree, and all vertices have equal thresholds [20]. Chen [10] shows a polylogarithmic approximation lower bound for the minimization problem described above, and his bound also holds for bounded degree bipartite graphs, even when the thresholds are taken from the set $\{1, 2\}$. We also mention that both problems are W[P]-complete, *i.e.* fixed-parameter intractable, when parameterized by the size of the solution target set [1].

The high inapproximability results for the optimization versions of the TARGET SET SELECTION problem mentioned above are a striking blow from the algorithm designer point of view. In light of these results, we must turn our consideration towards special cases of the problem, or otherwise resort to heuristic approaches. When considering special cases, it is desirable to obtain a robust algorithm that behaves relatively well also on more general cases. Furthermore, one must overcome the fact that the problem is already known to be hard for many restrictive cases; in particular, for notoriously easy classes of graphs such as bounded degree graphs and bipartite graphs.

In this paper we tackle these difficulties by considering the *treewidth* parameter of graphs. This parameter plays an important role in the design of many exact and approximation algorithms for many NP-hard problems. The notion was introduced by Robertson and Seymour [30] in their celebrated proof of the Graph Minor Theorem. Roughly, it measures the degree in which the given graph is similar to a tree in a very deep structural sense. For instance, trees have treewidth 1. We will show that the treewidth parameter governs the complexity of the TARGET SET SELECTION problem in a very strict sense. The first clue for this was given by Chen [10] who showed that the problem is polynomial-time solvable in trees. We generalize this result substantially. Letting n and w respectively denote the number of vertices and treewidth of our input graph, we prove the following theorem:

THEOREM 1. TARGET SET SELECTION *can be solved in* $n^{O(w)}$ *time.*

The proof of this theorem involves an elaborate dynamic-programming algorithm which utilizes various combinatorial properties of the TARGET SET SELECTION problem; deviating somewhat from standard dynamic-programming algorithms for small treewidth graphs. It is worth pointing out that the time complexity of this algorithm can be rewritten as $T^{O(w)} \cdot n$, where T is the maximum threshold of any vertex in the network. Also, the algorithm can be adopted to much more general settings, including the case of directed graphs, weighted edges, and weighted vertices. We defer details of all of this to the complete version of the paper.

On the other hand, we will show that we cannot do much better than Theorem 1 above. We prove that, under a well-established complexity-theoretic assumption, the above algorithm is optimal up to a quadratic factor in the exponent dependency on w. This shows that the treewidth of the given network indeed determines to a large extent whether one can efficiently compute an optimal target set in the network. This, of course, does not rule out the possibility of other parameters with better bounds, but nevertheless gives an important insight to the true complexity of the problem. The second main result of this paper is given in the following theorem.

THEOREM 2. TARGET SET SELECTION *cannot be solved in* $n^{o(\sqrt{w})}$ *time unless all problems in* SNP *can be solved in sub-exponential time.*

The rest of the paper is devoted to proving both Theorem 1 and Theorem 2.

2. PRELIMINARIES AND MODEL DEFINITIONS

All graphs in this paper are simple and undirected, unless stated otherwise. For any graph G, we use $V(G)$ and $E(G)$ to respectively denote the vertices and edges of G. We will mostly use G to denote our input graph, or social network, that we will be working on, and we use n to denote the number of vertices in G, and $w - 1$ its treewidth (see definition below). We also assume we have at hand a *threshold function* $t : V(G) \to \mathbb{N}$ for the vertices of G. For a subset of vertices $X \subseteq V(G)$, we let $G[X]$ denote the subgraph of G induced by X. That is, the subgraph G' with $V(G') = X$ and $E(G') = \{\{u, v\} \in E(G) : u, v \in X\}$.

2.1 Model Definitions

Let S be any subset of vertices in G. An *activation process in G starting at S* is a chain of vertex subsets $Active[0] \subseteq Active[1] \subseteq \ldots \subseteq V(G)$, with $Active[0] = S$, and $Active[i]$ including all vertices u such that either $u \in Active[i-1]$, or $t(u) \leq |\{v \in Active[i-1] : \{u, v\} \in E(G)\}|$, for all $i > 0$. We say that v is activated at iteration i if $v \in Active[i] \setminus Active[i-1]$. We assume that the activation process terminates at iteration z, where z is the smallest index for which $Active[z] = Active[z+1]$. Clearly, $z < n$. We say that S *activates* $Active[z]$ in G. We now give a formal definition of the key social networking problem we will be working on in this paper:

TARGET SET SELECTION:

Instance: Two integers $k, \ell \in \mathbb{N}$, and a graph G with thresholds $t : V(G) \to \mathbb{N}$.

Goal: Find a subset $S \subseteq V(G)$ of size at most k that activates at least ℓ vertices in G.

There are many natural generalizations of the above formulation. First, one can consider directed graphs instead of undirected, where now the activation of a vertex is determined only by its incoming neighbors. Another natural generalization is obtained by adding weights to the vertices of the network, and asking for a target set of total weight not exceeding k. Finally, one can model the situation where different vertices have different influences on each other, by adding *influence values* to the edges of the network. In this case, a vertex gets activated in an activation process, if the sum of influence from all of its active neighbors exceeds its threshold. Note that the influence value does not have to be positive. Indeed in some cases a negative influence is considered.

All these generalizations are supported by our methods, sometimes with a minor change. That is, our upper bound stands even when the graph is directed, the vertices are weighted and the edges have influence values that are taken from $\mathbb{Z}$.

Another generalization is the *Non Monotone Model*, where at iteration i only a node, v, with more than $t(v)$ active neighbors on iteration $i - 1$ is active. In this model, the first question should be: does the process becomes stable? Some other interesting question

can be bounding the number of active nodes, bounding the threshold values given a certain bound on the number of active nodes, and so on. In this non monotone model, our methods fail to solve an instance, even if the graph is undirected and unweighted. A few similar models to this were investigated in several researches, including [25, 4, 28, 29].

Not all diffusion process schemes fall into the threshold model. For instance, in [14] a *Voter* model is investigated. In this iterative model, at any iteration every vertex chooses an action from its neighbors' actions to preform. In [5] there are *active* and *inactive* vertices, only the *active* vertices have colors to denote different product adopted. Each edge has an *activation probability* and the diffusion is made by *activation attempts* by *active* vertices to their neighbors selected by the *activation probability*. They give an *FPTAS* for the problem of maximizing the influence of a single player when the underlying graph is a tree, and claim that this *FPTAS* can be extended to bounded treewidth graphs.

2.2 TreeWidth

We next briefly discuss the treewidth parameter of graphs which plays a central role in this paper. There are many ways for defining the treewidth of a graph. We will use the original version by Robertson and Seymour [30] which uses an extremely handy form of graph decompositions, namely tree-decompositions:

Definition 1. (Tree Decomposition, Treewidth [30]) A *tree decomposition* of a graph G is a pair $(\mathcal{T}, \mathcal{X})$, where $\mathcal{X}$ is a family of subsets of $V(G)$, and $\mathcal{T}$ is a tree over $\mathcal{X}$, satisfying the following conditions:

1. $\bigcup_{X \in \mathcal{X}} G[X] = G$, and
2. $\forall v \in V(G) : \{X \in \mathcal{X} \mid v \in V(X)\}$ is connected in $\mathcal{T}$.

The *width* of $\mathcal{T}$ is $\max_{X \in \mathcal{X}} |V(X)| - 1$. The *treewidth* of G is the minimum width over all tree decompositions of G.

Arnborg *et al.* [3] showed how to compute a tree-decomposition of width w for an n-vertex graph with treewidth bounded by w in $n^{w+O(1)}$ time. This algorithm was later improved to linear-time for constant values of w by Bodlaender [6]. See also [2, 7, 22] for various approximation algorithms.

Given a tree decomposition $(\mathcal{T}, \mathcal{X})$ of G, we will assume that $\mathcal{T}$ is rooted at some arbitrary $R \in \mathcal{X}$. With this in place, there is an important one-to-one correspondence between subgraphs of G and nodes X in $\mathcal{T}$. For a node $X \in \mathcal{X}$, let $\mathcal{T}_X$ denote the subtree of $\mathcal{T}$ rooted at X, and let $\mathcal{X}_X$ denote the collection of nodes in this tree, including X itself. The subgraph G_X associated with X in $\mathcal{T}_X$ is defined by $G_X = \bigcup_{Y \in \mathcal{X}_X} G[Y]$. The vertices of X are called the *boundary* of G_X.

3. COMPUTING TARGET SETS FOR SMALL TREEWIDTH NETWORKS

In this section we present an $n^{O(w)}$ algorithm for TARGET SET SELECTION in graphs with treewidth bounded by w. In particular, we provide a proof for Theorem 1. To simplify the presentation, we will first assume that we are required to compute what we call a *perfect target set* for G, which is a set S that activates all vertices of the graph. That is, we assume we are given an instance of TARGET SET SELECTION with $\ell = n$. This simplifies many details necessary for our algorithm; however, the essence of the problem remains the same. Later in the section, we will explain how to extend our algorithm for general values of ℓ.

3.1 Algorithm blueprint

Our algorithm first constructs a tree-decomposition $(\mathcal{T}, \mathcal{X})$ for G. Then it traverses the tree $\mathcal{T}$ in this decomposition in bottom-up fashion, constructing solutions for the subgraph G_X corresponding to the current node $X \in \mathcal{X}$ it is visiting by combining solutions for subgraphs G_Y corresponding to the children Y of X in $\mathcal{T}$. We will actually be working with a more convenient type of compositions called *nice tree decompositions*.

Definition 2. (Nice Tree Decomposition) A tree decomposition $(\mathcal{T}, \mathcal{X})$ of a graph G is nice if $\mathcal{T}$ is rooted, binary, each node in $\mathcal{X}$ has exactly w vertices, and is of one of the following three types:

- Leaf nodes are leaves in $\mathcal{T}$, and consists of w pairwise non-adjacent vertices of G.
- Replace nodes $X \in \mathcal{X}$ have one child Y in $\mathcal{T}$, with $X \setminus Y = \{u\}$ and $Y \setminus X = \{v\}$ for some pair of distinct vertices $u \neq v \in V(G)$.
- Join nodes $X \in \mathcal{X}$ have two children Y and Z in $\mathcal{T}$ with $X = Y = Z$.

Given a tree decomposition of width $w - 1$ for G, one can obtain in linear time a nice tree decomposition for G with the same width and with $O(wn)$ nodes (see for instance [13]). We will assume from here on out that we have a nice tree decomposition $(\mathcal{T}, \mathcal{X})$ at hand, of width $w - 1$.

Let us begin the description of our algorithm by discussing the difficulties in applying the generic solution-combining treewidth paradigm mentioned above to TARGET SET SELECTION. Consider the subgraph G_X corresponding to some join node $X \in \mathcal{X}$ of our nice-tree decomposition, and let Y and Z be the two children of X in $\mathcal{T}$ with $X = Y = Z$. Suppose $S \subseteq V(G_X)$ is a perfect target set for G_X. When restricting the activation process of S in G_X only to the part of G_Y, a boundary vertex v may have less than $t(v)$ G_Y-neighbors active, before it itself gets activated. We know only that the total number of active G_Y- and G_Z-neighbors of v in G_X

is $t(v)$ or more. For this reason, we need to consider perfect target sets for G_Y that activate the boundary vertices according to many different threshold values. As it turns out, we only need to consider different threshold assignments to the boundary vertices; we can keep the original thresholds of all remaining vertices in the graph.

Definition 3. (Threshold Vector) Let G_X be a subgraph of G corresponding to a node X of $\mathcal{T}$, and let $[n]$ denote the interval of non-negative integers $\{0, 1, \ldots, n\}$. A *threshold vector*, $T \in [n]^w$, is a vector with a coordinate for each boundary vertex in X. Letting $T(v)$ denote the coordinate in T corresponding to the boundary vertex $v \in X$, and t denote the original threshold function of G, the subgraph $G_X(T)$ is defined as the graph G_X with thresholds:

- $T(v)$ for any boundary vertex $v \in X$, and

- $t(u)$ for all other vertices $u \notin X$.

Another difficulty is that when combining perfect target sets S_Y and S_Z of $G_Y(T_Y)$ and $G_Z(T_Z)$, we need to make sure that their combination actually constitutes a perfect target set in $G_X(T)$. There are many problems in this: First, we need to add up the threshold vectors at the boundary correctly, since there can be intersections in the G_Y- and G_Z-neighborhoods of boundary vertices. Also, more importantly, there can be dependencies in the activation processes, causing a deadlock in the combined process: For instance, a boundary vertex u might require another boundary vertex v to be activated in $G_Y(T_Y)$ before u itself can be activated, while the situation could be reversed in $G_Z(T_Z)$. To overcome these difficulties, we introduce the notion of activation orders, and activation processes constrained by activation orders.

Definition 4. (Activation Order) Let G_X be some subgraph of G corresponding to a node X of $\mathcal{T}$, and recall that $[w]$ denotes the interval of non-negative integers $\{0, 1, \ldots, w\}$. An *activation order* is a function $A : X \to [w]$, where for any $v \in X$, $A(v)$ represents the relative iteration in the boundary at which v is activated.

We now change the definition of the activation process on $G_X(T)$ given in Section 2 so that it is constrained by an activation order on the boundary of $G_X(T)$. Given a subset $S \subseteq V(G_X)$ and an activation order $A : X \to [w]$, the *A-constrained activation process of S in $G_X(T)$* is defined similarly to the normal activation process of S in $G_X(T)$, except that a boundary vertex gets active at iteration i only if all boundary vertices u with $A(u) < A(v)$ are active at iteration $i - 1$. This includes all boundary vertices selected in the target set. Note that S may activate in a constraint activation process

only a subset of the vertices it activates in the normal activation process. Nevertheless, it is clear that all vertices that are activated by S in a normal activation process get activated in an A-constrained process for some activation order A. A set of vertices which activates all vertices of $G_X(T)$ in an A-constrained activation process is said to be a *perfect target set conforming with A*.

We can now describe the information that our algorithm computes for each subgraph G_X corresponding to node X of $\mathcal{T}$. This information is stored in a table, which we denote by OPT_{G_X}, that is indexed by two types of objects:

- A threshold vector $T \in [n]^w$ corresponding to the thresholds of the boundary vertices of G_X.

- An activation order A which constrains the order of activation on the boundary vertices.

The entry $OPT_{G_X}[T, A]$ will store the smallest possible perfect target set $G_X(T)$ conforming with A.

LEMMA 1. *The number of different entries in OPT_{G_X} is bounded by $n^{O(w)}$.*

PROOF. We can bound the number of different threshold vectors and activation orders by $(n+1)^w$ and w^w respectively. Thus, the number of different entries is bounded by $(n+1)^w \cdot w^w = n^{O(w)}$. $\square$

Recall that $G_r = G$ when r is the root of $\mathcal{T}$. Therefore, if we compute the OPT_{G_r} table for the root r, we can determine the optimal perfect target set for G. Furthermore, according to the above lemma, and since $\mathcal{T}$ has $O(wn)$ nodes, to obtain our promised time bound all that is required is to compute the OPT_{G_X} table of any node X in time polynomial with respect to the total sizes of the OPT_{G_Y} tables of its children Y in $\mathcal{T}$. In the next section we give details on how to perform this computation in polynomial time, thus completing the proof for the case where we are required to activate all vertices of the graph.

3.2 Implementation
To complete the description of our algorithm, we need to show how to compute the OPT_{G_X} table corresponding to the current node $X \in \mathcal{X}$ we are visiting in $\mathcal{T}$, from the table(s) correspond to its child(ren) in $\mathcal{T}$.

Leaf Nodes:. The base cases for our algorithm are the leaf nodes, where computing OPT_{G_X} is easy. Indeed, if X is a leaf in $\mathcal{T}$, then G_X is a graph with w isolated vertices, and so any perfect target set for G_X must include all vertices with threshold greater than 0. Furthermore,

all vertices will get activated regardless of the activation order we impose on the boundary, and so we can compute OPT_{G_X} when X is a leaf node by:

$$OPT_{G_X}[T, A] = X \setminus \{v : T(v) = 0\} \qquad (1)$$

Replace Nodes:. Suppose X is a replace node with child Y in $\mathcal{T}$. That is, G_X is obtained by adding a new boundary vertex u to G_Y, and removing another boundary vertex v from the boundary (but not from G_X). By the second condition of Definition 1, u can only be adjacent to other boundary vertices of G_X. Let d denote the number of these neighbors of u in G_X, and assume that they are ordered. Also, let G_X^i, for $i = 0, \ldots, d$, denote the subgraph of G_X obtained by adding the edges between u and and all of its neighbors in X, up-to and including the ith neighbor. To compute OPT_X, we will actually compute $OPT_{G_X^i}$ in increasing values of i.

When $i = 0$, u is isolated, and thus it must be included in any perfect target set when it has threshold greater than 0. For any threshold vector T, let T^{uv} denote the threshold vector obtained by setting: $T^{uv}(w) = T(w)$ for all $w \neq u$, and $T^{uv}(u) = t(v)$. For an order A and a vertex u on the boundary, let A^{-u} be the set of all orderings which agree with A on all orders but u. I.e., an order belongs to A^{-u} if all the nodes but u share exactly the same order as in A. Now let:

$$\widetilde{A} = \operatorname{argmin}_{A' \in A^{-u}} |OPT_{G_Y}[T^{uv}, A']|$$

$$\widetilde{A_u} = \operatorname{argmin}_{A' \in A^{-u}} |OPT_{G_Y}[T^{uv}, A'] \cup \{u\}|$$

According to the above, when X is a replace node, we get for $i = 0$:

$$OPT_{G_X^0}[T, A] =$$

$$\begin{cases} OPT_{G_Y}[T^{uv}, \widetilde{A}] & \text{if } T(u) = 0 \\ OPT_{G_Y}[T^{uv}, \widetilde{A_u}] \cup \{u\} & \text{if } T(u) \neq 0 \end{cases} \qquad (2)$$

Now if $i > 0$, then G_X^i is obtained from G_X^{i-1} by connecting u to some boundary vertex $w \in X$. For any threshold vector T, let T^{u-} denote the threshold vector obtained by setting $T^{u-}(u) = \max\{T(u) - 1, 0\}$, and all remaining thresholds the same. Define T^{w-} similarly. We have:

$$OPT_{G_X^i}[T, A] =$$

$$\begin{cases} OPT_{G_X^{i-1}}[T, A] & \text{if } A(w) = A(u) \\ OPT_{G_X^{i-1}}[T^{u-}, A] & \text{if } A(w) < A(u) \\ OPT_{G_X^{i-1}}[T^{w-}, A] & \text{if } A(u) < A(w) \end{cases} \qquad (3)$$

Join Nodes:. We now turn to describe the computation at join nodes. Let X be a join node with children Y and Z in $\mathcal{T}$. Recall that G_Y and G_Z are two subgraphs

whose intersection is exactly their boundary $Y = Z$, and G_X is obtained by taking the union of these two subgraphs. For a boundary vertex $v \in X$, let $X(v)$ denote the set of boundary vertices that are connected to v in G_X. For $v \in X$, and an activation order A, we define A^{-v} to be the set of all boundary vertices u such that $A(u) < A(v)$. For two threshold vectors T_Y, T_Z and an order A, we define the threshold vector $T_Y \oplus_A T_Z$ as the vector T with $T(v) = T_Y(v) + T_Z(v) - |X(v) \cap A^{-v}|$ for every $v \in X$. We compute $OPT_{G_X}[T, A]$ using the following equation: First compute $\widetilde{T_Y}, \widetilde{T_Z}$ by:

$$(\widetilde{T_Y}, \widetilde{T_Z}) =$$

$$\operatorname{argmin}_{T_Y \oplus_A T_Z = T} |OPT_{G_Y}[T_Y, A] \cup OPT_{G_Z}[T_Z, A]|$$

Then using $\widetilde{T_Y}, \widetilde{T_Z}$ we can find $OPT_{G_X}[T, A]$ through the following:

$$OPT_{G_X}[T, A] = OPT_{G_Y}[\widetilde{T_Y}, A] \cup OPT_{G_Z}[\widetilde{T_Z}, A] \quad (4)$$

Correctness of the above equation is clear. Indeed, any perfect target set S for $G_X(T)$ which conforms with A can be decomposed into two subsets $S_Y = S \cap V(G_Y)$ and $S_Z = S \cap V(G_Z)$ which activate in an A-constrained activation process all vertices in $G_Y(T_Y)$ and $G_Z(T_Z)$, for some pair of threshold vectors T_Y, T_Z for which $T_Y \oplus_A T_Z = T$. The converse is also true; any pair of perfect target sets for $G_Y(T_Y)$ and $G_Z(T_Z)$ conforming with A can be united into a perfect target set for $G_X(T_Y \oplus_A T_Z)$, also conforming with A.

3.3 Summary and Generalizations

It is easy to see that using the equations given in Section 3.2 above, we can correctly compute the OPT_{G_X} table corresponding to a node X in $\mathcal{T}$, in time polynomial with respect to the total sizes of the tables of its children. According to Lemma 1, and since $|\mathcal{X}| = O(wn)$, this gives us a total running-time of $n^{O(w)}$, as promised by Theorem 1.

Note that while our algorithm solves the TARGET SET SELECTION problem in case the given social network is represented by undirected and unweighted graph, it is easy to see that the algorithm can also straightforwardly be extended to natural generalizations such as directed graphs or weighted vertices. Adding influence values to edges of the network is another generalization our algorithm supports, by modifying the definition of the $\oplus$ operation.

Observe that these three generalizations give an easy way to alter the algorithm from computing a perfect target set to any general target set. Given an input directed graph G which we are required to activate at least ℓ vertices in, we construct a directed graph G' by adding a new universal vertex v with weight ∞ and threshold ℓ that has an influence value of $t(u)$ on every vertex u in G, and every vertex u in G has influence value of 1 on v. Now clearly a subset of vertices $S \subseteq V(G)$ that activates at least ℓ vertices in G is a perfect

target set in G', and vice-versa, every perfect target set in G' with total weight less than ∞ activates at least ℓ vertices in G. Note also that the treewidth of G' differs by at most one from G's.

4. COMPUTATIONAL LOWER BOUND

In this section we present our lower-bounds for Target Set Selection in small treewidth graphs, and in particular, we provide a proof of Theorem 2. At the core of this proof is a theorem of Chen *et al.* [9] which shows a similar lower-bound for the Clique problem. Recall that Clique is the problem of finding a pairwise adjacent subset of k vertices in a graph with n vertices. Chen *et al.* proved the following lower-bound for Clique:

Theorem 3 ([9]). Clique *cannot be solved in* $n^{o(k)}$ *time unless all problems in* **SNP** *can be solved in sub-exponential time.*

We will show a reduction from Clique to Target Set Selection where the treewidth of the graph in the reduced instance is relatively close to the size of the clique to be searched for in the graph of the source instance. For this, we will actually use an intermediate problem, called the Multi-Colored Clique problem, where we are given a graph with vertices that are each colored by one of k different colors, and the goal is to find a clique of size k where all vertices have different colors.

Lemma 2. Multi-Colored Clique *cannot be solved in* $n^{o(k)}$ *time unless all problems in* **SNP** *can be solved in sub-exponential time.*

Proof. We reduce from Clique. Given an instance (G, k) for Clique, we construct a graph G' by taking k copies $v_1, \ldots, v_k$ of each vertex v of G, and then coloring each vertex v_i with color $i \in [k]' = \{1, 2, \ldots, k\}$. We then add an edge in G' between two vertices u_i and v_j, $i \neq j$, iff u and v are connected in G. It is straightforward to verify that G has a clique of size k iff G' has a multicolored clique. Therefore if Multi-Colored Clique can be solved in $n^{o(k)}$ time, then Clique can be solved in $(k \cdot n)^{o(k)} = n^{o(k)}$ time, implying by Theorem 3 that all **SNP** problems are solvable in sub-exponential time. $\square$

The approach for using Multi-Colored Clique in reductions is described in [15], and has been proven to be very useful in showing hardness results in the parameterized complexity setting. Before giving details of our construction, we will need to introduce some new terminology. We use G to denote a graph colored with k colors given in an instance of Multi-Colored Clique, and G' to denote the graph in the reduced instance of Target Set Selection. For a color $c \in [k]'$, we let V_c denote the subset of vertices in G colored with color c, and for a pair of distinct colors $c_1, c_2 \in [k]'$, we let $E_{\{c_1, c_2\}}$ denote the subset of edges in G with endpoints colored c_1 and c_2. In general, we use u and v for denoting arbitrary vertices in G, and x to denote an arbitrary vertex in G'.

Our construction constructs G' using two types of gadgets that guarantee any perfect target set of G' with a specific size encodes a multi-colored clique in G. These gadgets are the *selection* and *validation* gadgets. The selection gadgets encode the selection of k vertices and $\binom{k}{2}$ edges that together encode a vertex and edge set of some multi-colored clique in G. The selection gadgets also ensure that in fact k distinct vertices are chosen from k distinct color classes, and that $\binom{k}{2}$ distinct edges are chosen from $\binom{k}{2}$ distinct edge color classes. The validation gadgets validate the selection done in the selection gadgets in the sense that they make sure that the edges chosen are in fact incident to the selected vertices. In the following we sketch the construction of these gadgets:

- *Selection:* For each color-class $c \in [k]'$, and each pair of distinct colors $c_1, c_2 \in [k]'$, we construct a *c-selection gadget* and a *$\{c_1, c_2\}$-selection gadget* which respectively encode the selection of a vertex colored c and an edge colored $\{c_1, c_2\}$ in G. The c-selection gadget consists of a vertex x_v for every vertex $v \in V_c$, and likewise, the $\{c_1, c_2\}$-selection gadget consists of a vertex $x_{\{u,v\}}$ for every edge $\{u, v\} \in E_{\{c_1, c_2\}}$. There are no edges between the vertices of the selection gadgets, *i.e.* the union of all vertices in these gadgets is an independent set in G'. We next add a guard vertex at each (vertex and edge) selection gadget that is connected to all vertices in the gadget. In this way, a selection gadget is no more than a star centered at a guard vertex.

- *Validation:* We assign to every vertex v in G two unique identification numbers, $low(v)$ and $high(v)$, with $low(v) \in [n]'$ and $high(v) = 2n - low(v)$. For every pair of distinct colors $c_1, c_2 \in [k]'$, we construct validation gadgets between the $\{c_1, c_2\}$-selection gadget and the c_1-and c_2-selection gadget. Let c_1 and c_2 be any pair of distinct colors. We describe the validation gadget between the c_1 -and $\{c_1, c_2\}$-selection gadgets. It consists of two vertices, the *validation-pair* of this gadget. The first vertex of this pair is connected to each vertex x_v, $v \in V_{c_1}$, by $low(v)$ parallel edges, and to each edge-selection vertex $x_{\{u,v\}}$, $\{u, v\} \in E_{\{c_1, c_2\}}$ and $v \in V_{c_1}$, by $high(v)$ multiple edges. The other vertex is connected to each x_v, $v \in V_{c_1}$, by $high(v)$ multiple edges, and to each $x_{\{u,v\}}$, $\{u, v\} \in E_{\{c_1, c_2\}}$ and $v \in V_{c_1}$, by $low(v)$ multiple edges. We next subdivide the edges between the selection and validation gadgets to obtain a simple graph, where all new vertices introduced by the subdivision are referred to as the *connection vertices*.

To complete the construction, we specify the thresholds of the vertices in G'. First, all guard vertices have threshold 1. All selection vertices have thresholds equaling their degree in G'. Second, the connection vertices all have thresholds 1. Finally, the vertices in the validation pairs all have thresholds equaling $2n$. Figure 1 depicts a schematic description of selection and validation gadgets.
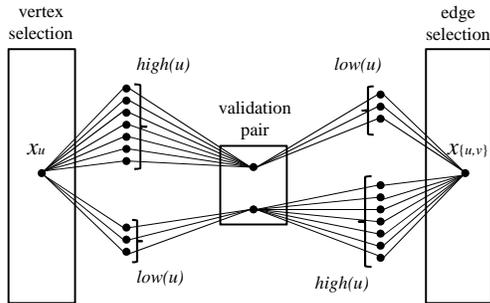
Figure 1: A graphical depiction of the validation gadget. In the example, $n = 5$ and $low(u) = 3$.

The main idea behind the validation gadgets is as follows: We bound the size of the required perfect target set, so that any solution must select at most one vertex from each selection gadget. When selecting from vertex and edge selection gadgets connected by a validation gadget, both vertices in the validation pair get active only if the vertex incident to that edge has been selected: This is because for any $u \neq v$ either $high(u) + low(v) < 2n$ or $low(u) + high(v) < 2n$. This allows us to state the following lemma:

LEMMA 3. *G has a k-multicolored clique iff G' has a perfect target set of size $k + \binom{k}{2}$.*

PROOF. For the easy direction of the lemma, suppose that K is a multi-colored clique in G of size k. Then we argue that the subset S of $k + \binom{k}{2}$ vertices, defined by

$$S = \{x_v : v \in K\} \cup \{x_{\{u,v\}} : u, v \in K\},$$

is a perfect target set for G'. Indeed, at the first iteration of the activation process of S in G, all guard vertices will be activated, since all of these have threshold 1, and each one has a neighbor in S. Furthermore, all connection vertices adjacent to vertices in S will also be activated. In the second iteration of the activation process all validation-pair vertices are activated, since each one has exactly $2n$ neighbors which are active, by construction. Finally, in the third iteration, all other connection vertices are activated, since all validation-pairs are active, which causes all remaining selection vertices to be activated in the fourth iteration.

For the converse direction, assume S is a perfect target set of size $k + \binom{k}{2}$ in G'. First observe that we can assume w.l.o.g. that S does not include any guard vertex,

since we can replace each guard vertex by an appropriate (edge-or vertex) selection vertex, and still activate G'. Furthermore, as guard vertices are connected only to selection vertices, there has to be at least one active vertex in each selection gadget, before all guards can be active. Since selection vertices not chosen in the target set of G' need their guards to be active before they can be activated, it follows that exactly one vertex from each selection gadget must be in any perfect target set S of size $k + \binom{k}{2}$ in G'. Finally, as discussed above, the only way to activate a validation pair between a vertex and edge selection gadget, is to select a pair of vertices corresponding to an incident vertex and edge pair in G. Thus all edges of G selected in the edge-selection gadgets of G', are incident to all vertices of G selected in the vertex selection gadgets of G', and thus S corresponds to a k-multicolored clique in G. $\square$

LEMMA 4. *G' has treewidth $O(k^2)$.*

PROOF. Removing all validation pairs in G' leaves a forest which has treewidth 1. Therefore, we can add all $O(k^2)$ vertices belonging to validation pairs to each node $X \in \mathcal{X}$ in a width 1 tree-decomposition of this forest, giving us a tree-decomposition of width $O(k^2)$ for G'. $\square$

According to the two lemmata above, we have shown a polynomial-time reduction that maps every instance (G, k) of CLIQUE to an instance (G', k') of TARGET SET SELECTION, $k' = k + \binom{k}{2}$, such that G has a multi-colored clique of size $k \iff G'$ has a perfect target set of size k', and G' has treewidth $O(k^2)$. Combining this with Lemma 2 completes the proof of Theorem 2. Indeed, if TARGET SET SELECTION has an $n^{o(\sqrt{w})}$ algorithm, where w is the treewidth of the input graph, then we could use the above reduction to map an instance (G, k) of MULTI-COLORED CLIQUE with $|G| = n$, to an instance (G', k') of TARGET SET SELECTION with $|G| = O(n^c)$, for a constant $c \in \mathbb{N}$, and $w = O(k^2)$, use this algorithm to determine whether G' has a perfect target set of size k', and according to this determine whether G has a multi-colored clique of size k. The running time of the entire procedure will be the running-time of the reduction which is polynomial in n and independent of k, plus the running-time of the presumed algorithm for TARGET SET SELECTION which is $(n^c)^{o(\sqrt{w})} = n^{o(k)}$. All together this gives us an $n^{o(k)}$ algorithm for MULTI-COLORED CLIQUE, which by Lemma 2 implies that all problems in SNP can be solved in sub-exponential time.

6. REFERENCES

[1] K. A. Abrahamson, R. G. Downey, and M. R. Fellows. Fixed parameter tractability and completeness iv: on completeness for w[p] and pspace analogues. *Annals Of Pure And Applied Logic*, 73:235Ü276, 1995.

[2] E. Amir. Efficient approximation for triangulation of minimum treewidth. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI-01)*, pages 7–15, 2001.

[3] S. Arnborg, D. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8(2):277–284, 1987.

[4] E. Berger. Dynamic monopolies of constant size. *J. Comb. Theory, Ser. B*, 83(2):191–200, 2001.

[5] S. Bharathi, D. Kempe, and M. Salek. Competitive influence maximization in social networks. In *WINE*, pages 306–311, 2007.

[6] H. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25:1305–1317, 1996.

[7] V. Bouchitté, D. Kratsch, H. Müller, and I. Todinca. On treewidth approximations. *Discrete Applied Mathematics*, 136(2-3), 2004.

[8] J. Brown and P. Reingen. Social ties and word-of-mouth referral behavior. *Journal of Consumer Research: An Interdisciplinary Quarterly*, 14(3):350–62, 1987.

[9] J. Chen, B. Chor, M. Fellows, X. Huang, D. Juedes, I. Kanj, and G. Xia. Tight lower bounds for certain parameterized NP-hard problems. In *Proc. of the 19th annual IEEE Conference on Computational Complexity (CCC)*, pages 150–160, 2004.

[10] N. Chen. On the approximability of influence in social networks. In *Proceedings of the 19th annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 1029–1037, 2008.

[11] Z. Dezső and A. Barabási. Halting viruses in scale-free networks. *Phys. Rev. E*, 65(5):055103, 2002.

[12] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the 7th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pages 57–66, 2001.

[13] R. Downey and M. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.

[14] E. Even-Dar and A. Shapira. A note on maximizing the spread of influence in social networks. In *WINE*, volume 4858 of *LNCS*, pages 281–286. Springer, 2007.

[15] M. Fellows, D. Hermelin, and F. Rosamond. On the parameterized complexity of multiple interval problems – Manuscript. 2008.

[16] L. C. Freeman. *The Development of Social Network Analysis: A Study in the Sociology of Science*. Vancouver, BC, Canada: Empirical Press, 2004.

[17] J. Goldenberg, B. Libai, and E. Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters*, pages 211–223, 2001.

[18] M. S. Granovetter. The strength of weak ties. *American Journal of Sociology, 78*, pages 1360–1380, 1973.

[19] M. Kearns and L. Ortiz. Algorithms for interdependent security games. In *Proceedings of the 17th Annual Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 288–297, 2003.

[20] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pages 137–146, 2003.

[21] D. Kempe, J. Kleinberg, and E. Tardos. Influential nodes in a diffusion model for social networks. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 1127–1138, 2005.

[22] T. Kloks and H. Bodlaender. Approximating treewidth and pathwidth of some classes of perfect graphs. In *Proceedings of the 3rd International Symposium on Algorithms And Computation (ISAAC)*, pages 116–125, 1992.

[23] R. T. Mikolajczyk and M. Kretzschmar. Collecting social contact data in the context of disease transmission: Prospective and retrospective study designs. *Social Networks*, 30(2):127–135, 2008.

[24] S. Milgram. The small world problem. *Psychology Today*, 2:60–67, 1967.

[25] S. Morris. Contagion. *The Review of Economic Studies*, 67(1):57–78, 2000.

[26] E. Mossel and S. Roch. On the submodularity of influence in social networks. In *Proceedings of the 39th annual ACM symposium on Theory of computing (STOC)*, pages 128–134, 2007.

[27] R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Phys. Rev. Lett.*, 86(14):3200–3203, 2001.

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

[29] D. Peleg. Local majorities, coalitions and monopolies in graphs: a review. *Theor. Comput. Sci.*, 282(2):231–257, 2002.

[30] N. Robertson and P. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *SIAM Journal of Algorithms*, 7:309–322, 1986.

[31] G. Silverman. *The secrets of word-of-mouth marketing*. AMACOM New York, 2001.

[32] D. S. Wilson. Levels of selection: An alternative to individualism in biology and the human sciences. *Social Networks*, 11(3):257–272, 1989.