Treewidth Governs the Complexity of Target Set Selection *

Oren Ben-Zwi† Danny Hermelin‡ Daniel Lokshtanov§ Ilan Newman¶

Abstract

In this paper we study the Target Set Selection problem proposed by Kempe, Kleinberg, and Tardos; a problem which gives a nice clean combinatorial formulation for many applications 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.

Our contribution is as follows: 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. This algorithm 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 SNP. 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. In the last part of the paper we also deal with the "non-monotone" variant of Target Set Selection, and show that this problem becomes #P-hard on graphs with edge weights.

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?

*This work appeared in *Proceedings of the 10th ACM Conference on Electronic Commerce (EC-2009)*, Stanford, California, USA, July 6–10, 355-362 2009

†Department of Computer Science, University of Haifa, Haifa 31905, Israel. nbenzv03@cs.haifa.ac.il.

‡Max-Planck-Institute for Informatics, 66123 Saarbru"cken, Germany. hermelin@mpi-inf.mpg.de.

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

¶Department of Computer Science, University of Haifa, Haifa 31905, Israel. ilan@cs.haifa.ac.il.

Research in the area of *viral marketing* [8, 12, 16] 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 [15, 22, 23], life sciences [11, 32] and medicine [11, 22, 26]. 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 word-of-mouth 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, 18]. 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, QZone, FaceBook, and MySpace, are just three of many social networking websites boasting more than five hundred million users world-wide (November 2009); 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* [17]. 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, and so in this sense the activation process is *monotone*. 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 [19]. 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 [20, 25]. Chen [10] studied the following related 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. We reduce these two optimization problems to a single search problem, which is the main focus of this paper. We refer to this search problem throughout as the Target Set Selection problem.

Unsurprisingly perhaps, the decision version of 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 [19]. 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 $\{1, 2\}$. Regarding the parameterized complexity status of the problem, Abrahamson, Downey, and Fellows show 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 and fixed-parameter intractability results for Target Set Selection 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 restricted 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 [29] in their celebrated proof of the Graph Minor Theorem. Roughly, it measures the extent a 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 1.1. Target Set Selection *can be solved in* $n^{O(w)}$ *time.*

It is worth pointing out that the time complexity in the theorem above can be rewritten as $t^{O(w)}n$, where t is the maximum threshold of any vertex in the network. Thus, the algorithm used in proving this theorem also shows that the problem is fixed-parameter tractable when parameterizing by both the treewidth and maximum degree of the graph. Also, this algorithm can be adopted to the more general setting of directed graphs with edge influences and vertex weights.

On the other hand, we will show that we cannot do much better than Theorem 1.1. 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 1.2. Target Set Selection *cannot be solved in* $n^{o(\sqrt{w})}$ *time unless all problems in* SNP *can be solved in sub-exponential time.*

In the last part of the paper we consider the Target Set Selection problem under a non-monotone activation model where vertices may get deactivated throughout the process. This problem has applications in several models of cellular automatons, *e.g.* Conway's Game of Life, and has been investigated by several researches in different contexts, see *e.g.* [4, 24, 27, 28]. As it turns out, Non-Monotone Target Set Selection is much more difficult than Target Set Selection. In particular, the algorithm of Theorem 1.1 above no longer applies, even if the graph is undirected and unweighted. We show that the variant where the graph is directed with edge-influences becomes #P-hard.

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, 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 $E(G') = \{\{u, v\} \in E(G) : u, v \in X\}$. For two graphs G' and G'', we let $G' \cup G$ denote the graph G with $V(G) := V(G') \cup V(G'')$ and $E(G) := E(G') \cup E(G'')$.

2.1 Model Definition

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. We also may allow influence values to be negative.

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 a slightly different definition from the original version by Robertson and Seymour [29] which uses an extremely handy form of graph decompositions, namely tree-decompositions:

Definition 2.1. *(Tree Decomposition, Treewidth [29]) 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. $\forall v \in V(G) : \{X \in \mathcal{X} \mid v \in X\}$ *is connected in* $\mathcal{T}$*.*

The width *of* $\mathcal{T}$ *is* $\max_{X \in \mathcal{X}} |X| - 1$*. 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 by Bodlaender [6] to a linear-time algorithm for constant values of w. See also [2, 7, 21] 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 Small Treewidth Networks

In this section we we provide a proof for Theorem 1.1 by presenting an $n^{O(w)}$ algorithm for Target Set Selection in graphs with treewidth bounded by w. To simplify matters, 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*, initially introduced in slightly different form by Bodlaender [5].

Definition 3.1. *(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 consist of* w *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 [5] and [13]). We will assume in the following 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 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 3.2. *(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 several problems with 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. 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 3.3. *(Activation Order) Let G_X be some subgraph of G corresponding to a node X of $\mathcal{T}$, and recall that $[w-1]$ denotes the interval of non-negative integers $\{0, 1, \ldots, w-1\}$. An* activation order *is a function $A : X \to [w-1]$, 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, 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 v becomes active at some iteration i only if all boundary vertices u with $A(u) < A(v)$ are active at iteration $i-1$, and only if all other boundary vertices w with $A(w) = A(v)$ will also become activate at this iteration. 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 in $G_X(T)$ conforming with the activation order A.

Lemma 3.4. *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 $(n+1)^w \cdot w^w = n^{O(w)}$. □

Recall that $G_X = G$ when X is the root of $\mathcal{T}$. Therefore, if we compute the OPT_{G_X} table for the root X, we can determine the optimal perfect target set for G. Our algorithm will compute the OPT_{G_X} tables in bottom-up fashion, where the computation at the leaves will be done by brute-force. According to Lemma 3.4 above, and since $\mathcal{T}$ has $O(wn)$ nodes, to obtain our promised time bound of Theorem 1.1 it suffices to OPT_{G_X} for any $X \in \mathcal{X}$ in $n^{O(w)}$ time. Since the graphs at the leaves only have w vertices, this can be done in $n^{O(w)}$ time for a leaf node X. The next section gives details on to compute OPT_{G_X} in case X is an internal node of $\mathcal{T}$.

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 $\mathcal{T}$. We recall that the computation of OPT_{G_X} is done by brute-force at a leaf $X \in \mathcal{X}$.

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 2.1, 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_{G_X}, we will actually compute $OPT_{G_X^i}$ in increasing values of i, letting $OPT_{G_X} := OPT_{G_X^d}$.

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 for X, let T^{uv} denote the threshold vector for Y obtained by setting: $T^{uv}(w) := T(w)$ for all $w \neq v$, and $T^{uv}(v) := T(u)$. For an order A for X, let A^{uv} denote the set of all orderings A' for Y with $A'(w) := A(w)$ for all boundary vertices $w \neq u, v$. Observe that we allow $A'(v) \neq A(u)$. According to the above, when X is a replace node we get for $i = 0$:

$$OPT_{G_X^0}[T, A] = \min_{A' \in A^{uv}} \begin{cases} OPT_{G_Y}[T^{uv}, A'] & \text{if } T(u) = 0, \\ OPT_{G_Y}[T^{uv}, A'] \cup \{u\} & \text{if } T(u) \neq 0. \end{cases} \quad (1)$$

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. Since the $\{u,w\}$ edge can only influence v if $A(w) < A(v)$, and vice-versa, we have:

$$OPT_{G_X^i}[T,A] = \begin{cases} OPT_{G_X^{i-1}}[T,A], & \text{if } A(w) = A(u), \\ OPT_{G_X^{i-1}}[T^{u-},A], & \text{if } A(w) < A(u), \\ OPT_{G_X^{i-1}}[T^{w-},A], & \text{if } A(u) < A(w). \end{cases} \tag{2}$$

Join Nodes: Let X be a join node with children Y and Z in $\mathcal{T}$. Due to the second condition of Definition 2.1, G_Y and G_Z are two subgraphs who share the same boundary vertices $Y = Z$, G_X is obtained by taking the union of these two subgraphs. Observe that this means that there are no edges between $V(G_Y) \setminus Y$ and $V(G_Z) \setminus Z$ in G_X. For a boundary vertex $v \in X$, let $N_{G[X]}(v)$ denote the set of boundary vertices that are connected to v in G_X. For $v \in X$, and an activation order A, let $A^{\leq v}$ be the set of all boundary vertices u such that $A(u) < A(v)$. Given an order A, and a pair of threshold threshold T_Y and T_Z, define the threshold vector $T_Y \oplus_A T_Z$ as the vector T where a coordinate $T(v)$ for $v \in X$ is defined by

$$T(v) := T_Y(v) + T_Z(v) - |N_{G[X]}(v) \cap A^{\leq v}|.$$

Observe that for a given activation order A, if $S_Y \subseteq V(G_Y)$ activates in an A-constrained activation process $T_Y(v)$ neighbors of v in G_Y, and $S_Z \subseteq V(G_Z)$ activates in an A-constrained activation process $T_Z(v)$ neighbors of v in G_Z, then $T(v)$ is exactly the number of neighbors of v activated by $S_Y \cup S_Z$ in an A-constrained activation process in G_X. This is because only boundary vertices w with $A(w) < A(v)$ will be active prior to v, and there are no edges between $V(G_Y) \setminus Y$ and $V(G_Z) \setminus Z$ in G_X. We thus can compute $OPT_{G_X}[T,A]$ using the following equation:

$$OPT_{G_X}[T,A] = \min_{T_Y \oplus_A T_Z = T} OPT_{G_Y}[T_Y,A] \cup OPT_{G_Z}[T_Z,A] \tag{3}$$

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 3.4, and since $|\mathcal{X}| = O(wn)$, this gives us a total running-time of $n^{O(w)}$, as promised by Theorem 1.1.

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 slightly altering the computation on the replace and join nodes of the tree decomposition.

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 the treewidth of G.

4 A 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 1.2. 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 4.1 ([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 4.2. 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 $i \in [k]$. 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 4.1 that all SNP problems are solvable in sub-exponential time. □

The approach for using Multi-Colored Clique in reductions is described in [14], 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'.

We construct G' using two types of gadgets. Our goal is to guarantee that 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 $high(v)$ parallel edges. The other vertex is connected to each x_v, $v \in V_{c_1}$, by $high(v)$ parallel edges, and to each $x_{\{u,v\}}$, $\{u, v\} \in E_{\{c_1,c_2\}}$ and $v \in V_{c_1}$, by $low(v)$ parallel 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 $2n$. Figure 1 depicts a schematic description of selection and validation gadgets.

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

Proof. Suppose that K is a multi-colored clique in G of size k. We argue that the subset S of $k + \binom{k}{2}$ vertices, defined by

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

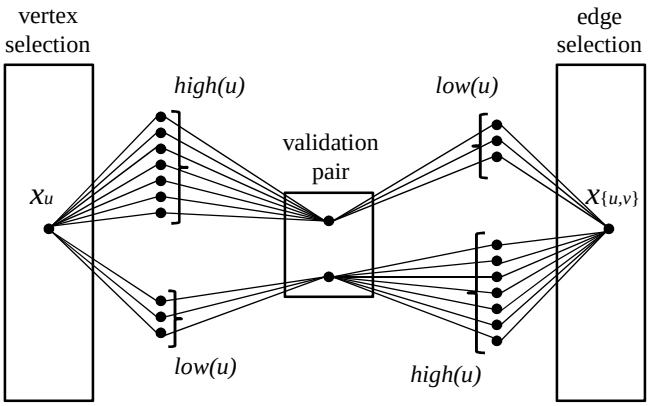

Figure 1: A graphical depiction of the validation gadget. In the example, $n = 5$ and $low(u) = 3$.

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. 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, without loss of generality, that S does not include any guard vertex. This is because we can replace each guard vertex by an appropriate 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. □

Lemma 4.4. *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'. □

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 4.2 completes the proof of Theorem 1.2. 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 4.2 implies that all problems in SNP can be solved in sub-exponential time.

5 A Non-Monotone Model

In this section we discuss the non-monotone variant of Target Set Selection. In Non-Monotone Target Set Selection, a vertex may become non-active in any iteration of the activation process once the total number of its active neighbors is smaller than its threshold. Thus, for example, the target set selected at the beginning of the process may get deactivated as the process continues. Formally, an activation process given a target set S is defined by a sequence of vertex subsets $Active[0], Active[1], \ldots$ which are no-longer necessarily a chain, where $Active[0] = S$, and $Active[i]$ for $i > 0$ is the set of all vertices u with $t(u) \leq |\{v \in Active[i-1] : \{u, v\} \in E(G)\}|$. A subset of vertices T is said to be activated by this process if $T \subseteq Active[i]$ for some i. The goal is thus to determine whether there exists a subset of k vertices that activates a subset of ℓ vertices in G. In what follows, the network G we consider is directed.

In the following we show that Non-Monotone Target Set Selection with edge influence values is #P-hard. Before this, let us first observe that the problem is in PSPACE. Consider the configuration graph C_G corresponding to G, which is a directed graph whose vertex-set is $2^{V(G)}$, and an edge (S, S') connects two subsets $S, S' \subseteq V(G)$ if in an activation process $Active[i] = S$ for some i, then $Active[i+1] = S'$. Explicitly storing this graph requires exponential space, but we can maintain an adjacency oracle (*i.e.* an algorithm outputting "yes" on input S and S' iff $(S, S') \in E(C_G)$) in polynomial-time and space. Now a non-deterministic algorithm can solve Non-Monotone Target Set Selection by guessing two vertex subsets $S, T \subseteq V(G)$, with $|S| = k$ and $|T| = \ell$, and then mimicking the PSPACE algorithm for S-T Connectivity on implicit graphs. Thus, Non-Monotone Target Set Selection is in NPSPACE, which is the same class as PSPACE due to Savitch's Theorem [30].

Theorem 5.1. Non-Monotone Target Set Selection *with edge influence values is* #P*-hard.*

Proof sketch. The proof follows by a reduction from the #P-complete problem #2-SAT, which asks to determine whether a 2-CNF formula φ has r satisfying assignments, for some $r \in \mathbb{N}$.

We say that a circuit C is *balanced* if the distance between any pair of input-output gates is the same. Before we explicitly describe our construction, we first show that given a balanced circuit C, we can construct a graph G and emulate the computation of C by an activation process on G. The graph G will be the graph isomorphic to the underlying graph of C, with vertex thresholds and edge influence values set as follows:

- If v corresponds to an input gate then we set its threshold to 1.
- If v corresponds a $\neg$-gate connected to a gate u, then we set $t(v) := -1$, and we let the influence value of the directed edge (u, v) be -2.

- If v corresponds to a $\vee$-gate connected to gates u_1 and u_2, we set $t(v) := 1$ and let the influences of (u_1, v) and (u_2, v) be 1.
- If v corresponds to a $\wedge$-gate connected to gates u_1 and u_2, we set $t(v) := 2$ and let the influences of (u_1, v) and (u_2, v) be 1.

Let $\{x_1, \ldots, x_n\}$ denote the input gates of C. It is clear that a truth assignment $\alpha : \{x_1, \ldots, x_n\} \to \{0, 1\}$ satisfies C iff the vertex corresponding to the output gate in G gets activated when the vertices corresponding to input gates x_i with $\alpha(x_i) = 1$ are selected in the target set. Thus, we can simulate the computation of any balanced circuit by an activation process in a graph G. In particular, we simulate a balanced circuit which computes the binary expansion of $f(x) := x + 1$ given the binary expansion of $x \in \mathbb{N}$ as input, and the balanced circuit which computes the binary expansion of $x + y$ given the binary expansion of x and y.

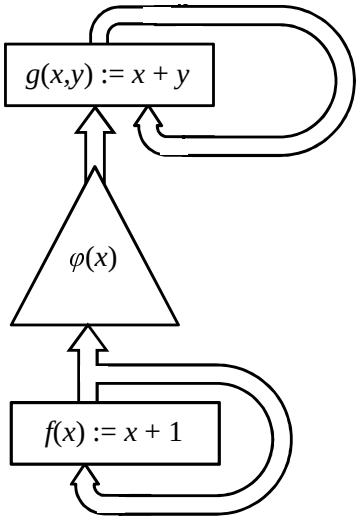

Figure 2: A schematic depiction of the way the circuits C_f, C_φ, and C_g are connected together.

Our construction works as follows (see Figure 2): We connect the outputs of a balanced circuit C_f computing $f(x) := x + 1$ back to its inputs, and also to the inputs of a balanced circuit C_φ computing φ. We connect the output of C_φ to the input of a circuit C_g computing $g(x, y) := x + y$. The output of C_φ is connected to the input corresponding to x in $g(x, y)$, and the outputs of C_g are connected to the inputs of C_g that correspond to y. In this way, C_f enumerates all assignments to C_φ, and C_g counts the number of these assignments that satisfy C_φ.

Note that there might by some synchronization issues when simulating C_f, C_φ, and C_g together. For instance, if C_f has depth (*i.e.* input-output distance) i, then we need to consider its output only at iterations i apart in the activation process. In this case, we can simply add a directed cycle of length i, with all vertex-thresholds and edge-influences set to 1, and connect one vertex of this cycle to the outputs of C_f by an $\wedge$-gate. We add similar synchronization gadgets for C_φ and C_g. Finally, to complete the construction, we add a gate u which has edges incoming from the outputs of C_g, whose influences are set in such a way so that u gets activated iff the output of C_g correspond to the binary expansion of r. We then connect u to another gate v that gets activated as soon as

u gets activated, and has outgoing edges to all other vertices with influences set in such a way so that they all get activated as soon as v is activated.

Let G denote the graph resulting from our construction. It is clear from our construction that φ has r satisfying assignments iff G has a target set of size 0 that activates all vertices in G. The theorem thus follows. □

6 Conclusions

In this paper we studied the Target Set Selection problem, a problem arising in viral marketing and other social and economic applications. We presented an algorithm running in $n^{O(w)}$ time for networks of size n and treewidth w, which also applies for various variants and generalizations of the problem. We also showed that this problem cannot be solved (under a natural complexity assumption) in time $n^{o(\sqrt{w})}$. Therefore, the time complexity needed to solve Target Set Selection is, in a sense, determined by the treewidth of the network. There are several open issues stemming from these two results. The following are three natural examples:

- Are there other parameters that govern the complexity of Target Set Selection?
- Can our lower bound extend to the pathwidth parameter of graphs?
- Can our upper and lower bounds be tightened?

For Non-Monotone Target Set Selection we showed that the most general case, where we have a directed network with edge influence values (which could be negative), is #P-hard and is thus much harder than the monotone problem. Note that our algorithm fails to solve even the most restrictive non-monotone variant where the graph is undirected and unweighted. We propose the following three questions:

- Is Non-Monotone Target Set Selection PSPACE-complete, or is it in #P?
- What is the complexity of the unweighted undirected variant of this problem?
- Is there a polynomial algorithm when the network is a tree?

Acknowledgments

We would like to thank Eyal Ackerman and Guy Wolfovitz for very fruitful discussions. In particular, Guy observed the simple reduction from computing a perfect target set to computing a regular one mentioned in Section 3.3. We also would like to thank David Kempe and an anonymous referee for pointing out some missing relevant references and for helpful remarks.

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*, 73:235–276, 1995.

[2] E. Amir. Efficient approximation for triangulation of minimum treewidth. In *Proceedings of the 17th conference on Uncertainty in Artificial Intelligence (UAI)*, pages 7–15, 2001.

[3] S. Arnborg, D.G. 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. *Journal of Combinatorial Theory, Series B*, 83(2):191–200, 2001.

[5] H.L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11:1–23, 1993.

[6] H.L. 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.J Brown and P.H. 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.W. Juedes, I. Kanj, and G. Xia. Tight lower bounds for certain parameterized NP-hard problems. In *Proceedings 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.L. Barabási. Halting viruses in scale-free networks. *Physical Review E*, 65(5), 2002.

[12] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the 7th ACM 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] M.R. Fellows, D. Hermelin, F.A. Rosamond, and S. Vialette. On the parameterized complexity of multiple interval problems. *Theoretical Computer Science*, 410(1):53–61, 2009.

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

[16] 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.

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

[18] 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.

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

[20] D. Kempe, J. Kleinberg, and É. 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.

[21] 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.

[22] 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.

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

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

[25] 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.

[26] R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Physics Review Letters*, 86(14):3200–3203, 2001.

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

[28] D. Peleg. Local majorities, coalitions and monopolies in graphs: a review. *Theoretical Computer Science*, 282(2):231–257, 2002.

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

[30] W.J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

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