https://doi.org/10.1016/j.tcs.2018.02.024

Active Influence Spreading in Social Networks*

Gennaro Cordasco
Department of Psychology
University of Campania "L.Vanvitelli", Italy.

Luisa Gargano
Department of Computer Science
University of Salerno, Italy.

Adele A. Rescigno
Department of Computer Science
University of Salerno, Italy.

December 11, 2017

Abstract

Identifying the most influential spreaders is an important issue for the study of the dynamics of information diffusion in complex networks. In this paper we analyze the following spreading model. Initially, a few nodes know a piece of information and are *active* spreaders of it. At subsequent rounds, spreaders communicate the information to their neighbors. Upon receiving the information, a node becomes *aware* of it but does not necessarily become a spreader; it starts spreading only if it gets the information from a sufficiently large number of its neighbors. A set of initial spreaders that guarantees that all the nodes become aware of the information is called a *perfect seed set*. We study the problem of choosing a perfect seed set of minimum size. We provide hardness results and show that the problem becomes tractable on trees. In case of general graphs, we provide an efficient algorithm and validate its effectiveness (in terms of the solution size) on real-life networks. We also study perfect seed sets in dense graphs and derive a bound on the size of a dominating set in Ore graphs.

*An extended abstract of this paper was presented at the 17th Italian Conference on Theoretical Computer Science (ICTCS 2016), [15]

1 Introduction

During the past decade the study of spreading processes in complex networks have experienced a particular surge of interest. A large part of the research activity in the area deals with the analysis of influence spreading in social networks. There are many situations where members of a network may influence their neighbors' behavior and decisions: By swaying their opinions, by suggesting what products to buy, or simply by passing on a misinformation [8, 33, 40]. A key research question, related to understand and control the spreading dynamics, is how to efficiently identify a set of users that can diffuse information within the network. This is the problem addressed in this paper. Our scenario posits a population consisting of a set of individuals that, with respect to the information, are subdivided into *ignorant, aware*, and *spreader*. Initially, all individuals are ignorant. Then an initial set of spreaders is selected. When a spreader informs an ignorant node v, the node v becomes aware; as soon as the individual v is informed by a number of spreaders equal or greater than a threshold $t(v)$, the node v starts spreading the information.
The motivations that lead to consider such a scenario come from experimental studies of how information spreads in social networks. Indeed, information doesn't flow freely in the network but it requires active sharing which, in turn, depends on individual conviction to pass it on. We refer to [3] for a study of how exposure to social signals affects diffusion.
Such a spreading model can be also seen as an idealization of the diffusion processes of memes. A *meme* [21] is a unit of cultural information, such as a cognitive or behavioral pattern, that can be transmitted from one individual to another one. Experimental observation about how memes evolve and spread within Facebook [2] suggests that our model makes a reasonable hypothesis on the meme spreading mechanism: An individual acquires a meme when it has been heard of from a friend, but people start spreading a meme only when it appears to be popular or important, i.e., when it has been heard of from several friends.

We model the network as an undirected graph $G = (V, E)$, where V is the set of individuals and the set of edges E represents the relationships among network members, i.e., $(u, v) \in E$ if individuals u and v can directly communicate. We denote by $N_G(v)$ the neighborhood of $v \in V$ in G.
Given a *threshold function* $t : V \to \{0, 1, 2, \ldots\}$, an *active diffusion process* starting at $S \subseteq V$ is a sequence of node subsets: $\mathsf{Spreader}_G[S, \tau]$, $\tau = 0, 1, \ldots$, such that

$$\mathsf{Spreader}_G[S, 0] = S,$$
$$\mathsf{Spreader}_G[S, \tau] = \mathsf{Spreader}_G[S, \tau - 1] \cup \Big\{u \,:\, \big|N(u) \cap \mathsf{Spreader}_G[S, \tau - 1]\big| \geq t(u)\Big\},\ \tau \geq 1.$$

In words, at each round $\tau \geq 1$, the set of spreaders is augmented with all the nodes u for which the number of neighbors that are already spreaders is at least equal to u's threshold $t(u)$. The process terminates when $\mathsf{Spreader}_G[S, \rho] = \mathsf{Spreader}_G[S, \rho - 1]$ for some $\rho > 1$. Hence, when the process stops, the sets of spreaders and of aware nodes are, respectively,

$$\begin{aligned} \mathsf{Spreader}_G[S] &= \mathsf{Spreader}_G[S, \rho] \\ \mathsf{Aware}_G[S] &= \mathsf{Spreader}_G[S] \cup \Big\{u \,:\, N(u) \cap \mathsf{Spreader}_G[S] \neq \emptyset\Big\}. \end{aligned}$$

We aim to identify a small node set S such that $\mathsf{Aware}_G[S] = V$.[1]

Definition 1 *Given a graph $G = (V, E)$, a set $S \subseteq V$ such that $\mathsf{Aware}[S] = V$ is called a* perfect seed set *for G. The nodes in S are denoted as* seeds.

Summarizing, we consider the following problem,

Perfect Awareness (PA).
Instance: A graph $G = (V, E)$, node thresholds $t : V \longrightarrow \mathbb{N}_0$.
Question: Find a perfect seed set $S \subseteq V$ of minimum size.

1.1 Related Work and Our Results

The problem of finding a perfect seed set of minimum size has its roots in the area of the *spread of influence* in Social Networks. Maximizing the spread of viral information across a network naturally suggests many interesting optimization problems (see [8, 23] and references quoted therein). The first authors to study spread of influence in networks from an algorithmic point of view were Kempe *et al.* [29, 30, 31]. Chen [7] studied the following minimization problem: given a graph G and fixed thresholds $t(v)$, for each node v in G, find a set of minimum size that eventually influences all (or a fixed fraction of) the nodes of G. This problem is usually referred as the Target Set Selection Problem (TSS). Chen proved a strong inapproximability result that makes unlikely the existence of an algorithm with approximation factor better than $O(2^{\log^{1-\epsilon}|V|})$. Chen's result stimulated a series of papers that isolated interesting cases in which the problem (and variants thereof) become tractable [1, 4, 6, 9, 10, 11, 12, 13, 17, 18, 19, 24, 37, 38]. Moreover, heuristics for the TSS problem that work for general graphs have been proposed in the literature [14, 22, 39].

However, the papers appeared in the scientific literature considered the basic model in which a node, as soon as it has enough influenced neighbors, immediately starts influencing its neighbors. The more refined model, considered in this paper, differentiates among spreaders

[1] In the rest of the paper we omit the subscript G whenever the graph is clear from the contex.

and plain aware nodes. It has been first considered in [16], where the authors studied the Awareness Maximization Problem which, having in input a graph G and a bound β, asks for a set of size at most β that achieves the maximum awareness in the network.

We first study the computational complexity of the Perfect Awareness problem and extend the TSS problem hardness result to the PA problem. In Section 5, we give an algorithm that outputs a perfect seed set for any input graph. Experimental evaluation of the proposed algorithm is given in Section 6; it shows that the proposed algorithm outperforms some heuristics developed for related problems. We would like to remark that if the threshold $t(v)$ is equal to the node degree $d(v)$, for each $v \in V$, then a perfect target set for G is, indeed, a dominating set for G. Hence, the proposed algorithm outputs a *dominating set* for G and computational experiments suggest that it performs very well in practice.

We also show, in Section 4, that the Perfect Awareness problem becomes tractable if the graph is a tree. Moreover, in Section 3 we study perfect seed sets in dense graphs; in this context, we derive a bound on the size of a dominating set in Ore graphs that can be of interest by its own.

2 Complexity

We prove the hardness of the PA problem by constructing a gap-preserving reduction from the TSS problem. We recall that the TSS problems, given $G = (V, E)$ with threshold function $t : V \to \mathbb{N}_0$, asks to identify a minimum size $S \subseteq V$ such that $\mathsf{Spreader}[S] = V$. Our Theorem 1 follows from the inapproximability results for the TSS problem given in [7].

Theorem 1 *The PA problem can not be approximated within a ratio of $O(2^{log^{1-\epsilon}n})$, for any $\epsilon > 0$, unless NP$\subseteq$DTIME($n^{polylog(n)}$).*

Proof. We give a reduction from the Target Set Selection problem. Consider an instance of the TSS problem consisting in a graph $G = (V, E)$ with threshold function $t(\cdot)$. Let $V = \{v_1, \ldots, v_n\}$, we build a graph $G' = (V', E')$ as follows:

- Replace each $v_i \in V$ by the gadget Λ_{v_i} (cfr. Fig. 1) in which the node set is $V'_i = \{v_{i,0}, v_{i,1}, v_{i,2}\}$. Formally,
  $-V' = \bigcup_{i=1}^{n} V'_i = \{v_{i,j} \mid 1 \leq i \leq n,\, 0 \leq j \leq 2\}$
  $-E' = \{(v_{i,0}, v_{\ell,0}) | \, 1 \leq i < \ell \leq n,\, (v_i, v_\ell) \in E\} \bigcup \{(v_{i,j}, v_{i,\ell}) | i = 1, \ldots, n,\; 0 \leq j < \ell \leq 2\}$;
- the thresholds are $t'(v_{i,0}) = t(v_i)$ and $t'(v_{i,1}) = t'(v_{i,2}) = 2$, for $i = 1, \ldots, n$.

Notice that G corresponds to the subgraph of G' induced by the set $\{v_{i,0} | 1 \leq i \leq n\}$. We show that there exists a target set $S \subseteq V$ for G iff there exists a perfect seed set $S' \subseteq V'$ for

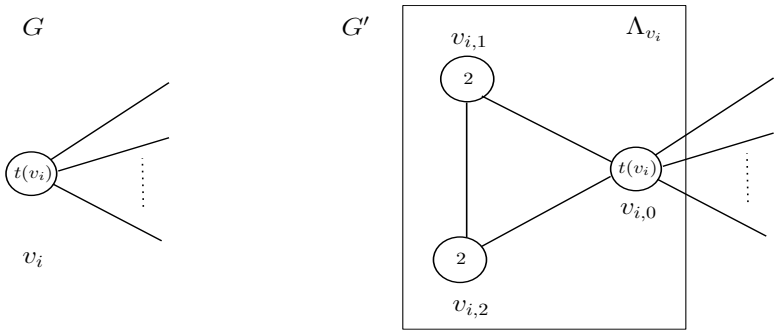


Figure 1: (a) A vertex $v_i \in V$ having threshold $t(v_i)$ in G. (b) The gadget Λ_{v_i} associated to v_i in G': It consists of a triangle where the node $v_{i,0}$ plays the role of v_i. The value inside each node represents the threshold of the node.

G' such that $|S'| = |S|$.
Assume first that $S \subseteq V$ is a target set for G. Since $\mathsf{Spreader}_G[S] = V$, then all the nodes $v_{i,0} \in V'_i$ will become spreaders in G' when the seed set is $S' = \{v_{i,0} \in V' | v_i \in S\}$. Once a node $v_{i,0}$ becomes a spreader the nodes $v_{i,1}, v_{i,2}$ are aware in the next round. Hence, S' is a perfect seed set for G', that is $\mathsf{Aware}_{G'}[S'] = V'$.
Assume now that $S' \subseteq V'$ is a perfect seed set for G'. Let $S'' = \{v_{i,0} \in V' \mid S' \cap V'_i \neq \emptyset\}$. It is easy to observe that $\mathsf{Aware}_{G'}[S''] = \mathsf{Aware}_{G'}[S'] = V'$. Let $V'_0 = \{v_{i,0} \mid 1 \leq i \leq n\}$. A node in V'_0 can make aware only 2 nodes in $V' - V'_0$—the other nodes of the triangle its belongs to. Hence, in order to make all the nodes in $V' - V'_0$ aware, all the nodes in V'_0 must be spreaders, that is, $\mathsf{Spreader}_{G'}[S''] = V'_0$. As a consequence, recalling that G is isomorphic to the subgraph of G' induced by V'_0, we get $\mathsf{Spreader}_G[\{v_i \mid S' \cap V'_i \neq \emptyset\}] = V$. ■

The Target Set Selection problem remains hard to approximate even when each node has threshold upper bounded by a constant; in particular, it was proved in [7] that approximating the problem when each node has threshold at most 2 is as hard as approximating it in the general setting, even for constant degree graphs. Our reduction allows to extend this result as well, namely one has that the PA problem remains hard to approximate even if all nodes have threshold at most 2.

Corollary 1 *Given any graph $G = (V, E)$, where $t(v) = 2$ (or $t(v) \leq 2$) for any node $v \in V$, the PA problem can not be approximated within the ratio of $O(2^{log^{1-\epsilon} n})$, for any $\epsilon > 0$, unless NP$\subseteq$DTIME($n^{polylog(n)}$).*

We also notice that if the threshold of each node is equal to the node degree then the problem becomes the dominating set problem, which is well-known to be hard to approximate [28].

Corollary 2 *Given any graph $G = (V, E)$, where $t(v) = d(v)$ for any node $v \in V$, the PA problem can not be approximated within the ratio of $(1 + o(1)) \ln \Delta$ (where Δ is the maximum degree of G), unless NP$\subseteq$DTIME($n^{polylog(n)}$).*

3 Dense Graphs.

In this section we study the problem of computing a perfect seed set for some graphs characterized by high edge density. We notice that real life social networks are characterized by the existence of highly connected communities and it was observed that in real networks, having high modularity [35], it is often difficult for information to flow from one community to another. This suggests that one should consider each (dense) community separately.

From the results in [16] and [25], we know that it is possible to relate the minimum graph degree to the size of a perfect seed set. Such results are summarized in the following theorems.

Theorem 2 [16] *Let $G = (V, E)$ be a graph with $t(v) \leq t$ and $d(v) \geq \frac{|V|+t-3}{2}$, for each $v \in V$. Any independent set which is either maximal or has size at least $2t - 2$ is a perfect seed set.*

Theorem 3 [25] *For any $t \geq 4$, let $G = (V, E)$ be a graph with $t(v) \leq t$ and $d(v) \geq \lfloor |V|/2 \rfloor + (t - 3)$, for each $v \in V$. There exists a perfect seed set for G of size t.*

Here, we concentrate on the problem of computing small perfect seed sets for a class of dense graphs known as Ore graphs.

Definition 2 *A graph G is an* Ore graph *if the sum of the degrees of any two independent nodes is at least $|V|$, that is, $|N(u)| + |N(v)| \geq |V|$ for all $u, v \in V$, such that $(v, u) \notin E$.*

We first give some results on dominating sets in Ore graphs and then we use such results to get perfect seed sets. We notice that any dominating set of a graph trivially is a perfect seed set for it (since it assures that any node becomes aware in one round). Moreover, in case all the nodes has threshold t (or $d(v)$, whichever is the smallest), then any perfect seed set of size less than t needs to be a dominating set of the graph (since no node outside the seed set can contribute to the spreading process).

The next property will be useful in the following.

Fact 1 *Any Ore graph has diameter two.*

Proof. Let u and v be any two independent nodes in the Ore graph $G = (V, E)$. By using Definition 2, we get

$$|N(u)| + |N(v)| \geq |V| \geq |N(u) \cup N(v) \cup \{u, v\}| = 2 + |N(u)| + |N(v)| - |N(u) \cap N(v)|$$

that leads to $N(u) \cap N(v) \neq \emptyset$ and the fact follows. ■

Some results are known about the size of a dominating set of a graph of diameter two. In particular, it is easy to see that the neighborhood of any node dominates G, if G has diameter two. Hence the next upper bound is immediate.

Fact 2 *[26] Let $G = (V, E)$ be a graph of minimum degree δ. If G has diameter two, then it has a dominating set of size at most δ.*

Hellwig and Volkmann [27] established an upper bound on the domination number of graphs of diameter two in terms of their order.

Theorem 4 *[27] If $G = (V, E)$ has diameter two, then it admits a dominating set of size at most $\lfloor |V|/4 \rfloor + 1$.*

In the next theorem we present an upper bound on the size of dominating sets in Ore graphs.

Theorem 5 (Dominating Sets in Ore Graphs) *Let $G = (V, E)$ be an Ore graph and let δ be the minimum degree of G. There exists a dominating set for G of size at most $3 + \left\lfloor \frac{\delta^2 - 5\delta + 6}{|V| - \delta} \right\rfloor$.*

Proof. Let v be a node with $d(v) = \delta$. For any $u \notin \{v\} \cup N(v)$, by using Definition 2, we get that

$$d(u) \geq |V| - d(v) = |V| - \delta.$$

Fix then any node $u \notin \{v\} \cup N(v)$ and let A_u denote the set of nodes that are neither neighbors of u nor of v, that is, $A_u = V - (\{v, u\} \cup N(u) \cup N(v))$. It is obvious that $\{v, u\}$ dominates all the nodes in V except for those in A_u. Hence, if $A_u = \emptyset$ then $\{u, v\}$ is a dominating set for G. Assume then that $A_u \neq \emptyset$. In the following we show how to augment the set $\{u, v\}$ so that also the nodes in A_u are dominated. In particular, we will prove that:

$$\text{There is } w \in V - (A_u \cup \{u, v\}) \text{ that has at least } |A_u| - \left\lfloor \frac{\delta^2 - 5\delta + 6}{|V| - \delta} \right\rfloor \text{ neighbors in } A_u. \quad (1)$$

This implies that the set $S_{u,w} = A_u - N(w)$ has size $|S_{u,w}| \leq \left\lfloor \frac{\delta^2 - 5\delta + 6}{|V| - \delta} \right\rfloor$ and $S = \{u, v, w\} \cup S_{u,w}$ is a dominating set for G.

We are then left to prove (1). To this aim, we first make some preliminary considerations. The number of nodes which are not neighbors of u are at most

$$|V| - 1 - d(u) \leq |V| - 1 - |V| + \delta = \delta - 1.$$

Since these $\delta - 1$ nodes include v, that is not a neighbor of u, we have $|A_u| \leq \delta - 2$.
Let $\sigma(A_u, V - A_u)$ be the number of edges connecting nodes in A_u with nodes in $V - A_u$.

Since the sum of the degrees of the nodes in the subgraph of G induced by A_u is at most $|A_u|(|A_u|-1)$ and since each $x \in A_u$ has $d(x) \geq |V| - \delta$, we get

$$\sigma(A_u, V - A_u) \geq |A_u|(|V| - \delta) - |A_u|(|A_u| - 1). \tag{2}$$

Furthermore, among the nodes in $V - A_u$ we know that both v and u are not neighbors of nodes in A_u. This implies that the neighbors of the nodes in A_u are at most $|V| - 2 - |A_u|$ (that is those in $V - (A_u \cup \{u, v\})$).

We prove now (1) by contradiction. Let $k = \left\lfloor \frac{\delta^2 - 5\delta + 6}{|V| - \delta} \right\rfloor$ and assume that each node in $V - (A_u \cup \{u, v\})$ has at most $|A_u| - k - 1$ neighbors in A_u. This assumption implies that $\sigma(A_u, V - A_u) \leq (|A_u| - k - 1)(|V| - 2 - |A_u|)$. As a consequence, by (2) we get

$$|A_u|(|V| - \delta) - |A_u|(|A_u| - 1) \leq (|A_u| - k - 1)(|V| - 2 - |A_u|).$$

Therefore,

$$(k+1)(|V| - 2) \leq |A_u|(k + \delta - 2).$$

Noticing that $k = \lfloor \frac{\delta^2 - 5\delta + 6}{|V| - \delta} \rfloor > \frac{\delta^2 - 5\delta + 6}{|V| - \delta} - 1$, we get

$$\delta - 2 < \frac{1}{2}\left(-k + \sqrt{k^2 + 4(k+1)(|V| - 2)}\right).$$

Recalling that $|A_u| \leq \delta - 2$, we then get

$$\begin{aligned}
(k+1)(|V| - 2) &\leq |A_u|(k + \delta - 2) \leq (\delta - 2)(k + \delta - 2) = (\delta - 2)^2 + k(\delta - 2) \\
&< \left(-\frac{k}{2} + \frac{1}{2}\sqrt{k^2 + 4(k+1)(|V| - 2)}\right)^2 \\
&\qquad + k\left(-\frac{k}{2} + \frac{1}{2}\sqrt{k^2 + 4(k+1)(|V| - 2)}\right) \\
&= (k+1)(|V| - 2),
\end{aligned}$$

leading to a contradiction. ∎

We notice that the size of the dominating set obtained in the proof of Theorem 5 is smaller than the bound δ in Fact 2 whenever G has minimum degree $\delta \leq |V|/2$. Moreover, if $\delta \leq \lfloor |V|/4 \rfloor + 1$ our bound strongly improves on the one in Theorem 4.

The above results allow to state the following upper bound on the perfect seed size of any Ore graph for any threshold function.

Corollary 3 *Let $G = (V, E)$ be an Ore graph and let δ be the minimum degree of G. For any threshold function $t : V \longrightarrow \mathbb{N}_0$, there exists a perfect seed set for G of size at most* $\min\left\{3 + \left\lfloor \frac{\delta^2 - 5\delta + 6}{|V| - \delta} \right\rfloor, \left\lfloor \frac{|V|}{4} \right\rfloor + 1,\ \delta \right\}$.

4 Trees.

In this section we give an algorithm that optimally solves the PA problem on trees.

Let $T = (V, E)$ be a tree rooted at any node r. The algorithm essentially computes the seed set while performing a visit (in BFS reverse order) of the tree, trying to procrastinate, as much as possible, the addition of nodes to the seed set. Once a node is elaborated, it is virtually pruned from the tree and the threshold of its parent is updated according to the choice made by the algorithm. Furthermore, in Case III (see Algorithm 1), in order to guarantee that the diffusion process will make all the nodes aware, the parent of the considered node must become a spreader. This information is maintained by adding the parent node to the set R.

Algorithm 1: TREE-PA(T, t), $T = (V, E)$ is a tree with thresholds $t(v)$ for $v \in V$

```
1  S = ∅;   A = ∅;   R = ∅;
2  foreach v ∈ V in a BFS reverse order do
3      if v ≠ r then                                          // v is not the root node
4          if t(v) = 0 then                                   // Case I
5              t(f_v) = t(f_v) − 1;                           // f_v denotes v's father
6              A = A ∪ {f_v}
7          else
8              if v ∈ R AND t(v) ≥ 2 then                     // Case II
9                  S = S ∪ {v};
10                 t(f_v) = t(f_v) − 1;
11                 A = A ∪ {f_v}
12             else                                           // Case III
13                 if v ∉ A OR (v ∈ R AND t(v) = 1) then
14                     R = R ∪ {f_v}                          // f_v must spread
15     else                                                   // v is the root node
16         if t(v) > 0 AND v ∉ A − R then S = S ∪ {v}
17 return S
```

Let T is the tree in Fig. 2. One can see that the algorithm TREE-PA(T, t) returns a optimal perfect seed set consisting of the three double-circled nodes in the figure.

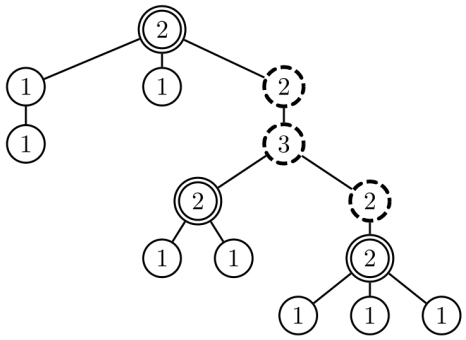


Figure 2: The integers inside circles are the node thresholds, double–lines denote seeds, dashed–lines denote aware nodes, solid–lines denote spreaders.

In order to show the optimality of the TREE-PA algorithm, in the following, we introduce a generalization of the PA problem.

> Perfect Awareness with Required Spreaders (PARS).
> **Instance:** A graph $G = (V, E)$, node thresholds $t : V \longrightarrow \mathbb{N}_0$ and required spreaders $R \subseteq V$.
> **Question:** Find a seed set $S \subseteq V$ of minimum size such that $\mathsf{Aware}[S] = V$ and $R \subseteq \mathsf{Spreader}[S]$.

We are going to show in Theorem 6 that the algorithm TREE-PA(T, t) is an optimal algorithm for the PARS problem. The optimality of TREE-PA(T, t) implies the optimality of the PA problem since at the beginning the set of required nodes is $R = \emptyset$.

We use the following notation. We denote by $n = |V|$ the number of nodes in T and by f_v the parent of node v with respect to the rooted tree T, for each $v \in V$. Moreover we denote:

- by v_i the node that is selected during the i-th iteration of the foreach in line 2 of Algorithm 1, for $i = 1, \ldots, n$;
- by $T(i)$ the tree induced by $V_i = \{v_n, v_{n-1}, \ldots, v_i\}$;
- by $t_i(v_i)$ the value of $t(v_i)$ as updated at the beginning of the ith iteration;
- by A_i the set of nodes aware thanks to the contribution of their children in T;
- by $R_i \subseteq V_i$ the set of nodes that during the diffusion process must become spreader.

We start with a technical Lemma.

Lemma 1 *Given a tree $T = (V, E)$, consider two instances of the PARS problem:*

- *$I = (T, t, R)$, consisting of T, a threshold function t, and a set $R \subseteq V$ and*

- $I' = (T, t', R')$ *where* $R' = R \cup \{v\}$ *and* t' *corresponds to* t *except for* $t'(v) = t(v) + 1$*, for some* $v \in V$*.*

Let S *and* S' *be two optimal solution for* I *and* I'*, respectively. We have that* $|S'| \leq |S \cup \{v\}|$*.*

Proof. It is easy to see that starting with S we can build a solution $S \cup \{v\}$ for I'. ■

Theorem 6 *PA*(T, t) *returns an* optimal *perfect seed set for any tree* T *in linear time.*

Proof. Thanks to the results in Theorem 7 we need to show only that the perfect seed set, identified by the algorithm TREE-PA(T, t) is optimal.

The basic observation is that each time a node $v \in V$ is not required to belong to the seed set (i.e., Case II of Algorithm 1 does not occur), we can discard the node without harming the optimality of the solution. Hence, we can assume without loss of generality that v is not added to the seed set, otherwise, we can add f_v instead of v and get a solution of the same size. When the node v is not added to the seed set, the algorithm checks whether, in order to guarantee v will become aware at the end of the diffusion process, it is mandatory that its parent f_v becomes a spreader (see line 13 of Algorithm 1).

Formally, we are going to show that the algorithm TREE-PA(T, t) is optimal for the PARS problem. The result will follow since the set R is initially empty.
We show the optimality of each choice made by the algorithm TREE-PA(T, t). Let v_i be the node considered during the i-th iteration of the foreach cycle in line 2 of Algorithm 1. Recall that any possible children of v_i has been already analyzed and virtually removed from the tree (that is v_i is a leaf of $T(i)$). Let S be the solution provided by TREE-PA and S^* be any optimal PARS solution.

First of all we show that if $v_i \in S$ then $v_i \in S^*$. Indeed v_i is added to S in Case II that is when $v_i \in R_i$ (i.e., v_i must become a spreader) and $t_i(v_i) \geq 2$. Since v_i is a leaf, the only way to make v_i a spreader is by adding it to the seed set.

On the other hand, if $v_i \notin S$ and $v_i \in S^*$, we can easily build another optimal solution by adding f_{v_i} instead of v_i to the set S^* and get a solution of at most the same size. Let $S' = S^* \cup \{f_{v_i}\} - v_i$, we can easily show that S' is a solution for the PARS problem. Since $v_i \notin S$, we have that Case II did not occur, that is, $v_i \notin R_i$ or $t(v_i) < 2$. In both cases f_{v_i} will be able to make v_i aware (if $v_i \notin R_i$) or spreader (if $t(v_i) < 2$). The selection of v_i as seed node may impact on R_{i+1} and $t_{i+1}(f_{v_i})$. Indeed, when $v \in S^*$ then $R_i = R_{i+1}$ and $t_{i+1}(f_{v_i}) = t_i(f_{v_i}) - 1$, while when $v \notin S^*$ we may have $R_{i+1} = R_i \cup \{f_{v_i}\}$ and $t_{i+1}(f_{v_i}) = t_i(f_{v_i})$. By Lemma 1, since $f_{v_i} \in S'$ we know that the optimal solutions for $I = (T(i+1), t_{i+1}, R_{i+1})$ and $I' = (T(i+1), t'_{i+1}, R'_i)$ have the same size and we can use S' instead of S^* without harming the optimality of the solution.

Hence, we can assume that if $v_i \notin S$ then $v_i \notin S^*$. The optimality of the algorithm is then guaranteed by the fact that it inserts a node in the set R_{i+1} only whenever it is required. Indeed the parent node f_{v_i} is added to R_{i+1} only when the node v_i is not made aware by the contribution of its children (i.e., $v_i \notin A_i$) or v_i must become a spreader and the residual threshold is 1 ($v_i \in T_i$ AND $t_i(v_i) = 1$) but this will only happen if f_{v_i} will become a spreader.

Finally, the root node $r = v_n$ is analyzed in line 16 of the algorithm as a specific case, considering that it has no parent nodes. In particular if $t_n(r) > 0$ and the root node is not aware thanks to the contribution of its children or must bust be a spreder ($r \notin A_n < R_n$) then r is added to the seed set and the same must happen for any other PARS solution. ■

5 A general algorithm for the PA problem

In this section we generalize the TREE-PA algorithm, given in Section 4, and propose an algorithm for the PA problem in case of arbitrary graphs and thresholds. The algorithm PA(G, t), given in Algorithm 2, works greedily by iteratively deprecating nodes from the input graph G unless a certain condition occurs which makes a node be added to the seed set S; it stops when all nodes have either been discarded or selected as seed. The algorithm maintains five sets of nodes:

- S that represents the current seed set;
- U that represents the set of nodes in the surviving graph (i.e., nodes not removed from the initial graph);
- $Temp$ which is a set of nodes moved into a *temporary waiting* state (such nodes still belong to U but their neighbors will not count on them for becoming spreaders);
- R that represents a set of nodes that must become spreaders (but will not do so with the current seed);
- A is the set of aware nodes (assuming that all the nodes in R will become spreaders).

Algorithm 2: PA(G, t) //$G = (V, E)$ is a graph with thresholds $t(v)$ for $v \in V$

```
1  S = ∅; Temp = ∅; U = V; R = ∅; A = ∅;
2  foreach v ∈ V do
3      k(v) = t(v);
4      δ(v) = |N(v)|;
5  while A ≠ V OR R ≠ ∅ do
6      if ∃v ∈ U s.t. k(v) = 0 then // Case 1): v is a spreader, thanks to its neighbors outside U
7          foreach u ∈ N(v) ∩ U do
8              k(u) = max(k(u) − 1, 0); A = A ∪ {u};
9              if v ∉ Temp then δ(u) = δ(u) − 1;
10         U = U − {v};   R = R − {v};   A = A ∪ {v};
11     else
12         if ∃v ∈ (U−Temp) ∩ R s.t. δ(v)<k(v) OR ∃v ∉ A s.t. δ(v) = 0 then
                                                  // Case 2): v must be a seed
13             S = S ∪ {v};
14             foreach u ∈ N(v) ∩ U do
15                 k(u) = k(u) − 1;
16                 δ(u) = δ(u) − 1;
17             U = U − {v};   R = R − {v};   A = A ∪ {v};
18         else
19             if U − Temp − R ≠ ∅ then   // Case 3): v is moved in the temporary repository
20                 v = argmin_{w∈U−Temp−R} {δ(w)}
21                 if v ∉ A then
22                     R = R ∪ {x} where x = argmax_{w∈N(v)∩(U−Temp)} {δ(w)}
23                     foreach z ∈ N(x) ∩ U do A = A ∪ {z};
24             else
25                 v = argmax_{w∈R} { k(w) / (δ(w)(δ(w)+1)) };
26             foreach u ∈ N(v) ∩ U do δ(u) = δ(u) − 1;
27             Temp = Temp ∪ {v};   R = R − {v};   A = A ∪ {v};
28 return S
```

The algorithm proceeds as follows: As long as there exists at least a node which is either non-aware or belongs to R, a node v is selected according to a certain function (see Case 3) and is moved into a *temporary waiting* state, represented by the set $Temp$. As a consequence of being in $Temp$, all the neighbors of v will not count on v for becoming spreaders (for each $u \in N(v)$ the value $\delta(u)$, which denotes the degree of u restricted to the nodes in the $U - Temp$, is reduced by 1).

Due to this update, some nodes in the surviving graph may remain with less "*usable*" neighbors (if a node $v \notin A$ has $\delta(v) = 0$ or $v \in R$ has $\delta(v) < t(v)$); in such a case (see Case 2) the nodes

are added to the seed set and removed from the graph, while the thresholds of their neighbors are decreased by 1 (since they have one more spreader neighbor). If (see Case 1) the surviving graph contains a node v whose threshold has been decreased down to 0 (which means that the nodes which have been already added to the seed set S (see Case 2) suffice to make v a spreader), v is deleted from the graph and the thresholds of its neighbors are decreased by 1 (since once v becomes a spreader, they have one more spreader neighbor). Notice that Case 1 can also apply to nodes in $Temp$. In such a case the value $\delta(\cdot)$ of the neighbors of the selected node v were already reduced by 1 (when v was moved to $Temp$ (and, therefore, it is not reduced further. By construction, once a node is moved to $Temp$, then it will be removed from the graph only by Case 1; indeed, Cases 2 and 3 only apply to nodes outside $Temp$. In other words, once a node is moved to $Temp$, it will not belong to the seed set.
When Case 3 applies the idea is to identify nodes that will never belong to the initial seed set. Two cases are considered, if the surviving graph still contains nodes which do not belong to the set R, then one of such nodes having minimum $\delta(\cdot)$ is moved to the set $Temp$. Otherwise all the nodes in the surviving graph must spread and the choice of the node to be deprecated is made according to a metric first studied in [20]. We notice that the metric used to choose which node to deprecate, that is to pose in the temporary repository when Case 3 applies, does not influence the correctness of the algorithm but it is the hearth for its effectiveness in terms of solution size.

Example 1 *Let G be a complete graph, the algorithm PA(G, t) optimally returns a single seed whichever the threshold function is: At the first iteration of the while loop, Case 3 applies and a node v_1 is selected; then a node v_2 is added to R (that is v_2 must become a spreader), while all the others (being neighbors of v_2) are marked aware; during the successive iterations, $|V| - t(v_2) - 1$ nodes are removed from U; finally Case 2 holds for v_2 which is added to S and the algorithm returns $S = \{v_2\}$.*

In the rest of the paper, we use the following notation. We denote by n the number of nodes in G, that is, $n = |V|$ and by λ the number of iterations of the while loop of algorithm PA(G, t). Given a subset $V' \subseteq V$ of nodes of G, we denote by $G[V']$ the subgraph of G induced by nodes in V'. Moreover, with respect to the iterations of the while loop in PA(G, t), for each $i = 1, \ldots, \lambda$ we denote:

- by $U_i, Temp_i, R_i, A_i, \delta_i(u)$, and $k_i(u)$, the sets $U, Temp, R, A$ and the values of $\delta(u), k(u)$, respectively, as updated at the beginning of the i-th iteration;
- by S_i the set of seeds selected by the algorithm from the i-th iteration until and including the λ-th iteration.

- by v_i the node selected during the i-th iteration.

When $i = 1$, the above values are those of the input graph G, that is: $U_1 = V$, $G[U_1] = G$, $S_1 = S$, $\delta_1(v) = |N(v)|$ and $k_1(v) = t(v)$, for each node v of G.

The following two facts highlight some properties that will be useful for the algorithm analysis. In particular, Fact 3 assures that at any iteration of algorithm PA, all the nodes removed from the initial graph are aware nodes (see 1.), as well as the nodes that are in the temporary waiting state (see 2.); moreover, the nodes that must become spreaders belong to the surviving graph but are not in the temporary waiting state (see 3.). The latter Fact 4 says that at any iteration of algorithm PA, the set of "usable" neighbors of any node in the surviving graph does not include the neighbors that are in the temporary waiting state.

Fact 3 *For each iteration i of the while loop in PA(G, t), the following relations hold:*

1. $V - U_i \subseteq A_i$

2. $Temp_i \subseteq A_i$

3. $R_i \subseteq U_i - Temp_i$.

Fact 4 *For each iteration i of the while loop in PA(G, t) and $u \in U_i$, it holds*

$$\delta_i(u) = |N(u) \cap (U_i - Temp_i)|.$$

Lemma 2 *Algorithm PA(G, t) executes at most $2n$ iterations of the while loop (i.e., $\lambda \leq 2n$).*

Proof. First of all we see that, during each iteration $i \geq 1$ of the while loop of PA(G, t), one node $v_i \in U_i$ is selected. If $R_i = \emptyset$ then $A_i \neq V$ (otherwise the algorithm terminates). Since by 1. of Fact 3 it holds $V - U_i \subseteq A_i$, we have that there exist $u \in U_i$ such that $u \notin A_i$. Then using 2. of Fact 3 we have that $u \notin Temp_i$ and consequently $U_i - Temp_i - R_i \neq \emptyset$. Hence a node is selected by Case 1 or by Case 2 or at line 20 of the algorithm. Otherwise ($R_i \neq \emptyset$) and a node is selected by Case 1 or by Case 2 or at line 25 of the algorithm. We conclude the proof noticing each $v \in V$ can be selected at most twice: Once v can be eventually added to $Temp$ (if Case 3 holds) and once v is removed from U (if either Case 1 or Case 2 holds). Indeed by 3. of Fact 3, Case 3 only applies to nodes in $U_i - Temp_i$. ■

Theorem 7 *For any graph $G = (V, E)$ and threshold function $t(\cdot)$, the algorithm $PA(G,t)$ returns a perfect seed set for G in $O(|E| \log |V|)$ time.*

Proof. In order to prove that the set S provided by the algorithm PA(G,t) is a perfect seed set for G, we first show that for each $i = 1, \ldots, \lambda$ the set S_i allows to make each node in

$$\mathcal{R}_i = \bigcup_{j=i}^{\lambda} (R_j \cup \{u : u \notin A_j,\ \delta_j(u) = 0\})$$

a spreader, that is,

$$\mathcal{R}_i \subseteq \mathsf{Spreader}_{G[U_i]}[S_i]. \tag{3}$$

Before proving (3), we show that it allows to conclude that the algorithm returns a perfect seed set. Indeed, this follows by recalling that $G[U_1] = G$ and observing that a node u is moved to the set A (at any round) only if $(\{u\} \cup N(u)) \cap \mathcal{R}_1 \neq \emptyset$ (that is, only if either the node itself or one of its neighbors is in $\mathcal{R}_1 \subseteq \mathsf{Spreader}_{G[U_1]}[S_1] = \mathsf{Spreader}_G[S]$) and that the algorithm terminates when all nodes are aware ($A = V$) and the set R is empty.

We show now (3) by induction on i from λ down to 1.
Consider first $i = \lambda$. Let v_λ be a node in $G[U_\lambda]$. Recalling that λ is the last step of the algorithm it must hold that at the beginning of the (non executed) round $\lambda + 1$

$$R_{\lambda+1} = \emptyset, \qquad A_{\lambda+1} = V. \tag{4}$$

Moreover, since at most one node is removed from R at each step, we have that either

$$R_\lambda = \emptyset \text{ or } R_\lambda = \{v_\lambda\}. \tag{5}$$

We distinguish three cases depending on which case of the algorithm applies to the selection of node v_λ.

- (Case 1 holds). In this case $k_\lambda(v_\lambda) = 0$ and v_λ is immediately a spreader in $G[U_\lambda]$. By (4) and (5) the statement (3) is clearly satisfied for $i = \lambda$.

- (Case 2 holds). In this case either it holds that $R_\lambda = \{v_\lambda\}$ and $k_\lambda(v_\lambda) > \delta_\lambda(v_\lambda)$ or it holds that $v_\lambda \notin A_\lambda$ and $\delta_\lambda(v_\lambda) = 0$. Moreover, $S_\lambda = \{v_\lambda\}$. Consequently, noticing that $A_{\lambda+1} = A_\lambda \cup \{v_\lambda\} = V$, we have $\mathcal{R}_\lambda \subseteq \{v_\lambda\} \subseteq \mathsf{Spreader}_{G[U_\lambda]}[S_\lambda]$.

- Finally we show that Case 3 can not apply during the last iteration of the algorithm. By contradiction assume that v_λ is selected in Case 3. Notice that since Case 1 and Case 2 do not apply, we have
  i) each $v \in U_\lambda$ has $k_\lambda(v) > 0$,
  ii) each $v \in U_\lambda - Temp_\lambda \cap R_\lambda$ has $\delta_\lambda(v) \geq k_\lambda(v)$,
  iii) each $v \notin A_\lambda$ has $\delta_\lambda(v) > 0$.
  Suppose first $R_\lambda = \emptyset$. We know that $A_\lambda \neq V$ (cfr. line 5 of the algorithm). If we assume $v_\lambda \in A_\lambda$ then no node is added to A except for the selected node v_λ (cfr. line 27 of the algorithm). Hence $A_{\lambda+1} = A_\lambda \cup \{v_\lambda\} = A_\lambda \neq V$ contradicting (4). Suppose now $R_\lambda = \emptyset$ and $v_\lambda \notin A_\lambda$. By iii) there exists a neighbor u of v_λ that is added to R (cfr. line 22 of

the algorithm). Hence, $R_{\lambda+1} \neq \emptyset$ contradicting (4).

Finally assume $R_\lambda = \{v_\lambda\}$. If $U_\lambda - Temp_\lambda - R_\lambda \neq \emptyset$ then v_λ should be selected in $U_\lambda - Temp_\lambda - R_\lambda = U_\lambda - Temp_\lambda - \{v_\lambda\}$ (cfr. line 20 of the algorithm), but this is obviously impossible. Otherwise, let $U_\lambda - Temp_\lambda - R_\lambda = \emptyset$. In this case, $U_\lambda - Temp_\lambda = \{v_\lambda\}$ and then $\delta_\lambda(v_\lambda) = 0$. But we know that, by ii) and i) we get $\delta_\lambda(v_\lambda) \geq k_\lambda(v_\lambda) > 0$.

Consider now $i < \lambda$ and suppose the algorithm be correct on $G[U_{i+1}]$, that is, we know that $\mathcal{R}_{i+1} \subseteq \mathsf{Spreader}_{G[U_{i+1}]}[S_{i+1}]$. We show that the algorithm is correct on $G[U_i]$ with thresholds $k_i(u)$, for $u \in U_i$.

The algorithm PA implies that for each $u \in U_i$ we have

$$k_{i+1}(u) = \begin{cases} \max(k_i(u) - 1, 0) & \text{if Case 1 or 2 hold and } u \in N(v_i) \cap U_i \\ k_i(u) & \text{otherwise,} \end{cases} \tag{6}$$

where v_i is the node selected at iteration i.

We distinguish three cases depending on which case of the algorithm applies when selecting v_i.

- (Case 3 holds). In this case, we know that $U_i = U_{i+1}$ and $S_{i+1} = S_i$. Moreover, by (6) we have $k_{i+1}(u) = k_i(u)$, for each $u \in U_{i+1}$.

  If we suppose $v_i \notin R_i$, we get that $R_i \subseteq R_{i+1}$ and consequently

  $$\mathcal{R}_i = \mathcal{R}_{i+1} \subseteq \mathsf{Spreader}_{G[U_{i+1}]}[S_{i+1}] = \mathsf{Spreader}_{G[U_i]}[S_i].$$

  Assuming otherwise that $v_i \in R_i$, we have $R_i \subseteq R_{i+1} \cup \{v_i\}$and $U_i - Temp_i - R_i = \emptyset$. By 3. of Fact 3, we then get $U_i - Temp_i = R_i$. As a consequence,

  $$(N(v_i) \cap (U_i - Temp_i)) \subseteq R_{i+1} \tag{7}$$

  Recalling that Case 3 of the algorithm holds and $v_i \in R_i$, we have $\delta_i(v_i) \geq k_i(v_i)$. From this, Fact 4, equation (7), and the induction hypothesis $\mathcal{R}_{i+1} \subseteq \mathsf{Spreader}_{G[U_{i+1}]}[S_{i+1}]$, we obtain that the set $S_{i+1} = S_i$ is able to make v_i a spreader in $G[U_i]$. Therefore, we can conclude that $\mathcal{R}_i \subseteq \mathsf{Spreader}_{G[U_i]}[S_i]$.

- (Case 2 holds). In this case we know that $U_{i+1} = U_i - \{v_i\}$, $R_i \subseteq R_{i+1} \cup \{v_i\}$, and $S_i = S_{i+1} \cup \{v_i\}$. Hence, $v_i \in \mathsf{Spreader}[S_i]$. Moreover by (6), it follows that for any $u \in N(v_i) \cap U_i$, if $u \in \mathsf{Spreader}[S_{i+1}]$ then $u \in \mathsf{Spreader}[S_i]$. Therefore, $\mathcal{R}_i \subseteq \mathsf{Spreader}[S_i]$.

- (Case 1 holds). In this case we have $k_i(v_i) = 0$, $U_{i+1} = U_i - \{v_i\}$, $R_i \subseteq R_{i+1} \cup \{v_i\}$ and $S_i = S_{i+1}$. Since $k_i(v_i) = 0$, node v_i is immediately a spreader in $G[U_i]$. Hence, each neighbor u of v_i in $G[U_i]$ has one more spreader neighbor and its threshold is updated according to (6). Therefore, since $\mathcal{R}_{i+1} \subseteq \mathsf{Spreader}_{G[U_{i+1}]}[S_{i+1}]$, we have that $\mathcal{R}_i \subseteq \mathsf{Spreader}_{G[U_i]}[S_i]$.

Finally, we notice that the PA algorithm can be implemented to run in $O(|E| \log |V|)$ time. Indeed we need to process the nodes $v \in V$ (each one at most two times (see Lemma 2)) according to the metrics $\delta(v)$ and $k(v)/(\delta(v)(\delta(v)+1))$, and the updates, that follows each processed node $v \in V$ involve at most $|N(v)|$ neighbors of v. ■

6 Experimental Results

Due to Theorem 1, we can not aim to any significant performance guaranteed on the seed set size for *general* graphs and threshold functions. Nonetheless, extensive experiments, show that the PA algorithm, presented and analyzed in Section 5, performs very well on real networks, both in terms of efficiency of the solution and of the convergence time.
In order to evaluate the PA algorithm, we conducted experiments on 12 real networks of various sizes from the Stanford Large Network Data set Collection (SNAP) [34], the Social Computing Data Repository at Arizona State University [41] and Newman's Network data [36]. The main characteristics of the studied networks are shown in Table 1.

Name	# of nodes	# of edges	Max degree	Size of the LCC	Clust. Coeff.	Modularity
Amazon0302 [34]	262111	1234877	420	262111	0.4198	0.6697
BlogCatalog3 [41]	10312	333983	3992	10312	0.4756	0.2374
BuzzNet [41]	101168	4284534	64289	101163	0.2508	0.3161
Ca-AstroPh [34]	18772	198110	504	17903	0.6768	0.3072
Ca-CondMath [34]	23133	93497	279	21363	0.7058	0.5809
Ca-GrQc [34]	5242	14496	81	4158	0.6865	0.7433
Ca-HepPh [34]	10008	118521	491	11204	0.6115	0.5085
Ca-HepTh [34]	9877	25998	65	8638	0.5994	0.6128
Cit-HepTh [34]	27770	352807	64	24700	0.3120	0.7203
Delicious [34]	103144	1419519	3216	536108	0.0731	0.602
Douban [41]	154907	327162	287	154908	0.048	0.5773
Facebook [34]	4039	88234	1045	4039	0.6055	0.8093
Jazz [36]	198	2742	100	198	17899	0.6334
Karate [36]	34	78	17	5	45	0.5879
Last.fm [41]	1191812	5115300	5140	1191805	0.1378	0.1378
Power grid [36]	4941	6594	19	4941	0.1065	0.6105
Youtube2 [41]	1138499	2990443	28754	1134890	0.1723	0.6506

Table 1: The networks.

The active information diffusion problem is a novel model of information diffusion and, to the best of our knowledge, no heuristic is known for the PA problem. For this reason we decided to evaluate the effectiveness of our algorithm (PA) with two heuristics that respectively solve

two problems related to the PA problem. The first heuristic, named *MTS* [20], is devoted to the minimum target set selection (TSS) problem where the aim is to have each node become a spreader. We have chosen this TSS heuristics since it experimentally outperforms the other known algorithms as well as simple baseline greedy strategies [14, 32, 39] for the TSS problem, see [20]. The rationale of this comparison is to show that by relaxing the goal of the TSS model for the new model (which only aims to make each node aware) we are able to identify significantly smaller seed sets. On the other hand, when all the thresholds $t(v)$ are equal to the node degrees $d(v)$, the PA problem is equivalent to the well known dominating set problem. For this reason we will compare our algorithm with the (best known) heuristic [5], named *DOM*, for the dominating set problem. It is worth mentioning that all the three competing algorithms have the same time complexity ($O(|E| \log |V|)$).

Thresholds values. We tested the three algorithms using two types of threshold function:

- *Random thresholds* where $t(v)$ is chosen uniformly at random in the interval $[1, d(v)]$.
- *Proportional thresholds,* where for each v the threshold $t(v)$ is set as $\alpha \times d(v)$ with $\alpha = 0.1, 0.2, \ldots, 1$. Notice that for $\alpha = 0.5$ we are considering a particular version of the activation process named "majority" thresholds, while for $\alpha = 1$ we are considering the dominating set problem.

Summarizing, our experiments compare both the size and the number of rounds to reach the final "all-aware" configuration of the seed sets generated by 3 algorithms (PA, MTS, Dom) on 17 networks (see Table 1), fixing the thresholds in 11 different ways (Randomly and Proportionally with $\alpha = 0.1, 0.2, \ldots, 1$). Overall we performed $3 \times 17 \times 11 = 561$ tests. Since the random thresholds test settings involve some randomization, we executed each test 10 times. The results were compared using means of target set sizes (the observed variance was negligible).

Random Thresholds. Table 2 depicts the results of the Random threshold test setting, that is using random thresholds (for each test setting, the same thresholds values have been used for both the algorithms). On the left side, each number represents the size of the perfect seed set generated by PA and MTS algorithms. Similarly, on the right side, each number represents the the number of rounds to reach the final "all-aware" configuration starting the diffusion process with the perfect seed set generated by PA and MTS algorithms. The value in bracket represents the overhead percentage of MTS algorithm compared to the PA algorithm.

Proportional thresholds. Figures 3–7 report the results for the proportional thresholds settings. For each network, the left plot depicts the size of the perfect seed set (Y-axis), for

each value of $\alpha \in [0.1, 1]$ (X-axis) and for each algorithm (series) while the right plot depicts the number of rounds to reach the final "all-aware" configuration (Y-axis), for each value of $\alpha \in [0.1, 1]$ (X-axis) and for each algorithm (series). We present the results only for 5 of the considered networks because the experiments performed on the other networks exhibit quite similar behaviors. Comparisons of the results obtained on all the networks for the cases $\alpha = 0.5$ (i.e., majority) and $\alpha = 1$ (i.e., domination) are reported in Figures 8 and 9 respectively. In both these figures a logarithmic scale is used for the Y-axis (i.e., the size of the perfect seed set).

The experiment results confirm our hypothesis: The size of the initial seed set provided by our PA algorithm is in general significantly smaller than the size of the set provided by the other strategies. We notice that the gap between the PA and the MTS algorithms increases with the value of the node thresholds (this result was expected: the larger the value of $t()$, the larger the difference between the models). The PA algorithm is always better than the DOM algorithm, when $t(v) < d(v)$. Moreover when $t(v) = d(v)$ (that is, when the PA problems becomes the dominating set problem), the two algorithms provide comparable results (see Figure 9), hence the PA algorithm could be considered as an effective alternative heuristics for the dominating set problem. With respect to the number of rounds needed to reach the desired "all-aware" configuration, a part the Dom strategy that always requires one round, the algorithms always provide comparable results.

Name	Seed set size PA	Seed set size MTS		Rounds PA	Rounds MTS
Amazon0302	36753	38804 (5.58%)		33	27 (-18.18%)
BlogCatalog3	10	12 (20%)		43	39 (-9.30%)
BuzzNet	141	998 (607.80%)		52	46 (-11.54%)
Ca-AstroPh	919	1157 (25.9%)		32	29 (-9.38%)
Ca-CondMath	1573	1810 (15.07%)		21	20 (-4.76%)
Ca-GrQc	636	661 (3.93%)		13	14 (7.69%)
Ca-HepPh	790	901 (14.05%)		35	29 (-17.14%)
Ca-HepTh	964	945 (-1.97%)		16	18 (12.5%)
Cit-HepTh	955	1045 (9.42%)		76	74 (-2.63%)
Delicious	43653	44578 (2.12%)		53	46 (-13.21%)
Douban	2374	2343 (-1.31%)		22	24 (9.09%)
Facebook	9	213 (2267%)		17	25 (47.06%)
Jazz	4	7 (75%)		11	15 (36.36%)
Karate	3	3 (0%)		7	5 (40%)
Last.fm	33046	33430 (9.75%)		22	27 (1.16%)
Power grid	352	340 (-3.41%)		16	23 (43.75%)
Youtube2	27403	27801 (1.45%)		38	36 (-5.26%)

Table 2: Random Thresholds Results: For each network and each algorithm, the average size of the perfect seed set and the number of rounds to reach the final "all-aware" configuration are given.

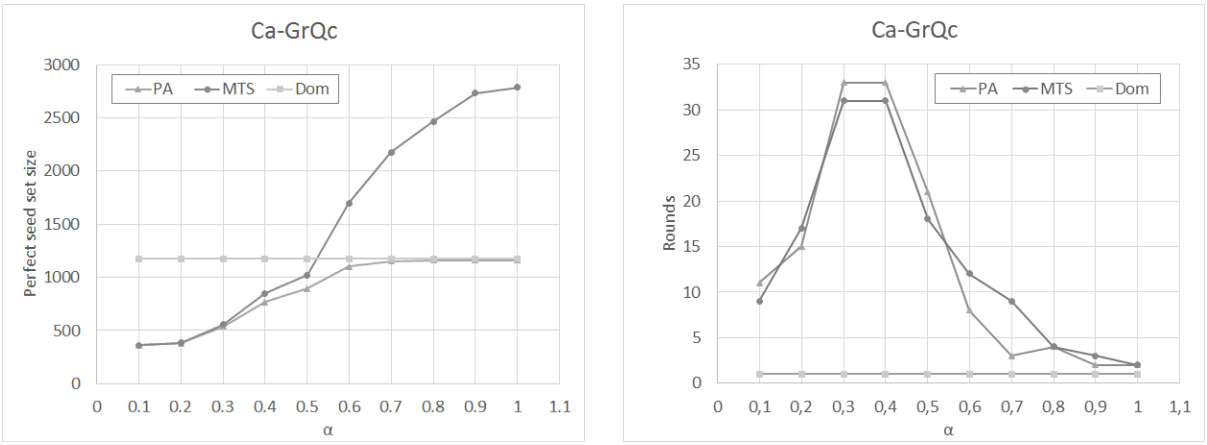


Figure 3: Proportional Thresholds Results: CA-GrQc network.

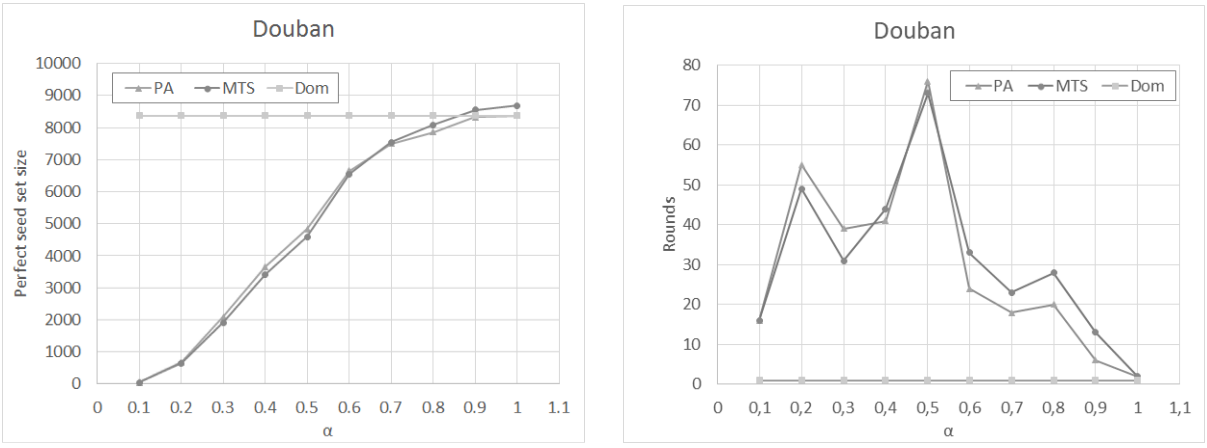


Figure 4: Proportional Thresholds Results: Douban network.

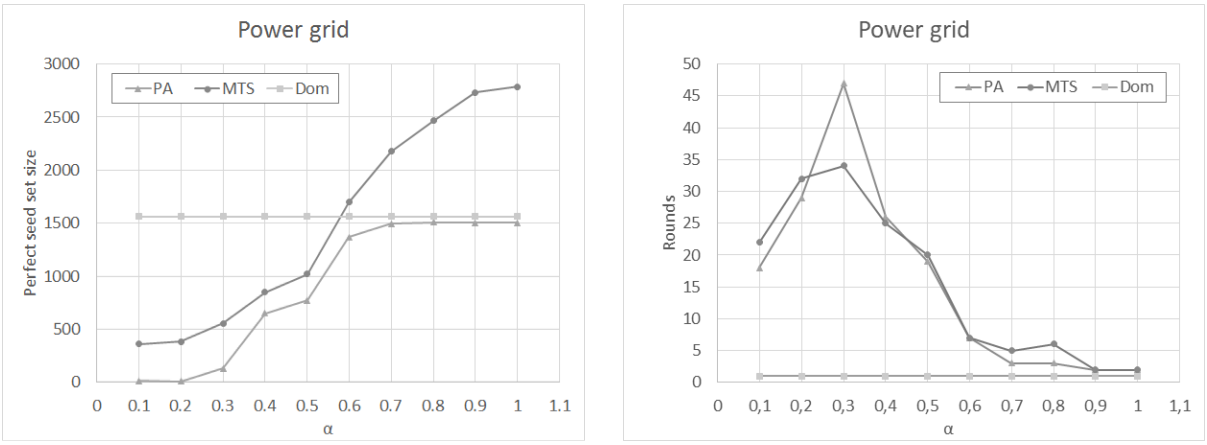


Figure 5: Proportional Thresholds Results: Power grid network.

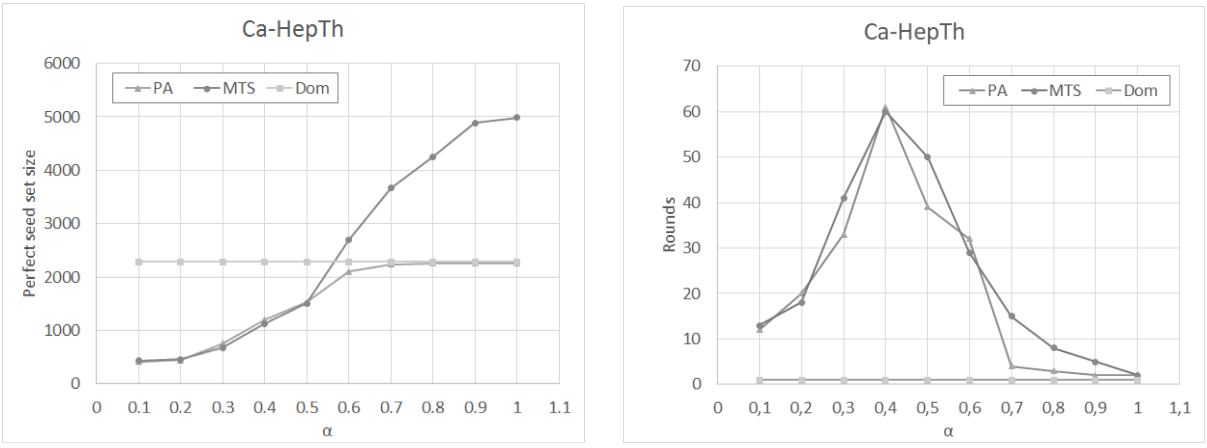


Figure 6: Proportional Thresholds Results: Ca-HepTh network.

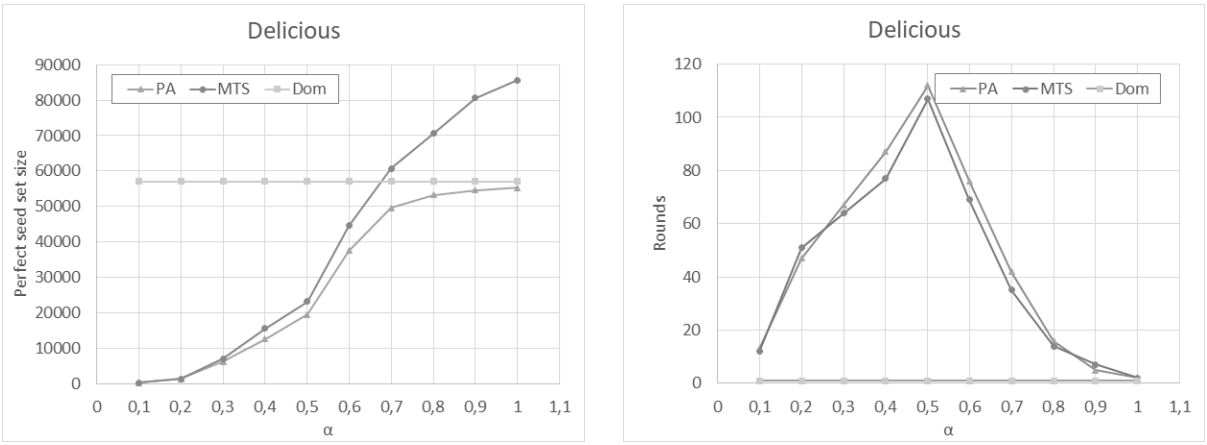
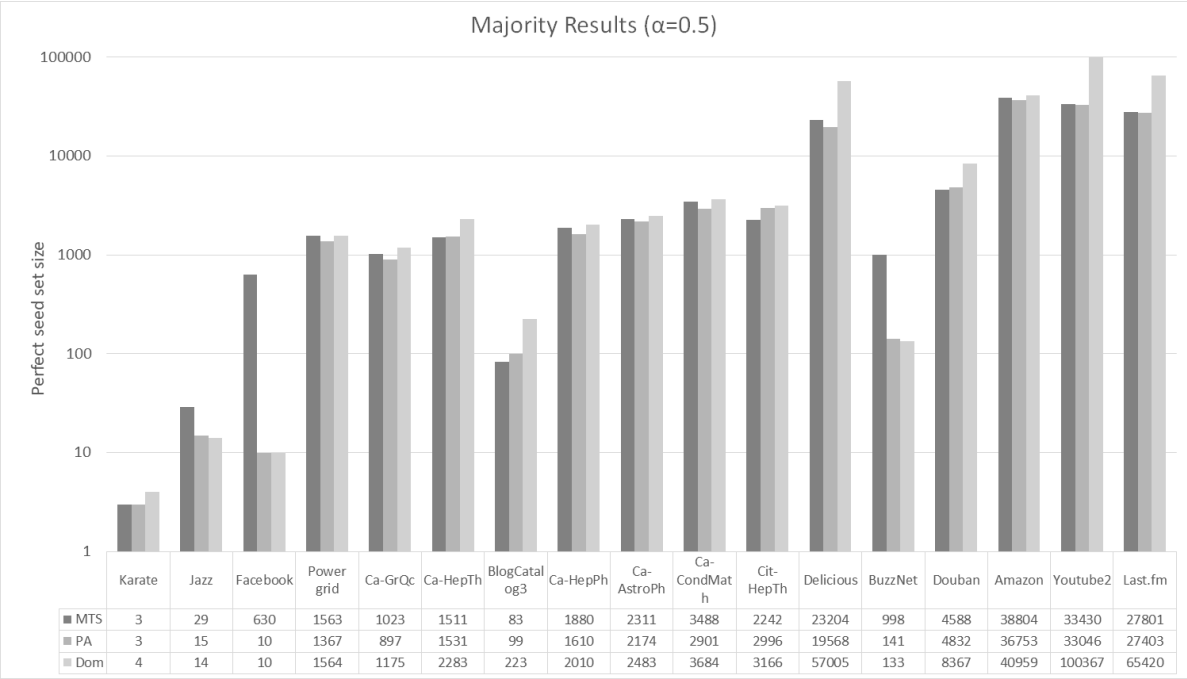
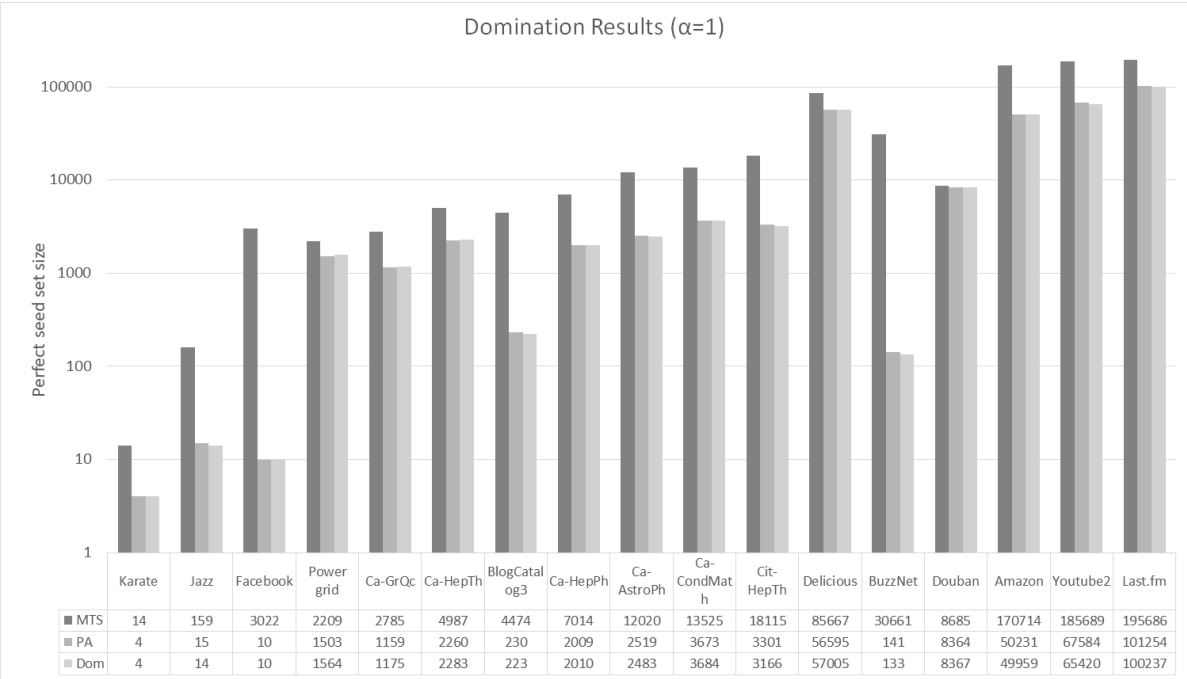


Figure 7: Proportional Thresholds Results: Delicious network.



	Karate	Jazz	Facebook	Power grid	Ca-GrQc	Ca-HepTh	BlogCatalog3	Ca-HepPh	Ca-AstroPh	Ca-CondMath	Cit-HepTh	Delicious	BuzzNet	Douban	Amazon	Youtube2	Last.fm
MTS	3	29	630	1563	1023	1511	83	1880	2311	3488	2242	23204	998	4588	38804	33430	27801
PA	3	15	10	1367	897	1531	99	1610	2174	2901	2996	19568	141	4832	36753	33046	27403
Dom	4	14	10	1564	1175	2283	223	2010	2483	3684	3166	57005	133	8367	40959	100367	65420

Figure 8: Majority Results ($\alpha = 0.5$).



	Karate	Jazz	Facebook	Power grid	Ca-GrQc	Ca-HepTh	BlogCatalog3	Ca-HepPh	Ca-AstroPh	Ca-CondMath	Cit-HepTh	Delicious	BuzzNet	Douban	Amazon	Youtube2	Last.fm
MTS	14	159	3022	2209	2785	4987	4474	7014	12020	13525	18115	85667	30661	8685	170714	185689	195686
PA	4	15	10	1503	1159	2260	230	2009	2519	3673	3301	56595	141	8364	50231	67584	101254
Dom	4	14	10	1564	1175	2283	223	2010	2483	3684	3166	57005	133	8367	49959	65420	100237

Figure 9: Domination Results ($\alpha = 1$).

7 Conclusion and Open Problems

We have studied some algorithmic aspects of a recently introduced information diffusion model, that differentiates among spreaders and aware nodes [16]. Many interesting questions related to this model remain open and might be interesting to study:

- It would be interesting to establish to what extent the general hardness result still holds for dense graphs.
- More generally, are there classes of graphs, other then trees and cliques, for which the problem can be either efficiently solved or admits a *small* approximation factor?
- It would also be interesting to determine a significant upper bound on the size of a perfect seed set returned by the proposed PA algorithm in terms of node degrees and thresholds in the spirit of the bound given in [1] for the TSS problem.

References

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

[2] L.A. Adamic, T.M. Lento, E. Adar, P.C. Ng. Information Evolution in Social Networks, *Proc. of the 9th ACM Inter. Conference on Web Search and Data Mining*, (2016) 473–482

[3] Eytan Bakshy, Itamar Rosenn, Cameron Marlow, and Lada Adamic. The role of social networks in information diffusion. In *Proceedings of the 21st International Conference on World Wide Web*, pages 519–528, 2012.

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

[5] Alina Campan, Traian Marius Truta, and Matthew Beckerich. Fast dominating set algorithms for social networks. In *MAICS*, 2015.

[6] Carmen C. Centeno, Mitre C. Dourado, Lucia Draque Penso, Dieter Rautenbach, and Jayme L. Szwarcfiter. Irreversible conversion of graphs. *Theoretical Computer Science*, 412(29):3693–3700, 2011.

[7] Ning Chen. On the approximability of influence in social networks. *SIAM Journal on Discrete Mathematics*, 23(3):1400–1415, 2009.

[8] Wei Chen, Carlos Castillo, and Laks Lakshmanan. *Information and Influence Propagation in Social Networks*. Morgan & Claypool, 2013.

[9] Chun-Ying Chiang, Liang-Hao Huang, and Hong-Gwa Yeh. Target set selection problem for honeycomb networks. *SIAM Journal on Discrete Mathematics*, 27(1):310–328, 2013.

[10] Morgan Chopin, André Nichterlein, Rolf Niedermeier, and Mathias Weller. Constant thresholds can make target set selection tractable. *Theory of Computing Systems*, 55(1):61–83, 2014.

[11] Ferdinando Cicalese, Gennaro Cordasco, Luisa Gargano, Martin Milanič, Joseph Peters, and Ugo Vaccaro. Spread of influence in weighted networks under time and budget constraints. *Theoretical Computer Science*, 586:40–58, 2015.

[12] Ferdinando Cicalese, Gennaro Cordasco, Luisa Gargano, Martin Milanič, and Ugo Vaccaro. Latency-bounded target set selection in social networks. *Theoretical Computer Science*, 535:1–15, 2014.

[13] Amin Coja-Oghlan, Uriel Feige, Michael Krivelevich, and Daniel Reichman. Contagious sets in expanders. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1953–1987, 2015.

[14] Gennaro Cordasco, Luisa Gargano, Marco Mecchia, Adele A. Rescigno, and Ugo Vaccaro. Discovering Small Target Sets in Social Networks: A Fast and Effective Algorithm. In *Algorithmica*, ISSN: 0178-4617, 2017

[15] Gennaro Cordasco, Luisa Gargano and Adele A. Rescigno. Active Spreading in Networks. In *Proceedings of the 17th Italian Conference on Theoretical Computer Science ICTCS,* Lecce, Italy, September 7-9, pages 149–162, 2016.

[16] Gennaro Cordasco, Luisa Gargano, Adele A. Rescigno, and Ugo Vaccaro. Evangelism in Social Networks: Algorithms and Complexity. In *NETWORKS*, ISSN: 1097-0037, 2017.

[17] Gennaro Cordasco, Luisa Gargano, Adele A. Rescigno, and Ugo Vaccaro. Optimizing Spread of Influence in Social Networks via Partial Incentives. In *Structural Information and Communication Complexity: 22nd International Colloquium, SIROCCO*, pages 119–134. Springer International Publishing, 2015.

[18] Gennaro Cordasco, Luisa Gargano, Adele A. Rescigno, and Ugo Vaccaro. Brief announcement: Active information spread in networks. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing PODC*, pages 435–437, New York, NY, USA, 2016.

[19] Gennaro Cordasco, Luisa Gargano, Adele A. Rescigno. On Finding Small Sets that Influence Large Networks. In *Social Network Analysis and Mining SNAM,* Vol. 6(1), 2016.

[20] Gennaro Cordasco, Luisa Gargano, and Adele Anna Rescigno. Influence propagation over large scale social networks. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2015*, pages 1531–1538, 2015.

[21] R. Dawkins. *The Selfish Gene*, Oxford University Press, (1989).

[22] Thang N. Dinh, Huiyuan Zhang, Dzung T. Nguyen, and My T. Thai. Cost-effective viral marketing for time-critical campaigns in large-scale social networks. *IEEE/ACM Trans. Netw.*, 22(6):2001–2011, December 2014.

[23] David Easley and Jon Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, New York, NY, USA, 2010.

[24] Luisa Gargano, Pavol Hell, Joseph G. Peters, Ugo Vaccaro. Influence diffusion in social networks under time window constraints. *Theor. Comput. Sci.*, 584(C):53–66, 2015.

[25] Gunderson, K. Minimum degree conditions for small percolating sets in bootstrap percolation. ArXiv e-prints, 1703.10741, 2017.

[26] T.W. Hayne, S.T. Hedetniemi, and P.J. Slater. Fundamentals of domination in graphs. Marcel Dekker, Inc., New York, 1998.

[27] A. Hellwig, and L. Volkmann. Some upper bounds for the domination number. *J. Combin. Math. Combin. Comput.*, 57:187–209, 2006.

[28] Dorit Hochbaum (Editor). Approximation Algorithms for NP-Hard Problems. PWS Publishing Company, 1997.

[29] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proc. of the ACM SIGKDD KDD 2003*, pages 137–146, 2003.

[30] David Kempe, Jon Kleinberg, and Éva Tardos. Influential nodes in a diffusion model for social networks. In *Proc. of ICALP 2005*, pages 1127–1138, 2005.

[31] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. *Theory of Computing*, 11(4):105–147, 2015.

[32] Suman Kundu and Sankar K. Pal. Deprecation based greedy strategy for target set selection in large scale social networks. *Information Sciences*, 316:107–122, 2015.

[33] Jure Leskovec, Lada A. Adamic, and Bernardo A. Huberman. The dynamics of viral marketing. *ACM Trans. Web*, 1(1), May 2007.

[34] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, 2015.

[35] Mark E. J. Newman. Modularity and community structure in networks. *Proc. Natl. Acad. Sci. U.S.A.* 103 (23): 85778582.

[36] Mark Newman. Network data, `http://www-personal.umich.edu/~mejn/netdata/`, 2015.

[37] André Nichterlein, Rolf Niedermeier, Johannes Uhlmann, Mathias Weller. On tractable cases of target set selection. *Social Network Analysis and Mining*, 3(2):233–256, 2013.

[38] T. V. Thirumala Reddy and C. Pandu Rangan. Variants of spreading messages. *J. Graph Algorithms Appl.*, 15(5):683–699, 2011.

[39] Paulo Shakarian, Sean Eyre, and Damon Paulo. A scalable heuristic for viral marketing under the tipping model. *Social Network Analysis and Mining*, 3(4):1225–1248, 2013.

[40] Michela Del Vicario, Alessandro Bessi, Fabiana Zollo, Fabio Petroni, Antonio Scala, Guido Caldarelli, Eugene Stanley and Walter Quattrociocchi. The spreading of misinformation onlines. *PNAS*, 113(3):554–559, 2016.

[41] Reza Zafarani and Huan Liu. Social computing data repository at ASU. http://socialcomputing.asu.edu, 2009.