Active Spreading in Networks

G. Cordasco[2], L. Gargano[1], and A. A. Rescigno[1]

[1] Department of Informatics, University of Salerno, Italy,
[2] Department of Psychology, Second University of Naples, Italy.

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. We study the problem of choosing the smallest set of initial spreaders that guarantee that all the nodes become aware of the information. 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.

1 Introduction

During the past decade spreading processes in complex networks have experienced a particular surge of interest. A large part of 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 [7,23,30]. 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 n individuals that, with respect to the information, are subdivided into *ignorant, aware*, and *spreading*. Initially, all individuals are ignorant. Then an initial set of spreaders is selected. When a spreader informs an ignorant node v, the ignorant node v becomes aware; as soon as the individual v is informed by a number of spreaders greater than a threshold $t(v)$, it starts spreading the information itself. The motivations that lead us to consider such a scenario come from experimental

V. Biló, A. Caruso (Eds.): ICTCS 2016, Proceedings of the 17th Italian Conference on Theoretical Computer Science, 73100 Lecce, Italy, September 7–9 2016, pp. 149–162 published in CEUR Workshop Proceedins Vol-1720 at `http://ceur-ws.org/Vol-1720`

studies of how information spread 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 [2] for a study of how exposure to social signals affects diffusion.

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 members of the network, i.e., $(u, v) \in E$ if individuals u and v can directly communicate. We posit a threshold function $t : V \to \{0, 1, 2, \ldots\}$, and we denote by $N(v)$ the neighborhood of $v \in V$. 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 $\mathsf{Spreader}_G[S, 0] = S$ and

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

for $\tau \geq 1$. The process terminates when $\mathsf{Spreader}_G[S, \rho] = \mathsf{Spreader}_G[S, \rho - 1]$ for some $\rho > 1$. We denote by $\mathsf{Spreader}_G[S] = \mathsf{Spreader}_G[S, \rho]$. Hence, when the process stops the set of aware nodes is

$$\mathsf{Aware}_G[S] = \mathsf{Spreader}_G[S] \cup \Big\{ u \, s.t. \, N(u) \cap \mathsf{Spreader}_G[S] \neq \emptyset \Big\}.$$

Given G, a threshold function $t(\cdot)$, we aim to identify a small node set $S \subseteq V$ such that $\mathsf{Aware}_G[S] = V$.[3] Namely, 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 seed set $S \subseteq V$ of minimum size such that $\mathsf{Aware}[S] = V$.

We refer to the set S for which $\mathsf{Aware}[S] = V$ as a perfect seed set and to the nodes in S as *seeds*.

1.1 Related Work and Our Results

The above algorithmic problem 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 [7,17] and references quoted therein). The first authors to study spread of influence in networks from an algorithmic point of view were Kempe *et al.* [19,20,21]. Chen [6] studied the following minimization problem: given a graph G and fixed thresholds $t(v)$, for each vertex 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). He 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

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

of papers [1,3,5,8,9,10,11,12,18,27,28] that isolated interesting cases in which the problem (and variants thereof) become tractable. Heuristics for the TSS problem that work for general graphs have been proposed in the literature [13,16,29].

However, the papers appeared in the scientific literature considered the basic model in which *any* node, as soon as it is influenced by its neighbors, it immediately starts spreading influence. In this paper we consider a more refined model that differentiates among spreaders and plain aware node. This model has been first considered in [14], where the Awareness Maximization Problem in which one asks for a set S, with $|S| \leq \beta$, that achieves the maximum awareness in the network has been studied.

In Section 2, we study the computational complexity of the PA problem and extend the TSS problem hardness result to the PA problem. In Section 3, we give an algorithm that outputs a perfect seed set for any input graph. Experimental evaluation of the proposed algorithm is given in Section 4; it shows that the proposed algorithm outperforms some heuristics developed for related problems. Finally, we show that our problem becomes tractable if the graph is a tree (Section 5).

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.

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 $t : V \to N$, 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 [6].

Theorem 1. *The PA problem cannot 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 a triangle 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 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 influence at most 2 nodes in $V' - V'_0$—the other vertices of the triangle its belongs to. Hence, in order to influence all the nodes in $V' - V'_0$ all 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 $\mathsf{Spreader}_G[\{v_i \mid S' \cap V'_i \neq \emptyset\}] = V$. □

We notice that the Target Set Selection problem remains hard when each node has threshold upper bounded by a constant; in particular, it was proved in [6] that approximating it when each node has threshold at most 2 is as hard as approximating the problem 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.

3 A general algorithm for the PA problem

In this section we propose an algorithm for the PA problem in case of arbitrary graphs and thresholds. The algorithm PA(G, t), given in Algorithm 1, 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 being influenced); 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 be indeed spreaders).

The algorithm proceeds as follows: As long as there exists at least a non-aware node or there is a node in 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 being influenced (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 receive v's influence).

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 will receive its influence). Notice that Case 1 can also apply to nodes in $Temp$.

Algorithm 1: 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 ∪ {u} where u = argmax_{w∈N(v)∩(U−Temp)} {δ(w)}
23 |  |  |  |  |  foreach z ∈ N(u) ∩ 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
```

In such a case the value of $\delta()$ of the neighbors of the selected node v were already reduced by 1—when v 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, Case 2 and 3 only apply to nodes outside $Temp$. In other words, nodes moved to $Temp$ will never 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 $\delta()$ 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 [15]. 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 $\mathrm{PA}(G, t)$ optimally returns a single seed: At the first iteration of the while loop, Case 3) applies and a node v_1 is selected; then a node v_2 is marked as required 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 $\mathrm{PA}(G, t)$. Given a subset $V' \subseteq V$ of vertices 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 $\mathrm{PA}(G, t)$, for each $i = 1, \ldots, \lambda$ we denote:

- by v_i the node selected during the i-th iteration;
- by $U_i, Temp_i, S_i, R_i, A_i, \delta_i(u)$, and $k_i(u)$, the sets $U, Temp, S, R, A$ and the values of $\delta(u), k(u)$, respectively, as updated at the beginning of the i-th iteration.

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

The following properties will be useful for the algorithm analysis.

Fact 1 *For each iteration i of the while loop in PA(G, t),*
1. $V - U_i \subseteq A_i$ **2.** $Temp_i \subseteq A_i$ **3.** $R_i \subseteq U_i - Temp_i$

Fact 2 *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 1. *Algorithm PA(G, t) executes at most* $2n$ *iterations of the while loop (i.e.,* $\lambda \leq 2n$*).*

Proof. First of all we prove that, at each iteration $i \geq 1$ of the while loop of PA(G, t), a 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 1 $V - U_i \subseteq A_i$ we have that there exist $u \in U_i$ such that $u \notin A_i$. Then using 2. of Fact 1 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 is eventually inserted in $Temp$ (if Case 3 applies) and once v is removed from U (if either Case 1 or Case 2 apply). Indeed by 3. of Fact 1, Case 3 only applies to nodes in $U_i - Temp_i$.

Theorem 2. *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 show 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 is able to make all the nodes in

$$\mathcal{R}_i = \bigcup_{j=i}^{\lambda} (R_i \cup \{u \notin A_i \text{ such that } \delta_i(u) = 0\})$$

a spreader, that is $\mathcal{R}_i \subseteq \mathsf{Spreader}_{G[U_i]}[S_i]$. We show it by induction with i going from λ back to 1.

Consider first $i = \lambda$. Let v_λ a node in $G[U_\lambda]$. Since λ is the last step and at most one node is removed from R at each step, we have that $R_\lambda = \emptyset$ or $R_\lambda = \{v_\lambda\}$. We distinguish three cases on the selected node v_i.

- (Case 1 holds). In this case $k_\lambda(v_\lambda) = 0$ and v_λ is immediately spreader in $G[U_\lambda]$ and the statement is clearly satisfied.
- (Case 2 holds). In this case ($R_\lambda = \{v_\lambda\}$ and $k_\lambda(v_\lambda) > \delta_\lambda(v_\lambda)$) or ($v_\lambda \notin A_\lambda$ and $\delta_\lambda(v_\lambda) = 0$) and consequently $S_\lambda = \{v_\lambda\}$ and $\mathcal{R}_\lambda \subseteq \mathsf{Spreader}_{G[U_\lambda]}[S_\lambda]$.
- Finally we show that case 3 cannot hold at the last iteration of the algorithm. Indeed if $R_\lambda = \emptyset$ then $v_\lambda \notin A_\lambda$ (otherwise the algorithm cannot terminate at round λ). In this case a new node is added to R at the line 22 of the algorithm and the algorithm cannot terminate at round λ. We notice that this node must exists, otherwise $\delta_\lambda(v_\lambda) = 0$ and Case 2 holds. On the other hand, if $R_\lambda = \{v_\lambda\}$ then $U_\lambda - Temp_\lambda - R_\lambda = \emptyset$ and we have $U_\lambda - Temp_\lambda = \{v_\lambda\}$ and consequently $\delta_\lambda(v_\lambda) = 0$. Since we are in case tree we also know that $k_\lambda(v_\lambda) > 0$ and Case 2 holds.

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

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

where v_i is the node selected at iteration i.

We distinguish three cases on the selected node v_i.

- (Case 3 holds). In this case $U_i = U_{i+1}$ and $S_{i+1} = S_i$. Moreover by (1), $k_{i+1}(u) = k_i(u)$ for each $u \in U_{i+1}$. If $v_i \notin R_i$ then $R_i \subseteq R_{i+1}$ and consequently $\mathcal{R}_i = \mathcal{R}_{i+1} \subseteq \mathsf{Spreader}_{G[U_{i+1}]}[S_{i+1}] = \mathsf{Spreader}_{G[U_i]}[S_i]$.
  Otherwise ($v_i \in R_i$) we have $R_i \subseteq R_{i+1} \cup \{v_i\}$, $U_i - Temp_i - R_i = \emptyset$ and by 3. of Fact 1 , we have $U_i - Temp_i = R_i$. Hence,
  $$(N(v) \cap (U_i - Temp_i)) \subseteq R_{i+1} \tag{2}$$
  Since we are in Case 3 and $v_i \in R_i$ then $\delta_i(v) \geq k_i(v)$. Using this, Fact 2 and equation (2), we have that since $\mathcal{R}_{i+1} \subseteq \mathsf{Spreader}_{G[U_{i+1}]}[S_{i+1}]$ then $S_{i+1} = S_i$ is able to make v_i a spreader in $G[U_i]$ and we have $\mathcal{R}_i \subseteq \mathsf{Spreader}_{G[U_i]}[S_i]$.
- (Case 2 holds). In this case $U_{i+1} = U_i - \{v_i\}$, $R_i \subseteq R_{i+1} \cup \{v_i\}$ and $S_i = S_{i+1} \cup \{v_i\}$. Hence $v_i \in \mathsf{Spreader}[S_i]$. Moreover by (1), it follows that for any $u \in N(v_i) \cap U_i$, if $u \in \mathsf{Spreader}[S_{i+1}]$ then $u \in \mathsf{Spreader}[S_i]$. Hence $\mathcal{R}_i \subseteq \mathsf{Spreader}[S_i]$.
- (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 spreader in $G[U_i]$. Hence by (1), each neighbor u of v_i in $G[U_i]$ is influenced by v_i and its threshold is updated according to (1). Therefore, since $\mathcal{R}_{i+1} \subseteq \mathsf{Spreader}_{G[U_{i+1}]}[S_{i+1}]$, we have that $\mathcal{R}_i \subseteq \mathsf{Spreader}_{G[U_i]}[S_i]$.

The statement follows since $G[U_1] = G$.

The theorem follows by observing that a node is moved to the set A only if $(v \cup N(v)) \cap \mathcal{R}_1 \neq \emptyset$ and that the algorithm terminates when all nodes are aware ($A = V$) and the set R is empty.

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 1)—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.

4 Experimental Results

Due to Theorem 1, we cannot aim to any significant performance guaranteed on the seed set size for *general* graphs and threshold functions. Nonetheless, extensive experiments show that our algorithm performs very well on large real networks, both in terms of efficiency of the solution and of the running time.

We conducted experiments on 12 real networks of various sizes from the Stanford Large Network Data set Collection (SNAP) [24], the Social Computing Data Repository at Arizona State University [31] and Newman's Network data [26]. The main characteristics of the studied networks are shown in Table 1.

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

Name	# of nodes	# of edges	Max degree	Size of the LCC	Clust. Coeff.	Modularity
BlogCatalog3 [31]	10312	333983	3992	10312	0.4756	0.2374
Ca-AstroPh [24]	18772	198110	504	17903	0.6768	0.3072
Ca-CondMath [24]	23133	93497	279	21363	0.7058	0.5809
Ca-GrQc [24]	5242	14496	81	4158	0.6865	0.7433
Ca-HepPh [24]	10008	118521	491	11204	0.6115	0.5085
Ca-HepTh [24]	9877	25998	65	8638	0.5994	0.6128
Cit-HepTh [24]	27770	352807	64	24700	0.3120	0.7203
Douban [31]	154907	327162	287	154908	0.048	0.5773
Facebook [24]	4039	88234	1045	4039	0.6055	0.8093
Jazz [26]	198	2742	100	198	17899	0.6334
Karate [26]	34	78	17	5	45	0.5879
Power grid [26]	4941	6594	19	4941	0.1065	0.6105

Table 1. The networks.

problem. The first heuristic, named *MTS* [15], 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 [13,22,29] for the TSS problem, see [15].

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 [4], named *DOM*, for the Dominating Set problem.
Thresholds values. We tested the three algorithms using two categories of threshold function:

- *Random thresholds* where $t(v)$ is chosen uniformly at random in the interval $[1, d(v)]$. 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);
- *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.

4.1 Test Results

Random Thresholds. Table 2 depicts the results of the Random threshold test setting. Each number represents the average size of the perfect seed set generated by PA and MTS algorithms on each network using random thresholds (for each test setting, the same thresholds values have been used for both the algorithms). The

Name	PA	MTS
BlogCatalog3	10	12 (20%)
Ca-AstroPh	919	1157 (25.9%)
Ca-CondMath	1573	1810 (15.07%)
Ca-GrQc	636	661 (3.93%)
Ca-HepPh	790	901 (14.05%)
Ca-HepTh	964	945 (-1.97%)
Cit-HepTh	955	1045 (9.42%)
Douban	2374	2343 (-1.31%)
Facebook	9	213 (2267%)
Jazz	4	7 (75%)
Karate	3	3 (0%)
Power grid	352	340 (-3.41%)

Table 2. Random Thresholds Results: For each network and each algorithm, the average size of the perfect seed set is depicted.

value in bracket represents the overhead percentage of MTS algorithm compared to the PA algorithm.

Constant and Proportional thresholds. Figures 1 and 2 report the results for the proportional thresholds settings. For each network the 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). We present the results only for 4 networks because of space limits; the experiments performed on the other networks exhibit similar behaviors.
The results in Fig. 1 and 2 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 increase 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, hence the PA algorithm could be considered as an effective alternative heuristics for the dominating set problem.

5 Trees.

Let $T = (V, E)$ be a tree rooted at any node r, and let $T(v)$ the subtree rooted at v, for any $v \in V$. We can prove that the algorithm PA outputs an optimal perfect seed set whenever the input graph is a tree.

Theorem 3. *PA(T, t) returns an* optimal *perfect seed set for any tree T.*

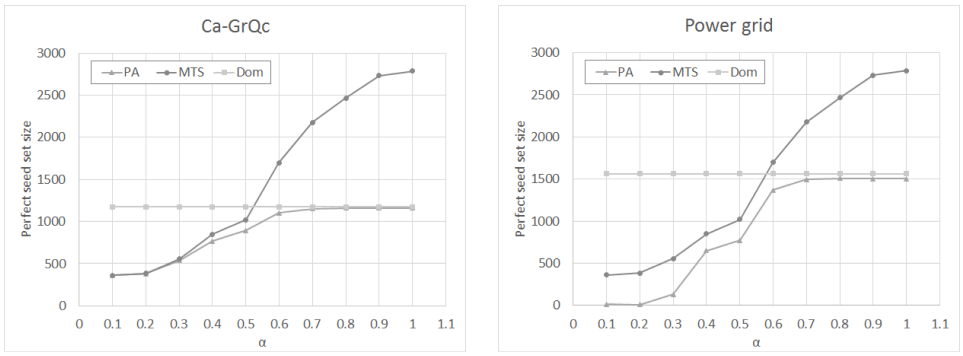

Fig. 1. Proportional Thresholds Results: CA-GrQc network and Power grid network

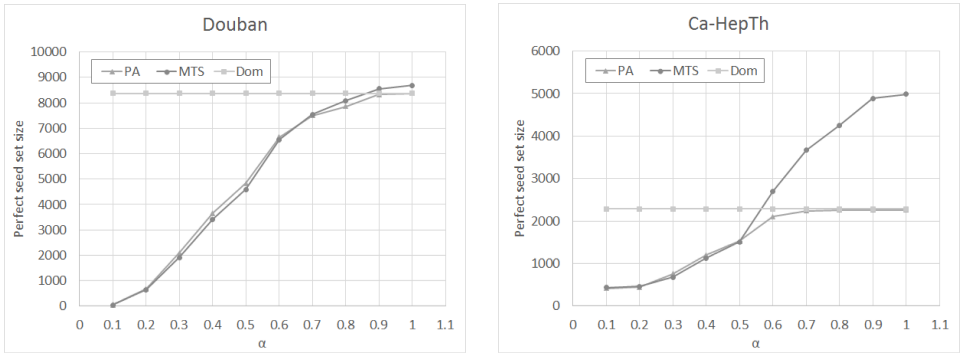

Fig. 2. Proportional Thresholds Results: Douban network and Ca-HepTh network.

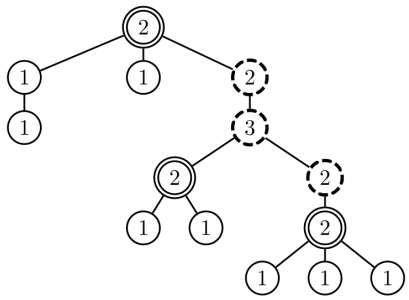

Fig. 3. Numbers inside circles are the node thresholds, double–circled denote seeds, dashed–circled lines denote aware nodes, solid–circled nodes denote spreaders.

If T is the tree in Fig. 3 one can see that the algorithm PA(T, t) returns a optimal seed set—consisting of the three double-circled nodes in the figure.

In order to evaluate the time complexity for trees, we report as TREE-PA the rewriting of the general PA algorithm in Section 3 in case the input graph is known to be a tree. One can see that the algorithm essentially computes the seed set while performing a visit (in BFS reverse order) of the tree. We can then show that

Theorem 4. *The* PA *problem can be solved in linear time for any tree.*

Algorithm 2: TREE-PA(T, t), $T = (V, E)$ is a tree with thresholds $t(v)$ for $v \in V$

```
1  S = ∅;   A = ∅;   P = ∅;
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
5              t(f_v) = t(f_v) − 1;                    // f_v denotes v's father
6              A = A ∪ {f_v}
7          else
8              if v ∈ P AND t(v) ≥ 2 then
9                  S = S ∪ {v};
10                 t(f_v) = t(f_v) − 1;
11                 A = A ∪ {f_v}
12             else
13                 if v ∉ A OR (v ∈ P AND t(v) = 1) then
14                     P = P ∪ {f_v}                   // f_v must spread
15     if v = r AND t(v) > 0 AND v ∉ A − P then S = S ∪ {v}
16 return S
```

6 Conclusion and Open Problems

We have studied some algorithmic aspects of a recently introduced information diffusion model, that differentiates among spreaders and aware nodes [14]. Many interesting questions related to this model remain open and might be interesting to study:

- Real life social networks are characterized by the existence of highly connected communities and it was observed that in real networks, having high modularity [25], it is often difficult for information to flow from one community to another. This suggests that one should consider each (dense) community separately. From a result in [14], we know that it is possible to relate the minimum graph degree to the size of a perfect seed set. Namely, in any graph G 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 $2t - 2$ is a perfect seed set for G. Establishing a significant lower

bound on the size of the seed set of a dense graph has (so far) eluded our efforts. However, we recall that deciding if there exists a perfect seed set of size less than t is a hard problem in general. It would be interesting to establish to what extent such an hardness result still holds for dense graphs.

- More generally, are there class 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 in terms of node degree and threshold, in the spirit of the bound derived 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. 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.
3. 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.
4. Alina Campan, Traian Marius Truta, and Matthew Beckerich. Fast dominating set algorithms for social networks. In *MAICS*, 2015.
5. 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.
6. Ning Chen. On the approximability of influence in social networks. *SIAM Journal on Discrete Mathematics*, 23(3):1400–1415, 2009.
7. Wei Chen, Carlos Castillo, and Laks Lakshmanan. *Information and Influence Propagation in Social Networks*. Morgan & Claypool, 2013.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. Gennaro Cordasco, Luisa Gargano, Marco Mecchia, Adele A. Rescigno, and Ugo Vaccaro. A fast and effective heuristic for discovering small target sets in social networks. In *Proc. of COCOA 2015*, volume 9486, pages 193–208, 2015.

14. Gennaro Cordasco, Luisa Gargano, Adele A. Rescigno, and Ugo Vaccaro. Evangelism in social networks. In *proceedings of 27th International Workshop on Combinatorial Algorithms (To Appear)*, 2016.
15. Gennaro Cordasco, Luisa Gargano, and Adele Anna Rescigno. Influence propagation over large scale social networks. In *Proceedings of ASONAM 2015, Paris, France*, pages 1531–1538, 2015.
16. 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.
17. David Easley and Jon Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, New York, NY, USA, 2010.
18. 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.
19. 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.
20. 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.
21. David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. *Theory of Computing*, 11(4):105–147, 2015.
22. 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.
23. Jure Leskovec, Lada A. Adamic, and Bernardo A. Huberman. The dynamics of viral marketing. *ACM Trans. Web*, 1(1), May 2007.
24. Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, 2015.
25. Mark E. J. Newman. Modularity and community structure in networks. *Proc. Natl. Acad. Sci. U.S.A.* 103 (23): 8577âĂŞ8582.
26. Mark Newman. Network data, `http://www-personal.umich.edu/~mejn/netdata/`, 2015.
27. 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.
28. T. V. Thirumala Reddy and C. Pandu Rangan. Variants of spreading messages. *J. Graph Algorithms Appl.*, 15(5):683–699, 2011.
29. 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.
30. 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.
31. Reza Zafarani and Huan Liu. Social computing data repository at ASU. http://socialcomputing.asu.edu, 2009.