Time-Bounded Influence Diffusion with Incentives

Gennaro Cordasco[1(✉)], Luisa Gargano[2], Joseph G. Peters[3], Adele A. Rescigno[2], and Ugo Vaccaro[2]

[1] Department of Psychology, Università della Campania "Luigi Vanvitelli", Caserta, Italy
gennaro.cordasco@unicampania.it
[2] Department of Computer Science, Università di Salerno, Fisciano, Italy
[3] School of Computing Science, Simon Fraser University, Burnaby, Canada

Abstract. A widely studied model of influence diffusion in social networks represents the network as a graph $G = (V, E)$ with an influence threshold $t(v)$ for each node. Initially the members of an initial set $S \subseteq V$ are influenced. During each subsequent round, the set of influenced nodes is augmented by including every node v that has at least $t(v)$ previously influenced neighbours. The general problem is to find a small initial set that influences the whole network. In this paper we extend this model by using *incentives* to reduce the thresholds of some nodes. The goal is to minimize the total of the incentives required to ensure that the process completes within a given number of rounds. The problem is hard to approximate in general networks. We present polynomial-time algorithms for paths, trees, and complete networks.

1 Introduction

The *spread of influence* in social networks is the process by which individuals adjust their opinions, revise their beliefs, or change their behaviours as a result of interactions with others (see [14] and references therein quoted). For example, *viral marketing* takes advantage of peer influence among members of social networks for marketing [13]. The essential idea is that companies wanting to promote products or behaviours might try to target and convince a few individuals initially who will then trigger a cascade of further adoptions. The intent of maximizing the spread of viral information across a network has suggested several interesting optimization problems with various adoption paradigms. We refer to [5] for a recent discussion of the area. In the rest of this section, we will explain and motivate our model of information diffusion, describe our results, and discuss how they relate to the existing literature.

1.1 The Model

A social network is a graph $G = (V, E)$, where the node set V represents the members of the network and E represents the relationships among members.

Z. Lotker and B. Patt-Shamir (Eds.): SIROCCO 2018, LNCS 11085, pp. 280–295, 2018.
https://doi.org/10.1007/978-3-030-01325-7_25

We denote by $n = |V|$ the number of nodes, by $N(v)$ the neighbourhood of v, and by $d(v) = |N(v)|$ the degree of v, for each node $v \in V$.

Let $t : V \rightarrow \mathbb{N} = \{1, 2, \ldots\}$ be a function assigning integer thresholds to the nodes of G; we assume w.l.o.g. that $1 \leq t(v) \leq d(v)$ holds for all $v \in V$. For each node $v \in V$, the value $t(v)$ quantifies how hard it is to influence v, in the sense that easy-to-influence elements of the network have "low" $t(\cdot)$ values, and hard-to-influence elements have "high" $t(\cdot)$ values [16]. An *influence process in* G starting from a set $S \subseteq V$ of initially influenced nodes is a sequence of node subsets,[1]

$\mathsf{Influenced}_G[S, 0] = S$

$\mathsf{Influenced}_G[S, \ell] = \mathsf{Influenced}_G[S, \ell-1] \cup \Big\{ v : \big|N(v) \cap \mathsf{Influenced}_G[S, \ell-1]\big| \geq t(v) \Big\}$, $\ell > 0$.

Thus, in each round ℓ, the set of influenced nodes is augmented by including every uninfluenced node v for which the number of *already* influenced neighbours is at least as big as v's threshold $t(v)$. We say that v is influenced *at* round $\ell > 0$ if $v \in \mathsf{Influenced}_G[S, \ell] \setminus \mathsf{Influenced}_G[S, \ell - 1]$. A target set for G is a set S such that it will influence the whole network, that is, $\mathsf{Influenced}_G[S, \ell] = V$, for some $\ell \geq 0$.

The classical *Target Set Selection* (TSS) problem having as input a network $G = (V, E)$ with thresholds $t : V \longrightarrow \mathbb{N}$, asks for a target set $S \subseteq V$ of *minimum* size for G [1,8]. The TSS problem has roots in the general study of the spread of influence in social networks (see [5,14]). For instance, in the area of viral marketing [13], companies wanting to promote products or behaviors might try to initially convince a small number of individuals (by offering free samples or monetary rewards) who will then trigger a cascade of influence in the social network leading to the adoption by a much larger number of individuals.

In this paper, we extend the classical model to make it more realistic. It was first observed in [12] that the classical model limits the optimizer to a binary choice between zero or complete influence on each individual whereas customized incentives could be more effective in realistic scenarios. For example, a company promoting a new product may find that offering one hundred free samples is far less effective than offering a ten percent discount to one thousand people.

Furthermore, the papers mentioned above do not consider the time (number of rounds) necessary to complete the influence diffusion process. This could be quite important in viral marketing; a company may want to influence its potential customers quickly before other companies can market a competing product.

With this motivation, we formulate our model as follows. An assignment of incentives to the nodes of a network $G = (V, E)$ is a function $p : V \rightarrow \mathbb{N}_0 = \{0, 1, 2, \ldots\}$, where $p(v)$ is the amount of influence initially applied on $v \in V$. The effect of applying the incentive $p(v)$ on node v is to decrease its threshold, i.e., to make v more susceptible to future influence. It is clear that to start the process, there must be some nodes for which the initially applied influences are at least as large as their thresholds. We assume, w.l.o.g., that

[1] We will omit the subscript G whenever the graph G is clear from the context.

$0 \leq p(v) \leq t(v) \leq d(v)$. An influence process in G starting with incentives given by a function $p : V \rightarrow \mathbb{N}_0 = \{0, 1, 2, \ldots\}$ is a sequence of node subsets
$\mathsf{Influenced}[p, 0] = \{v\ :\ p(v) = t(v)\}$
$\mathsf{Influenced}[p, \ell] = \mathsf{Influenced}[p, \ell-1] \cup \Big\{v : \big|N(v) \cap \mathsf{Influenced}[p, \ell-1]\big| \geq t(v) - p(v)\Big\}$, $\ell > 0$.
The cost of the incentive function $p : V \longrightarrow \mathbb{N}_0$ is $\sum_{v \in V} p(v)$.

Let λ be a bound on the number of rounds available to complete the process of influencing all nodes of the network. The Time-Bounded Targeting with Incentives problem is to find incentives of minimum cost which result in all nodes being influenced in at most λ rounds:

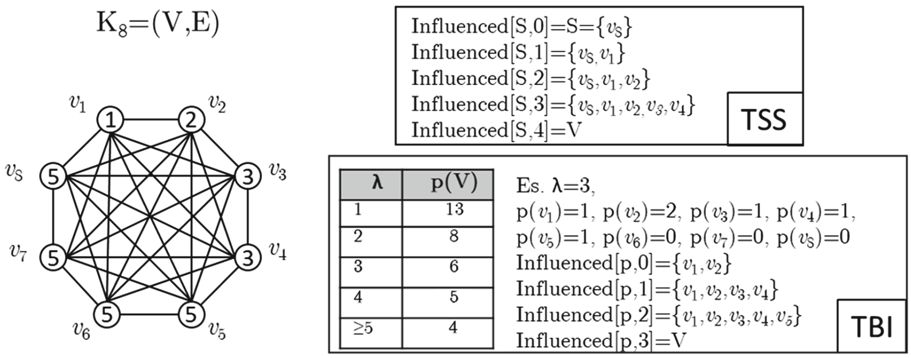
Time-Bounded Targeting with Incentives (TBI).

Instance: A network $G = (V, E)$ with thresholds $t : V \longrightarrow \mathbb{N}$ and time bound λ.

Problem: Find incentives $p : V \longrightarrow \mathbb{N}_0$ of minimum cost $\sum_{v \in V} p(v)$ s.t.

$$\mathsf{Influenced}[p, \lambda] = V.$$

Example 1. Solutions to the TBI problem can be quite different from solutions to the TSS problem for a given network. Consider a complete graph K_8 on 8 nodes with thresholds shown in Fig. 1. The optimal target set is $S{=}\{v_8\}$ which results in all nodes being influenced in 4 rounds. The TBI problem admits different optimal solutions (with different incentive functions) depending on the value of λ, as shown in Fig. 1.



Influenced[S,0]=S=$\{v_8\}$
Influenced[S,1]=$\{v_8, v_1\}$
Influenced[S,2]=$\{v_8, v_1, v_2\}$
Influenced[S,3]=$\{v_8, v_1, v_2, v_3, v_4\}$
Influenced[S,4]=V

λ	p(V)
1	13
2	8
3	6
4	5
≥5	4

Es. λ=3,
p(v_1)=1, p(v_2)=2, p(v_3)=1, p(v_4)=1,
p(v_5)=1, p(v_6)=0, p(v_7)=0, p(v_8)=0
Influenced[p,0]=$\{v_1, v_2\}$
Influenced[p,1]=$\{v_1, v_2, v_3, v_4\}$
Influenced[p,2]=$\{v_1, v_2, v_3, v_4, v_5\}$
Influenced[p,3]=V

Fig. 1. A complete graph K_8. The number inside each circle is the node threshold. Optimal solutions for the TSS problem and the TBI problem, with various values of λ, are shown.

1.2 Related Work and Our results

The study of the spread of influence in complex networks has experienced a surge of interest in the last few years. Kempe *et al.* [17] introduced the Influence

Maximization (IM) problem, where the goal is to find a subset of nodes in a social network that has cardinality bounded by a certain budget β and that could maximize the spread of influence. However, they were mostly interested in networks with randomly chosen thresholds.

Chen [6] studied the TSS problem. 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 of papers including [1–4,7–9,15,20,22] that isolated many interesting scenarios in which the problem (and variants thereof) become tractable.

The problem of maximizing the number of nodes activated within a specified number of rounds has also been studied [8,9,23]. The problem of dynamos or dynamic monopolies in graphs is essentially the target set problem with every node threshold being half its degree [21].

The Influence Maximization problem with incentives was introduced in [12]. In this model the authors assume that the thresholds are randomly chosen values in $[0,1]$ and they aim to understand how a fractional version of the Influence Maximization problem differs from the original version. To that purpose, they introduced the concept of partial influence and showed that, in theory, the fractional version retains essentially the same computational hardness as the integral version, but in practice, better solutions can be computed using heuristics in the fractional setting.

The Targeting with Partial Incentives (TPI) problem, of finding incentives $p : V \longrightarrow \mathbb{N}_0$ of minimum cost $\sum_{v \in V} p(v)$ such that all nodes are eventually influenced, was studied in [11]. Exact solutions to the TPI problem for special classes of graphs were proposed in [10,11]. Variants of the problem, in which the incentives are modelled as additional links from an external entity, were studied in [18,19]. The authors of [23] study the case in which offering discounts to nodes causes them to be influenced with a probability proportional to the amount of the discount.

It was shown in [11] that the TPI problem cannot be approximated to within a ratio of $O(2^{\log^{1-\epsilon} n})$, for any fixed $\epsilon > 0$, unless $NP \subseteq DTIME(n^{polylog(n)})$, where n is the number of nodes in the graph. As a consequence, for general graphs, the same inapproximability result still holds for the time bounded version of the problem that we study in this paper.

Theorem 1. *The TBI problem cannot be approximated to within a ratio of $O(2^{\log^{1-\epsilon} n})$, for any fixed $\epsilon > 0$, unless $NP \subseteq DTIME(n^{polylog(n)})$, where n is the number of nodes in the graph.*

Our Results. Our main contributions are polynomial-time algorithms for path, complete, and tree networks. In Sect. 2, we present a linear-time greedy algorithm to allocate incentives to the nodes of a path network. In Sect. 3, we design a $O(\lambda n \log n)$ dynamic programming algorithm to allocate incentives to the nodes of a complete network. In Sect. 4, we give an $O(\lambda^2 \Delta n\})$algorithm to allocate incentives to a tree with n nodes and maximum degree Δ.

2 A Linear-Time Algorithm for Paths

In this section, we present a greedy algorithm to allocate incentives to nodes of a path network. We prove that our algorithm is linear-time.

We denote by $L(0, n-1)$ the path with n nodes $0, \ldots, n-1$ and edges $\{(i, i+1) : 0 \leq i \leq n-2\}$. Since the threshold of each node cannot exceed its degree, we have that $t(0) = t(n-1) = 1$ and $t(i) \in \{1, 2\}$, for $i = 1, \ldots, n-2$. For $0 \leq j \leq k \leq n-1$, we denote by $L(j, k)$ the subpath induced by the nodes $j, \ldots, k$.

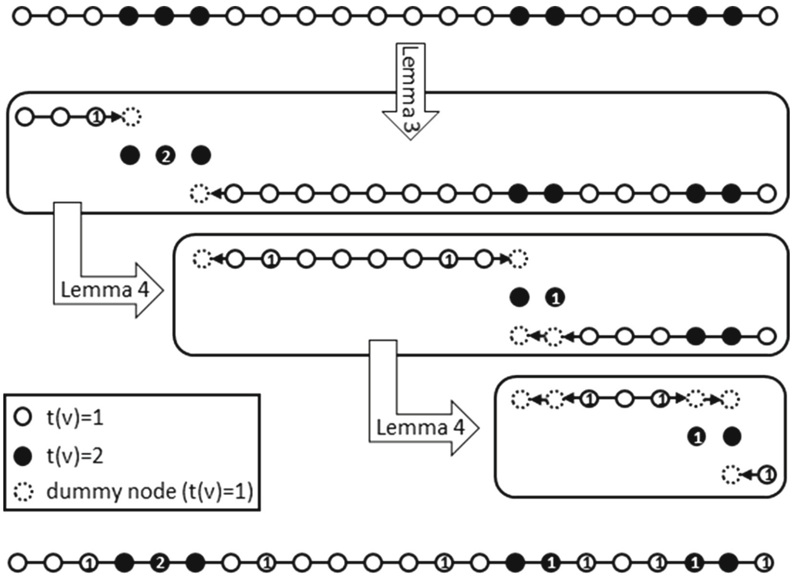


Fig. 2. An example of the execution of the Algorithm 1 on a path $L(0, 21)$ with a 2-path satisfying Lemma 3 and two 2-paths satisfying Lemma 4. Filled nodes represents nodes having threshold 2. Dashed nodes represents dummy nodes. The number inside the nodes represents the incentive assigned to the node.

Lemma 1. *Let $L(j, k)$ be a subpath of $L(0, n-1)$ with $t(j+1) = \cdots = t(k-1) = 2$ and $t(j) = t(k) = 1$. For any incentive function $p : V \to \{0, 1, 2\}$ that solves the TBI problem on $L(0, n-1)$ and for any λ,*

$$\sum_{i=j+1}^{k-1} p(i) \geq \begin{cases} k-j-2 & \text{if both } j+1 \text{ and } k-1 \text{ are influenced by } j \text{ and } k, \text{ resp.} \\ k-j-1 & \text{if either } j+1 \text{ or } k-1 \text{ is influenced by its neighbour } (j \text{ or } k) \\ k-j & \text{otherwise.} \end{cases}$$

Proof. Let p be an incentive function that solves the TBI problem on $L(0, n-1)$. For any node $i \in \{j+1, \ldots, k-1\}$, let $inf(i) \in \{0, 1, 2\}$ be the amount of influence that i receives from its neighbours in $L(0, n-1)$ during the influence process starting with p (that is, the number of i's neighbours that are influenced before round i).

For each $i = j+1, \ldots, k-1$, it must hold that $inf(i) + p(i) \geq t(i) = 2$. Hence,

$$\sum_{i=j+1}^{k-1} p(i) \geq \sum_{i=j+1}^{k-1} (2 - inf(i)) \geq 2(k-j-1) - \sum_{i=j+1}^{k-1} inf(i). \quad (1)$$

Noticing that each link in E is used to transmit influence in at most one direction, we have

$$\sum_{i=j+1}^{k-1} inf(i) \leq \begin{cases} k-j, & \text{if both } j+1 \text{ and } k-1 \text{ are influenced by } j \text{ and } k, \text{ resp.} \\ k-j-1, & \text{if either } j+1 \text{ or } k-1 \text{ is influenced by its neighbour } (j \text{ or } k) \\ k-j-2, & \text{otherwise.} \end{cases}$$

As a consequence, using Eq. (1) gives the desired result.

In the following we assume that $\lambda \geq 2$. The case $\lambda = 1$ will follow from the results in Sect. 4, since the algorithm for trees has linear time when both λ and the maximum degree are constant.

Definition 1. *We denote by $OPT(0, n-1)$ the value of an optimal solution $p : V \to \{0, 1, 2\}$ to the TBI problem on $L(0, n-1)$ in λ rounds. For any subpath $L(j,k)$ of $L(0, n-1)$, we denote by:*

(i) $OPT(j,k)$ the value $\sum_{i=j}^{k} p(i)$ where p is an optimal solution to the TBI problem on $L(j,k)$;
(ii) $OPT(j,k,\leftarrow)$ the value $\sum_{i=j}^{k} p(i)$ where p is an optimal solution to the TBI problem on $L(j,k)$ with the additional condition that the node k gets one unit of influence from $k+1$;
(iii) $OPT(j,k,\overset{\ell}{\to})$ the value $\sum_{i=j}^{k} p(i)$ where p is an optimal solution to the TBI problem on $L(j,k)$ with the additional condition that k is influenced by round $\lambda - \ell$ without getting influence from node $k+1$;
(iv) $OPT(\to, j, k)$ the value $\sum_{i=j}^{k} p(i)$ where p is an optimal solution to the TBI problem on $L(j,k)$ with the additional condition that j gets one unit of influence from $j-1$;
(v) $OPT(\overset{\ell}{\leftarrow}, j, k)$ the value $\sum_{i=j}^{k} p(i)$ where p is an optimal solution to the TBI problem on $L(j,k)$ with the additional condition that node j is influenced by round $\lambda - \ell$ without getting influence from $j-1$.

Lemma 2. *For any subpath $L(j,k)$ and for each $1 \leq \ell < \ell' \leq \lambda$:*

(1) If $t(k)=1$ then $OPT(j,k,\leftarrow) \leq OPT(j,k) \leq OPT(j,k,\overset{\ell}{\to}) \leq OPT(j,k,\overset{\ell'}{\to}) \leq OPT(j,k,\leftarrow)+1$.

(2) If $t(j) = 1$ then $OPT(\to, j, k) \leq OPT(j,k) \leq OPT(\overset{\ell}{\leftarrow}, j, k) \leq OPT(\overset{\ell'}{\leftarrow}, j, k) \leq OPT(\to, j, k)+1$.

Proof. We first prove (1). We notice that each of the first three inequalities $OPT(j,k,\leftarrow) \leq OPT(j,k)$, $OPT(j,k) \leq OPT(j,k,\overset{\ell}{\to})$, $OPT(j,k,\overset{\ell}{\to}) \leq OPT(j,k,\overset{\ell'}{\to})$ is trivially true since each solution that satisfies the assumptions of the right term is also a solution that satisfies the assumptions of the left term. It remains to show that $OPT(j, k, \underset{\rightarrow}{\ell'}) \leq OPT(j, k, \leftarrow) + 1$. Let p be a solution that gives $OPT(j, k, \leftarrow)$. Consider p' such that $p'(i) = p(i)$, for each $i = j, \ldots, k-1$ and $p'(k) = 1$. Recalling that $t(k) = 1$, we get that the cost increases by at most 1 and p' is a solution in which node k is influenced at round $0 \leq \lambda - \ell'$. A similar proof holds for (2). □

Definition 2. *$L(j, k)$, with $j+1 \leq k-1$, is called a 2-path if $t(j+1) = \ldots = t(k-1) = 2$ and $t(j) = t(k) = 1$.*

Lemma 3. *For any value of λ, if $L(j, k)$ is a 2-path with $m = k - j - 1 \neq 2$ then $OPT(0, n-1) = OPT(0, j, \underset{\rightarrow}{1}) + k - j - 2 + OPT(\underset{\leftarrow}{1}, k, n-1)$.*

Outline of Proof. The proof shows that one can always find an optimal solution in which the $m = k - j - 1$ nodes in $L(j+1, k-1)$ receive the incentives using the sequence $0(20)^*$ when m is odd and $01(20)^*$ when $m > 2$ is even. See Fig. 2 for an example of the odd case. □

Lemma 4. *For any time bound $\lambda > 1$, if $t(0) = t(1) = \ldots = t(j-1) = 1$ and $L(j, j+3)$ is a 2-path then $OPT(0, n-1)$ is equal to*

$$1 + \min\left\{OPT(0, j, \underset{\rightarrow}{1}) + OPT(\underset{\leftarrow}{2}, j+3, n-1), OPT(0, j, \underset{\rightarrow}{2}) + OPT(\underset{\leftarrow}{1}, j+3, n-1)\right\}.$$

Outline of Proof. The proof is a case analysis that shows that there is always an optimal solution in which either

1. $p(j+1) = 0, p(j+2) = 1$ or 2. $p(j+1) = 1, p(j+2) = 0$. □

Lemma 5. *For any value of λ, the minimum cost for the TBI problem on a path of n nodes having threshold 1 is $\lceil n/(2\lambda + 1) \rceil$.*

Outline of Proof. The basic idea is to break the path into subpaths of $2\lambda + 1$ nodes and assign an incentive of 1 to the middle node of each subpath and incentive 0 to the others. □

Remark 1. $OPT(j, k, \underset{\rightarrow}{\ell})$ can be obtained by solving the TBI problem on an augmented path $L(j, k+\ell)$ obtained from $L(j, k)$ by concatenating ℓ dummy nodes on the right of k with $t(k+1) = t(k+2) = \ldots = t(k+\ell) = 1$. Notice that, for $\ell \leq \lambda$, it is always possible to find an optimal assignment of incentives for the augmented path $L(j, k+\ell)$ in which all dummy nodes get incentive 0. Indeed it is possible to obtain such an assignment starting from any optimal assignment and moving the incentives from the dummy nodes to node k. An analogous observation holds for $OPT(\underset{\leftarrow}{\ell}, j, k)$.

Our algorithm iterates from left to right, identifying all of the 2-paths and, using Lemma 3 or 4 and Lemma 5, it optimally assigns incentives both to the nodes of threshold 2 and to the nodes (of threshold 1) on the left. It then removes them from the original path. Eventually, it will deal with a last subpath in which all of the nodes have threshold 1.

Algorithm 1. TBI-Path($L(0,n{-}1)$)

```
Input: A Path $L(0,n{-}1)$, thresholds $t(i) \in \{1,2\}$, $i = 0,\dots,n-1$, and a time
       bound $\lambda$.
Output: A solution $p(i) : V \to \{0,1,2\}$ of the TBI problem.
1  $i = 0$
2  while there exists a node $j$ with $t(j) = 2$ for some $i < j < n-1$ do
3      Identify the leftmost 2-path in the current path $L(i,n-1)$; let it be $L(j,k)$.
4      if $L(j,k)$ is a 2-path satisfying Lemma 3 then
5          assign incentives to the nodes $j+1,\dots,k-1$ as in Lemma 3;
6          $t(j+1) = t(k-1) = 1$;
7          obtain $p(i),\dots,p(j)$ by using Lemma 5 on $L(i,j{+}1)$ with
             $t(i) = \cdots = t(j{+}1) = 1$;
8          $i = k-1$;
9      else if $L(j,k=j+3)$ is a 2-path satisfying Lemma 4 then
10         if $j-i+2 = c(2\lambda+1)$ for some $c > 0$ then      // Case 1 of Lemma 4
11             $p(j+1) = 0$;   $p(j+2) = 1$;   $i' = j+1$;
12         else                                                // Case 2 of Lemma 4
13             $p(j+1) = 1$;   $p(j+2) = 0$;   $i' = j+2$;
14         $t(j+1) = t(j+2) = 1$;
15         obtain $p(i),\dots,p(j)$ by using Lemma 5 on $L(i,i')$ with
             $t(i) = \cdots = t(i') = 1$;
16         $i = i'$;
17 Assign incentives to $L(i,n-1)$ (with $t(i) = \ldots = t(n-1) = 1$), using Lemma 5;
18 return $p$;
```

Theorem 2. *For any time bound $\lambda > 1$, Algorithm 1 provides an optimal solution for the TBI problem on any path $L(0,n{-}1)$ in time $O(n)$.*

Proof. We show that the algorithm selects an optimal strategy according to the length and the position of the leftmost 2-path $L(j,k)$ and then iteratively operates on the subpath $L(i,n-1)$ where $i = k-1$ (one dummy node on the left) or $i = k-2$ (two dummy nodes on the left). See Fig. 2.

Let $L(i,n-1)$ be the current path and $L(j,k)$ be the leftmost 2-path. If $L(j,k)$ satisfies the hypothesis of Lemma 3, then we have

$$OPT(i,n-1) = OPT(i,j,\underset{\rightarrow}{1}) + k - j - 2 + OPT(\underset{\leftarrow}{1},k,n-1).$$

Hence, we can obtain optimal incentives for nodes $i,\dots,j$ by using the result in Lemma 5 on $L(i,j+1)$ (where $j+1$ is a dummy node). Moreover, we assign $k-j-2$ incentives to the nodes $j+1,\dots,k-1$ as suggested in Lemma 3 (i.e., $0(20)^*$ when the length of the 2-path is odd and $01(20)^*$ otherwise) and the algorithm iterates on $L(k-1,n-1)$ (where $k-1$ is a dummy node).

Now suppose that $L(j,k=j+3)$ satisfies the hypothesis of Lemma 4. We have that $OPT(i,n{-}1)$ is equal to

$$1{+}\min\Big\{OPT(i,j,\underset{\rightarrow}{1}){+}OPT(\underset{\leftarrow}{2},k,n{-}1), OPT(i,j,\underset{\rightarrow}{2}){+}OPT(\underset{\leftarrow}{1},k,n{-}1)\Big\}. \quad (2)$$

We have two cases to consider, according to the distance between i and j.
First assume that $j-i+2 = c(2\lambda+1)$ for some $c > 0$. By Lemma 5 and Remark 1 we know that in this case $OPT(i, j, \overset{2}{\rightarrow}) = OPT(i, j, \overset{1}{\rightarrow}) + 1$ and since by (2) of Lemma 2 we know that $OPT(\overset{2}{\leftarrow}, k, n-1) \leq OPT(\overset{1}{\leftarrow}, k, n-1) + 1$, we have that $OPT(i, j, \overset{1}{\rightarrow}) + OPT(\overset{2}{\leftarrow}, k, n-1)$ corresponds to the minimum of Eq. (2) and hence the solution described by case 1 in Lemma 4 (i.e., $p(j+1) = 0, p(j+2) = 1$) is optimal. Incentives to $i, \ldots, j$ are assigned exploiting the result in Lemma 5 on $L(i, j+1)$ (where $j+1$ is a dummy node) and the algorithm iterates on $L(k-2, n-1)$ (where both $k-1$ and $k-2$ are dummy nodes).
Now assume that $j - i + 2 \neq c(2\lambda + 1)$ for some $c > 0$. In this case, we have $OPT(i, j, \overset{2}{\rightarrow}) = OPT(i, j, \overset{1}{\rightarrow})$.
By (1) of Lemma 2 we know that $OPT(\overset{1}{\leftarrow}, k, n-1) \leq OPT(\overset{2}{\leftarrow}, k, n-1)$. Hence, $OPT(i, j, \overset{2}{\rightarrow}) + OPT(\overset{1}{\leftarrow}, k, n-1)$ corresponds to the minimum of Eq. (2) and the solution in case 2 in Lemma 4 (i.e., $p(j+1) = 1, p(j+2) = 0$) is optimal. Incentives to $i, \ldots, j$ are assigned using the result in Lemma 5 on $L(i, j+2)$ (considering both $j+1$ and $j+2$ as dummy nodes) and the algorithm iterates on $L(k-1, n-1)$ (where $k-1$ is a dummy node).

If there remains a last subpath of nodes of threshold one, this is solved optimally using Lemma 5.

Complexity. The identification of the 2-paths and their classification can be easily done in linear time. Then, the algorithm operates in a single pass from left to right and the time is $O(n)$. □

3 An $O(\lambda n \log n)$ Algorithm for Complete Graphs

In this section, we present an $O(\lambda n \log n)$ dynamic programming algorithm to allocate incentives to the nodes of a complete network $K_n = (V, E)$. We begin by proving that for any assignment of thresholds to the nodes of K_n, there is an optimal solution in which the thresholds of all nodes that are influenced *at* round ℓ are at least as large as the thresholds of all nodes that are influenced before round ℓ for every $1 \leq \ell \leq \lambda$.

Let K_m be the subgraph of K_n that is induced by $V_m = \{v_1, v_2, \ldots, v_m\}$. We will say that an incentive function $p : V_m \longrightarrow \mathbb{N}_0$ is *ℓ-optimal for K_m*, $1 \leq m \leq n$, $0 \leq \ell \leq \lambda$, if $\sum_{v \in V_m} p(v)$ is the minimum cost to influence all nodes in V_m in ℓ rounds.

Lemma 6. *Given K_m, thresholds $t(v_1) \leq t(v_2) \leq \ldots \leq t(v_m)$, and $1 \leq \ell \leq \lambda$, if there exists an ℓ-optimal solution for K_m that influences $k < m$ nodes by the end of round $\ell - 1$, then there is an ℓ-optimal solution that influences $\{v_1, v_2, \ldots, v_k\}$ by the end of round $\ell - 1$.*

Proof. Let p^* be an ℓ-optimal incentive function for K_m that influences a set $V_k^* = \{u_1, u_2, \ldots, u_k\}$ of k nodes of K_m by the end of round $\ell - 1$. We will show how to construct an ℓ-optimal incentive function for K_m that influences nodes $V_k = \{v_1, v_2, \ldots, v_k\}$ by the end of round $\ell - 1$ where $t(v_1) \leq t(v_2) \leq \ldots \leq t(v_k)$ and $t(v_j) \geq t(v_k)$ for $j = k+1, k+2, \ldots, m$.

Suppose that p is an incentive function for K_m that influences nodes $V_k = \{v_1, v_2, \ldots, v_k\}$ by the end of round $\ell - 1$. If V_k^* is different from V_k, then there is some $u_i \in V_k^* \backslash V_k$ and some $v_j \in V_k \backslash V_k^*$ such that $t(u_i) \geq t(v_j)$. Since v_j is influenced *at* round ℓ in the ℓ-optimal solution p^*, it must require the influence of $t(v_j) - p^*(v_j)$ neighbours. (If it required the influence of fewer neighbours, then p^* would not be ℓ-optimal.) Note that $t(v_j) - p^*(v_j) \geq 0$. Similarly, u_i requires the influence of $t(u_i) - p^*(u_i) \geq 0$ neighbours. Consider the set of nodes $V_k^* \cup \{v_j\} \backslash \{u_i\}$ and define p as follows. Choose $p(v_j)$ and $p(u_i)$ as

$$t(v_j) - p(v_j) = t(u_i) - p^*(u_i) \quad \text{and} \quad t(u_i) - p(u_i) = t(v_j) - p^*(v_j)$$

so that v_j is influenced at the same round as u_i was influenced in the ℓ-optimal solution and u_i is influenced at round ℓ. Set $p(v) = p^*(v)$ for all other nodes in K_m. The difference in value between p and p^* is

$$p(v_j) + p(u_i) - p^*(v_j) - p^*(u_i) = 0$$

We can iterate until we find an ℓ-optimal solution that influences $\{v_1, v_2, \ldots, v_k\}$ by the end of round $\ell - 1$. □

By Lemma 6, our algorithm can first sort the nodes by non-decreasing threshold value w.l.o.g. The sorting can be done in $O(n)$ time using counting sort because $1 \leq t(v) \leq n - 1 = d(v)$ for all $v \in V$. In the remainder of this section, we assume that $t(v_1) \leq t(v_2) \leq \ldots \leq t(v_n)$.

Let $Opt_\ell(m)$ denote the value of an ℓ-optimal solution for K_m, $1 \leq m \leq n$, $0 \leq \ell \leq \lambda$. Any node v that is influenced at round 0 requires incentive $p(v) = t(v)$ and it follows easily that

$$Opt_0(m) = \sum_{i=1}^{m} t(v_i),\ 1 \leq m \leq n. \tag{3}$$

Now consider a value $Opt_\ell(m)$ for some $1 \leq m \leq n$ and $1 \leq \ell \leq \lambda$. If exactly j nodes, $1 \leq j \leq m$, are influenced by the end of round $\ell - 1$ in an ℓ-optimal solution for K_m, then each of the $m - j$ remaining nodes in V_m has j influenced neighbours at the beginning of round ℓ and these neighbours are $v_1, v_2, \ldots, v_j$ by Lemma 6. For such a remaining node v to be influenced at round ℓ, either $t(v) \leq j$ or v has an incentive $p(v)$ such that $t(v) - p(v) \leq j$. It follows that

$$Opt_\ell(m) = \min_{1 \leq j \leq m} \left\{ Opt_{\ell-1}(j) + \sum_{i=j+1}^{m} \max\{0, t(v_i) - j\} \right\},\ 1 \leq m \leq n. \tag{4}$$

We will use $Ind_\ell(m)$ to denote the index that gives the optimal value $Opt_\ell(m)$, that is,

$$Ind_\ell(m) = \arg\min_{1 \leq j \leq m} \left\{ Opt_{\ell-1}(j) + \sum_{i=j+1}^{m} \max\{0, t(v_i) - j\} \right\},\ 1 \leq m \leq n. \tag{5}$$

A dynamic programming algorithm that directly implements the recurrence in Eqs. (3) and (4) will produce the optimal solution value $Opt_\lambda(n)$ in time $O(\lambda n^3)$. We can reduce the complexity by taking advantage of some structural properties.

Lemma 7. *For any* $1 \leq \ell \leq \lambda$, *if* $k < m$ *then* $Ind_\ell(k) \leq Ind_\ell(m)$, $1 \leq k \leq n-1$, $2 \leq m \leq n$.

Outline of Proof. The lemma always holds when $k < Ind_\ell(m)$. Assuming that $Ind_\ell(k) > Ind_\ell(m)$ when $k \geq Ind_\ell(m)$ leads to a contradiction.

Theorem 3. *For any complete network* $K_n = (V, E)$, *threshold function* $t : V \longrightarrow \mathbb{N}$, *and* $\lambda \geq 1$, *the TBI problem can be solved in time* $O(\lambda n \log n)$.

Proof. Our dynamic programming algorithm computes two $n \times (\lambda + 1)$ arrays $VALUE$ and $INDEX$ and returns a solution p of n incentives. $VALUE[m, \ell] = Opt_\ell(m)$ is the value of an ℓ-optimal solution for K_m (for a given threshold function $t : V \longrightarrow \mathbb{N}$), and $INDEX[m, \ell] = Ind_\ell(m)$ is the index that gives the optimal value, $1 \leq m \leq n$, $0 \leq \ell \leq \lambda$.

The array entries are computed column-wise starting with column 0. The entries in column $VALUE[*, 0]$ are sums of thresholds according to (3) and the indices in $INDEX[*, 0]$ are all 0, so these columns can be computed in time $O(n)$. In particular, $VALUE[j, 0] = \sum_{i=1}^{j} t(v_i)$, $j = 1, \ldots, m$.

Suppose that columns $1, 2, \ldots, \ell - 1$ of $VALUE$ and $INDEX$ have been computed according to (4) and (5) and consider the computation of column ℓ of the two arrays. To compute $INDEX[m, \ell]$ for some fixed m, $1 \leq m \leq n$, we define a function

$$A(j) = Opt_{\ell-1}(j) + \sum_{i=j+1}^{m} \max\{0, t(v_i) - j\}, \quad 1 \leq j \leq m$$

and show how to compute each $A(j)$ in $O(1)$ time.
By (5), $Ind_\ell(m) = \arg\min\{A(j) \mid 1 \leq j \leq m\}$.

First we compute an auxiliary vector a where $a[j]$ contains the smallest integer $i \geq 1$ such that $t(v_i) \geq j$, $1 \leq j \leq n$. This vector can be precomputed once in $O(n)$ time because the nodes are sorted by non-decreasing threshold value. Furthermore, the vector a together with the entries in column $VALUE[*, 0]$ allow the computation of $\sum_{i=j+1}^{m} \max\{0, t(v_i) - j\}$ in $O(1)$ time for each $1 \leq j \leq n$. Since $Opt_{\ell-1}(j) = VALUE[j, \ell-1]$ has already been computed, we can compute $A(j)$ in $O(1)$ time. The values $Opt_\ell(m) = VALUE[m, \ell]$ can also be computed in $O(1)$ time for each $1 \leq m \leq n$ given $Ind_\ell(m) = INDEX[m, \ell]$, vector a, and column $VALUE[*, 0]$. The total cost so far is $O(\lambda n)$. It remains to show how to compute each column $INDEX[*, \ell]$ efficiently.

The following algorithm recursively computes the column $INDEX[m, \ell]$, $1 \leq m \leq n$ assuming that columns $0, 1, 2, \ldots, \ell - 1$ of $INDEX$ and $VALUE$ have already been computed. The algorithm also assumes that two dummy rows have been added to array $INDEX$ with $INDEX[0, \ell] = 1$ and $INDEX[n + 1, \ell] = n$,

$0 \leq \ell \leq \lambda$, to simplify the pseudocode. The initial call of the algorithm is COMPUTE-INDEX$(1, n)$.

We claim that algorithm COMPUTE-INDEX$(1, n)$ correctly computes the values $\mathit{INDEX}[m, \ell]$ for $1 \leq m \leq n$. First, it can be proved by induction that when we call COMPUTE-INDEX(x, y), the indices $\mathit{INDEX}[x-1, \ell]$ and $\mathit{INDEX}[y+1, \ell]$ have already been correctly computed. By Lemma 7, $\mathit{Ind}_\ell(x-1) \leq \mathit{Ind}_\ell(\lceil \frac{x+y}{2} \rceil) \leq \mathit{Ind}_\ell(y+1)$, so the algorithm correctly searches for $\mathit{INDEX}[m, \ell]$ between $\mathit{INDEX}[x-1, \ell]$ and $\mathit{INDEX}[y+1, \ell]$.

Algorithm 2. COMPUTE-INDEX(x, y)

```
Input: Indices $x, y$.
Output: The values $\mathit{INDEX}[i, \ell]$ for $i = x, \ldots y$.
1 if $x \leq y$ then        // Assume that $\mathit{INDEX}[0, \ell] = 1$ and $\mathit{INDEX}[n+1, \ell] = n$
2     $m = \lceil \frac{x+y}{2} \rceil$;
3     $\mathit{INDEX}[m, \ell] =$
        $\arg\min \{A(j) \mid \mathit{INDEX}[x-1, \ell] \leq j \leq \min\{\mathit{INDEX}[y+1, \ell], m\}\}$;
4     COMPUTE-INDEX$(x, m-1)$;
5     COMPUTE-INDEX$(m+1, y)$;
```

It is not hard to see that the height of the recursion tree obtained calling COMPUTE-INDEX$(1, n)$ is $\lceil \log(n+1) \rceil$. Furthermore, the number of values $A(j)$ computed at each level of the recursion tree is $O(n)$ because the ranges of the searches in line 3 of the algorithm do not overlap (except possibly the endpoints of two consecutive ranges) by Lemma 7. Thus, the computation time at each level is $O(n)$, and the computation time for each column ℓ is $O(n \log n)$. After all columns of VALUE and INDEX have been computed, the value of the optimal solution will be in $\mathit{VALUE}[n, \lambda]$. The round during which each node is influenced and the optimal function p of incentives can then be computed by backtracking through the array INDEX in time $O(\lambda + n)$. The total complexity is $O(\lambda n \log n)$. $\square$

4 A Polynomial-Time Algorithm for Trees

In this section, we give an algorithm for the TBI problem on trees. Let $T = (V, E)$ be a tree having n nodes and the maximum degree Δ. We will assume that T is rooted at some node r. Once such a rooting is fixed, for any node v, we denote by T_v the subtree rooted at v, and by $C(v)$ the set of children of v. We will develop a dynamic programming algorithm that will prove the following theorem.

Theorem 4. *For any $\lambda > 1$, the TBI problem can be solved in time $O(n\lambda^2\Delta)$ on a tree having n nodes and maximum degree Δ.*

The rest of this section is devoted to the description and analysis of the algorithm that proves Theorem 4. The algorithm performs a post-order traversal of the tree T so that each node is considered after all of its children have been processed. For each node v, the algorithm solves some TBI problems on the subtree T_v, with some restrictions on the node v regarding its threshold and the round during which it is influenced. For instance, in order to compute some of these values we will consider not only the original threshold $t(v)$ of v, but also the reduced threshold $t'(v) = t(v) - 1$ which simulates the influence of the parent node.

Definition 3. *For each node $v \in V$, integers $\ell \in \{0, 1, \ldots, \lambda\}$, and $t \in \{t'(v), t(v)\}$, let us denote by $P[v, \ell, t]$ the minimum cost of influencing all of the nodes in T_v, in at most λ rounds, assuming that*

- *the threshold of v is t, and for every $u \in V(T_v) \setminus \{v\}$, the threshold of u is $t(u)$;*
- *v is influenced by round ℓ in T_v and is able to start influencing its neighbours by round $\ell + 1$.*[2]

Formally the value of $P[v, \ell, t]$ corresponds to $P[v, \ell, t] = \min_{\substack{p: T_v \to \mathbb{N}_0,\ \mathsf{Influenced}_{T_v}[p, \lambda] = T_v \\ |C(v) \cap \mathsf{Influenced}_{F(v,d)}[p, \ell - 1]| \geq t - p(v)}} \left\{ \sum_{v \in T_v} p(v) \right\}$ We set $P[v, \ell, t] = \infty$ when the above problem is infeasible. Denoting by $p_{v,\ell,t} : V(T_v) \to \mathbb{N}_0$ the incentive function attaining the value $P[v, \ell, t]$, the parameter ℓ is such that:

1. if $\ell = 0$ then $p_{v,\ell,t}(v) = t$,
2. otherwise, v's children can influence v at round ℓ, i.e. $|\{C(v) \cap \mathsf{Influenced}[p_{v,\ell,t}, \ell - 1]\}| \geq t - p_{v,\ell,t}(v)$.

Remark 2. It is worthwhile mentioning that $P[v, \ell, t]$ is monotonically non-decreasing in t. However, $P[v, \ell, t]$ is not necessarily monotonic in ℓ.

Indeed, partition the set $C(v)$ into two sets: $C'(v)$, which contains the c children that influence v, and $C''(v)$, which contains the remaining $|C(v)| - c$ children that may be influenced by v. A small value of c may require a higher cost on subtrees rooted at a node $u \in C'(v)$, and may save some budget on the remaining subtrees; the opposite happens for a large value of c.

The minimum cost to influence the nodes in T in λ rounds follows from decomposing the optimal solution according to the round on which the root is influenced and can then be obtained by computing

$$\min_{0 \leq \ell \leq \lambda} P[r, \ell, t(r)]. \tag{6}$$

[2] Notice that this does not exclude the case that v becomes an influenced node at some round $\ell' < \ell$.

We proceed using a post-order traversal of the tree, so that the computations of the various values $P[v, \ell, t]$ for a node v are done after all of the values for v's children are known. For each leaf node v we have

$$P[v, \ell, t] = \begin{cases} 1 & \text{if } \ell = 0 \text{ and } t = t(v) = 1 \\ 0 & \text{if } 1 \le \ell \le \lambda \text{ and } t = t(v) - 1 = 0 \\ \infty & \text{otherwise.} \end{cases} \tag{7}$$

Indeed, a leaf v with threshold $t(v) = 1$ is influenced in the one-node subtree T_v only when either $p_{v,\ell,t}(v) = 1$ $(\ell = 0)$, or for some $1 \le \ell \le \lambda$, it is influenced by its parent (i.e., the residual threshold $t = t(v) - 1 = 0$).

For any internal node v, we show how to compute each value $P[v, \ell, t]$ in time $O(d(v) \cdot t \cdot \lambda)$.

In the following we assume that an arbitrary order has been fixed on the $d = d(v) - 1$ children of any node v, that is, we denote them as $v_1, v_2, \ldots, v_d$, according to the fixed order. Also, we define $F(v, i)$ to be the forest consisting of the subtrees rooted at the first i children of v. We will also use $F(v, i)$ to denote the set of nodes it includes.

Definition 4. *Let v be a node with d children and let $\ell = 0, 1, \ldots, \lambda$. For $i = 0, \ldots, d$, $j = 0, 1, \ldots, t(v)$, we define $A_{v,\ell}[i, j]$ (resp. $A_{v,\ell}[\{i\}, j]$) to be the minimum cost for influencing all nodes in $F(v, i)$, (resp. T_{v_i}) within λ rounds, assuming that:*

(i) if $\ell \ne \lambda$, at time $\ell + 1$ the threshold of v_k is $t'(v_k)$, for each $k = 1, \ldots, i$;
(ii) if $\ell \ne 0$, at least j nodes in $\{v_1, v_2, \ldots, v_i\}$ (resp. $\{v_i\}$) are influenced by round $\ell - 1$, that is

$$|\{v_1, v_2, \ldots, v_i\} \cap \mathsf{Influenced}[\pi_{v,\ell,i,j}, \ell - 1]| \ge j,$$

where $\pi_{v,\ell,i,j} : F(v, i) \to \mathbb{N}_0$ denotes the incentive function attaining $A_{v,\ell}[i, j]$.

We also define $A_{v,\ell}[i, j] = \infty$ when the above constraints are not satisfiable.

By decomposing a solution according to how many nodes in $C(v)$ are influenced prior to the root v being influenced and denoting this number as j, the remaining cost to influence the root v is $t - j$ Hence, we can easily write $P[v, \ell, t]$ in terms of $A_{v,\ell}[d, j]$ as follows.

Lemma 8. *For each node v with d children, each $\ell = 0, \ldots, \lambda$ and each $t \in \{t(v), t'(v)\}$*

$$P[v, \ell, t] = \begin{cases} t + A_{v,0}[d, 0] & \text{if } \ell = 0 \\ \min_{0 \le j \le t} \left\{ t - j + A_{v,\ell}[d, j] \right\} & \text{otherwise.} \end{cases} \tag{8}$$

Lemma 9. *For each node v, each $t \in \{t(v), t'(v)\}$, and each $\ell = 1, \ldots, \lambda$, it is possible to compute $A_{v,\ell}[d, t]$, as well as $A_{v,0}[d, 0]$, recursively in time $O(\lambda d t)$ where d is the number of children of v.*

Outline of Proof. The proof shows that the values $A_{v,0}[d,0]$ and $A_{v,\ell}[d,t]$ can be computed, in time $O(\lambda d)$ and $O(\lambda dt)$ respectively, using the following recursive equations. Let $M(\ell_1,\ell_2,t) = \min_{\ell_1 \le \ell' \le \ell_2}\{P[v_i,\ell',t]\}$ we have $A_{v,0}[d,0] = \sum_{v_i \in C(v)} \min\{P[v_i,0,t(v_i)], M(1,\lambda,t'(v_i))\}$,

$$A_{v,\ell}[i,j]=\begin{cases} 0, & \text{if } i=j=0 \\ \infty, & \text{if } i<j \\ \min\Big\{A_{v,\ell}[i-1,j-1]+M(0,\ell-1,t(v_i)), & \\ A_{v,\ell}[i-1,j]+\min\Big\{M(0,\ell,t(v_i)),M(\ell+1,\lambda,t'(v_i))\Big\}\Big\}, & \text{otherwise.} \end{cases}$$

□

Lemmas 8 and 9 imply that for each $v \in V$, for each $\ell = 0,\dots,\lambda$, and $t \in \{t'(v),t(v)\}$, the value $P[v,\ell,t]$ can be computed recursively in time $O(\lambda d(v)t(v))$. Hence, the value in (6) can be computed in time $\sum_{v\in V} O(\lambda d(v)t(v)) \times O(\lambda) = O(\lambda^2 \Delta) \times \sum_{v\in V} O(d(v)) = O(\lambda^2\Delta n)$, where Δ is the maximum node degree. Standard backtracking techniques can be used to compute the (optimal) influence function p^* that influences all of the nodes in the same $O(\lambda^2\Delta n)$ time.

References

1. Ackerman, E., Ben-Zwi, O., Wolfovitz, G.: Combinatorial model and bounds for target set selection. Theor. Comput. Sci. **411**, 4017–4022 (2010)
2. Ben-Zwi, O., Hermelin, D., Lokshtanov, D., Newman, I.: Treewidth governs the complexity of target set selection. Discret. Optim. **8**, 87–96 (2011)
3. Chopin, M., Nichterlein, A., Niedermeier, R., Weller, M.: Constant thresholds can make target set selection tractable. In: Even, G., Rawitz, D. (eds.) MedAlg 2012. LNCS, vol. 7659, pp. 120–133. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34862-4_9
4. Coja-Oghlan, A., Feige, U., Krivelevich, M., Reichman, D.: Contagious sets in expanders. In: Proceedings of SODA 2015, pp. 1953–1987 (2015)
5. Chen, W., Lakshmanan, L.V.S., Castillo, C.: Information and Influence Propagation in Social Networks. Morgan & Claypool, San Rafael (2013)
6. Chen, N.: On the approximability of influence in social networks. SIAM J. Discrete Math. **23**, 1400–1415 (2009)
7. Chiang, C.-Y., Huang, L.-H., Li, B.-J., Wu, J., Yeh, H.-G.: Some results on the target set selection problem. Journal of Comb. Opt. **25**(4), 702–715 (2013)
8. Cicalese, F., Cordasco, G., Gargano, L., Milanič, M., Peters, J., Vaccaro, U.: Spread of influence in weighted networks under time and budget constraints. Theor. Comput. Sci. **586**, 40–58 (2015)
9. Cicalese, F., Cordasco, G., Gargano, L., Milanič, M., Vaccaro, U.: Latency-Bounded target set selection in social networks. Theor. Comput. Sci. **535**, 1–15 (2014)
10. Cordasco, G., Gargano, L., Rescigno, A.A.: On finding small sets that influence large networks. Soc. Netw. Anal. Min. **6**(94) (2016)

11. Cordasco, G., Gargano, L., Rescigno, A.A., Vaccaro, U.: Optimizing spread of influence in social networks via partial incentives. In: Scheideler, C. (ed.) Structural Information and Communication Complexity. LNCS, vol. 9439, pp. 119–134. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25258-2_9
12. Demaine, E.D., et al.: How to influence people with partial incentives. In: Proceedings of WWW 2014, pp. 937–948 (2014)
13. Domingos, P., Richardson, M.: Mining the network value of customers. In: Proceedings of 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 57–66 (2001)
14. Easley, D., Kleinberg, J.: Networks, Crowds, and Markets: Reasoning About a Highly Connected World. Cambridge University Press, Cambridge (2010)
15. Gargano, L., Hell, P., Peters, J.G., Vaccaro, U.: Influence diffusion in social networks under time window constraints. Theor. Comput. Sci. **584**, 53–66 (2015)
16. Granovetter, M.: Thresholds models of collective behaviors. Am. J. Sociol. **83**(6), 1420–1443 (1978)
17. Kempe, D., Kleinberg, J.M., Tardos, E.: Maximizing the spread of influence through a social network. In: Proceedings of 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 137–146 (2003)
18. Lafond, M., Narayanan, L., Wu, K.: Whom to befriend to influence people. In: Suomela, J. (ed.) SIROCCO 2016. LNCS, vol. 9988, pp. 340–357. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48314-6_22
19. Narayanan, L., Wu, K.: How to choose friends strategically. In: Das, S., Tixeuil, S. (eds.) SIROCCO 2017. LNCS, vol. 10641, pp. 283–302. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-72050-0_17
20. Nichterlein, A., Niedermeier, R., Uhlmann, J., Weller, M.: On tractable cases of target set selection. Soc. Netw. Anal. Min. **3**(2), 233–256 (2013)
21. Peleg, D.: Local majorities, coalitions and monopolies in graphs: a review. Theor. Comput. Sci. **282**, 231–257 (2002)
22. Reddy, T.V.T., Rangan, C.P.: Variants of spreading messages. J. Graph Algorithms Appl. **15**(5), 683–699 (2011)
23. Liu, X., Yang, Z., Wang, W.: Exact solutions for latency-bounded target set selection problem on some special families of graphs. Discret. Appl. Math. **203**(C), 111–116 (2016)