On the constrained maximization of influence over a directed tree structure

Niccolò Di Marco, Andrea Frosini and Shinichi Nakano

Abstract

Maximizing influence in networks is a widely studied problem with applications across various domains. In this paper, we approach this problem by considering influence maximization in directed tree structures. Given a tree T, each node is associated with a binary label, obtaining $1-$nodes and $0-$nodes. The influence of $1-$nodes over T is then computed as the total number of $0-$nodes in their neighbourhood. Given an integer k, our objective is to identify a subset of k $1-$nodes that maximizes the influence over T. After analysing heuristic approaches and their limitations, we present a dynamic programming algorithm for solving this problem exactly when the maximum degree and k are fixed. Our findings provide new insights into influence maximization in directed tree structures and establish a foundation for further exploration of constrained optimization problems.

1 Introduction

Maximizing influence in network structures is a fundamental problem in various domains, including graph theory [1, 2], epidemiology [3] and information dissemination [4, 5, 6, 7].

Generally speaking, the problem asks to find a set of nodes (also called *seed nodes*) that maximize the expected influence over the tree, where influence is usually computed as the number of infected nodes reached by the seed ones.

Notably, this problem has been extensively studied, with prior works proving its NP-hardness in the general setting and establishing that the best approximation algorithm is the greedy one [8, 9].

In this work, we focus on a related problem: influence maximization over directed tree structures, where nodes are labeled with a binary state, and influence is measured in terms of the number of affected neighbors by one class in another. Given a tree structure and a constraint on the number of influential nodes, our goal is to determine the labeling that maximizes the overall influence.

Notably, previous studies [4] have explored related optimization problems in tree-based structures, finding a linear-time algorithm to solve the unconstrained version of this problem, i.e., where any number of influential nodes can be selected, through dynamic programming techniques.

However, the constrained variant, in which exactly k influential nodes must be chosen, presents additional computational challenges and remains an open problem.

In this work, given a directed tree T, we formally define and approach the *Constrained Influence Maximization Problem* $CIMP(T, k)$. We propose several heuristics and recursive methods for its solution, analyzing their theoretical properties and practical performance.

The work is organized as follows: first, we provide a family of greedy heuristics, proving its inefficacy to find the optimal solution in the general case. Then, we provide a dynamic programming approach to find the optimal labeling in $O(|V|(k+d)^{\min\{k,d\}})$ time, being d the maximum degree of the nodes of the tree. Through theoretical analysis and empirical validation, we provide insights into the efficiency and effectiveness of our proposed approaches.

2 Definitions

We consider a directed tree $T = (V, E)$, where V is the set of nodes and E is the set of directed edges. In general, each tree will be rooted in a single root r and $|T| = |V|$. For simplicity, we denote the out-neighbourhood of a node v (without v itself) as $N(v)$ and its out-degree as $d(v)$. We suppose each node v is associated with a binary label l_v. To keep it general, we say that v is a $1-$node if $l_v = 1$ and a $0-$node otherwise. Therefore, the labeling determines a partition of the nodes $V = V_1^l \cup V_0^l$, where $V_i^l = \{v \in V \mid l_v = i\}$ is the subset of $i-$nodes. Finally, we refer to the number k_l of $1-$nodes in the labeling l as its cardinality, i.e., $k_l = |V_1^l|$ (if the related label is obvious, we will omit it).

Here, we are interested in defining a measure of influence of $1-$nodes over $0-$nodes. To do that, we can specialize the out-degree of a node using $d_0^l(v) = |\{w \in N(v) \mid l_w = 0\}|$, i.e. the number of $0-$nodes in the out-neighbourhood of v. Please note that, when the context is clear, we will omit the l apex. Then, the influence of a configuration (T, l) is computed as

$$I(T, l) = \sum_{v \in V_1^l} d_0^l(v) \tag{1}$$

In this work, we study the following problem:

Constrained Influence Maximization Problem ($CIMP(T, k)$):
Provided a directed tree T and an integer $0 \leq k \leq |T|$, find a labeling $\bar{l}$ such that $I(T, \bar{l}) = \max_l I(T, l)$, with $k = |V_1^l|$.

Example 1. *Figure 1 shows two labelings on the same tree: the left one has six (red-colored)* $1-$*nodes. In that case,* $I(T, l) = 3$*. On the right, three 1-nodes provide* $I(T, l) = 5$*.*

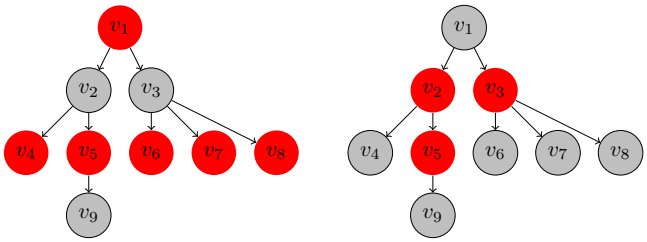


Figure 1: Two examples of trees and their labelings. From here on, the red nodes represent 1−nodes. In more details, on the left v_1, v_4, v_5, v_6, v_7 and v_8 are 1−nodes. On the right, v_2, v_3 and v_5 are 1−nodes.

Remark 1. *For a given tree T there may exist different labelings, possibly with a different number of 1-nodes, providing the same (maximal) influence. For example, consider the tree depicted in Fig. 1. The labeling $l = \{v_2, v_3, v_5\}$ and $l' = \{v_2, v_3\}$ provide the same influence $I(T, l) = I(T, l') = 5$.*

Remark 2. *We point out that, when $k \in \{0, 1, |V|\}$; $CIMP(T, k)$ admits only trivial solutions: in the first case no 1-nodes are present in T, so the influence is 0, in the second case the only 1-node of T is chosen among those having maximum degree, while, in the latter, all the nodes are 1-nodes and $I(T, l) = 0$. Therefore, the problem acquires relevance when $1 < k < |T|$.*

3 An heuristic approach to $CIMP(T, k)$

To start our study of $CIMP(T, k)$, we define the so-called *switch* operator. Let l be a labeling of the nodes of a given directed tree T and let v and w be two of its nodes. The operator $switch(v, w)$ creates a new label $\bar{l}$ such that $\bar{l}_v = l_w$, $\bar{l}_w = l_v$, and $\bar{l}_u = l_u$ for each $u \in V$, $u \neq v, w$.

The switch operator modifies a label l by exchanging the values of two nodes v and w.

To understand its functionality, consider two nodes $v \in V_1^l$ and $w \in V \setminus V_1^l$. We formally define the concept of *increment* $\Delta(v, w)$ as:

$$\Delta(v, w) := d_0^{\bar{l}}(w) - d_0^l(v) - p_{\bar{l}}(w) + p_l(v), \tag{2}$$

where $\bar{l}$ is the label resulting from $switch(v, w)$ and $p_l(x)$ is a binary variable equal to one if and only if the parent of x is a 1−node, according to label l. Equation (2) defines the increment obtained in the influence after one application of *switch*.

The motivation is easily obtained: $d_0^{\bar{l}}(w)$ counts the number of 0−nodes in the out-neighbour of w that has to be added to the total influence of $\bar{l}$, while $d_0^l(v)$ are the out-neighbour nodes of v that have to be subtracted when it changes into a 0-node. Moreover, we must add 1 to the influence if the parent

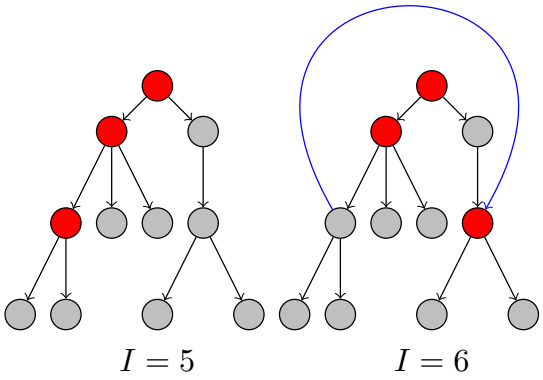


Figure 2: The operator switch exchanges the labels of two nodes producing an increment in the influence of the new label.

of v is a 1-node, i.e., if $p_l(v) = 1$. In fact, when v is changed to a 0-node, its parent gains a new 0-node in its neighborhood. Similarly, we subtract 1 from the influence if the parent of w is a 1-node, i.e., if $p_{\bar{l}}(w) = 1$.

Therefore, if $\Delta(v, w) > 0$, then $switch(v, w)$ assures that $I_{\bar{l}} > I_l$. Figure 2 shows an example of an application of *switch* that leads to an increment in the influence.

3.1 First heuristic: *Treelabeling*

We propose the heuristic *TreeLabeling* (Algorithm 1) that relies on *TrySwitch* (Algorithm 2) to obtain an approximation of an optimal labeling l having cardinality k.

Algorithm 1 *TreeLabeling*

Require: $T = (V, E)$, k;
Ensure: a labeling l of T having cardinality k;
  for $i = 1 : k$ **do** ▷ Place the first k nodes
    $W = \{w \in V \setminus V_1^l \ : \ d_0(w) - p_l(w) = max_{u \in V} \left(d_0(u) - p_l(u)\right)\}$;
    Select randomly a node $v \in W$;
    $l_v = 1$;
  end for
  Return $l = TrySwitch(T, l)$;

Indeed, the heuristic performs a greedy approach to the problem: Algorithm 1 first labels with 1 the k nodes with maximal influence, then, it applies Algorithm 2 to the obtained label to increase the influence through a sequence of *switch* applications until no further increment occur. Obviously, in case of pairs

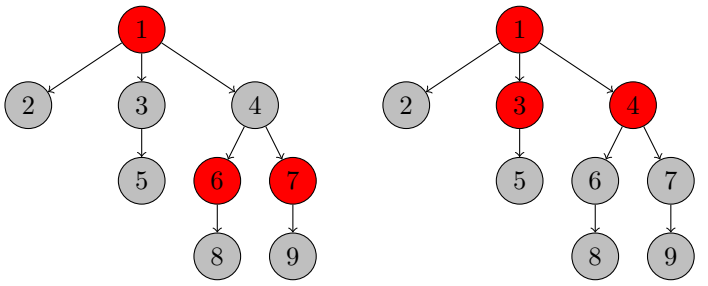
Algorithm 2 *TrySwitch*

Require: $T = (V, E)$ and one of its labels l;
Ensure: a new label $\bar{l}$;
  $check = TRUE$
  while *check* **do** ▷ Try to apply *switch*
    $check = FALSE$;
    $S = \emptyset$;
    for (u, v) such that $u \in V_1^l$ and $v \in V \setminus V_1^l$ **do**
      if $\Delta(u, v) > 0$ **then**
        $check = TRUE$;
        $S = S \cup \{(u, v)\}$;
      end if
      choose randomly $(u, v) \in S$ such that $\Delta(u, v) \geq \Delta(u', v')$, for each $(u', v') \in S$;
      $l = switch(u, v)$;
    end for
  end while
  Return l;

of nodes with equal maximal increment, the algorithm randomly selects one of them to be switched. Therefore, it is possible that two calls on the same tree could return different labels.

Unfortunately, this simple heuristic does not provide an optimal label, as witnessed by the following example.

Example 2. *Consider the directed tree depicted in Figure 3, and set $k = 3$. The label in (a) shows the (only) optimal solution while the label in (b) shows the output of the heuristic TreeLabeling.*



(a) (b)

Figure 3: (a) optimal labeling. (b) labeling obtained using *TreeLabeling*.

In particular, the algorithm follows these steps:

1. $l_v = 0$ *for each* $v \in V$ $(I = 0)$;

2. $l_1 = 1$ *(*$I = 3$*);*

3. $l_4 = 1$ *(*$I = 4$*);*

4. $l_3 = 1$ *(*$I = 4$*).*

5. *Since applying switch does not increase the influence, the algorithm stops.*

Configuration (a) *obtains an influence equal to* 5*, while* (b) *gets only* 4*, providing a counterexample to the heuristic optimality.*

3.2 A slightly extended version of *TreeLabeling*

To extend the class of the instances solved by (Algorithm 1), we propose a second heuristic *ExtTreeLabeling* (Algorithm 3), that acts after an initial call of *TreeLabeling* whose output is a labeling l of cardinality k and such that $\Delta(u, v) \leq 0$, for all $(v, u) \in V_1^l \times \left(V \setminus V_1^l\right)$.

Algorithm 3 *ExtTreeLabeling*

Require: $T = (V, E)$, k;
Ensure: a labeling l of T of cardinality k;
  $l =$ *TreeLabeling*(T, k);
  for (u, v) such that $u \in V_1^l$ and $v \in V \setminus V_1^l$ and $\Delta(v, u) = 0$ **do**
    $l_1 = switch(u, v)$;
    $l_2 = TrySwitch(T, l_1)$;
    if $l_1 \neq l_2$ **then**;
      $l = l_2$; ▷ the influence has been increased
    end if
  end for
  Return l;

For each couple (v, u) of nodes of T such that $\Delta(v, u) = 0$, the operator $switch(v, u)$ produces an equivalent (from the influence point of view) label, indicated as l_1 in Algorithm 3. If the call $TrySwitch(T, l_1)$ returns a different labeling l_2 with increased influence, i.e., $l_1 \neq l_2$, then l is updated to l_1, otherwise, l is preserved and another couple of tree nodes is considered until no further couples are present. Note that Algorithm 3 produces the optimal label in the situation depicted in Figure 3, allowing us to move from the labeling (b) to the labeling (a).

Although Algorithm 3 produces the optimal label in a wider class of instances than Algorithm 1, there still exists some cases where this strategy fails, as witnessed in the next example.

Example 3. *Consider the directed graph* T *depicted in Figure 4. We consider* $ExtTreeLabeling(T, 5)$.

This call first initializes $l_v = 0$ *for each* $v \in V$ *(*$I = 0$*). Then it sets* $l_1 = 1$ *(*$I = 3$*),* $l_2 = 1$ *(*$I = 5$*), and* $l_{10} = 1$ *(*$I = 7$*). Since the addition of further 1-nodes does not increase the influence, the algorithm randomly chooses two nodes*

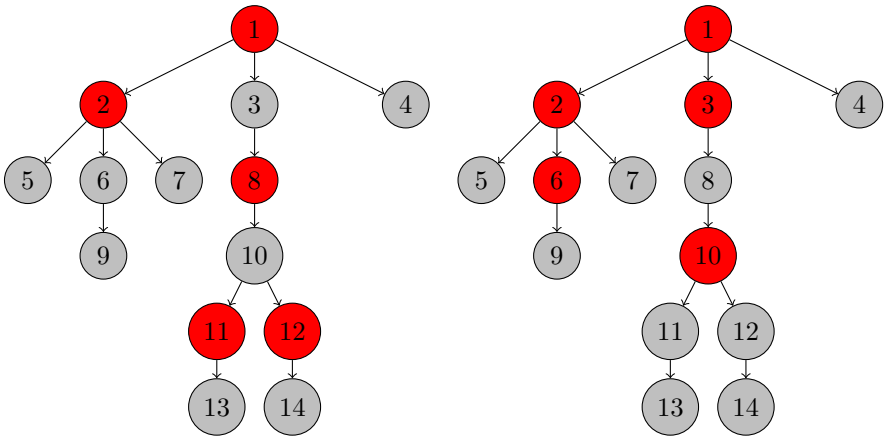


Figure 4: A directed tree with two different labelings: in (a) the (only) optimal one ($I = 8$) obtained by *ExtTreeLabeling*$(T, 5)$, while in (b) the labeling obtained by *TreeLabeling*$(T, 5)$ ($I = 7$).

to reach a labeling l of cardinality 5, say $l_6 = 1$ and $l_3 = 1$. Notice that any other applications of switch on two nodes of T does not increase the influence (see Fig. 4, (b)).

On the other hand, two successive applications of switch lead to the desired improvement: the first one is performed by $l_1 = switch(3, 11)$ and the last one, i.e., $switch(6, 12)$, is performed by the call $l_2 = TrySwitch(T, l_1)$. The optimal solution with $I = 8$ is finally reached (see Fig. 4, (a)).

3.3 A counterexample for the general heuristic

The two previous heuristics can be generalized (with a relevant increase in complexity) to the case in which k applications of the *switch* operator are performed before the final call of Algorithm 2. In the next property, we show a general tree structure, depending on k, in which $k - 2$ applications of *switch* are needed to obtain the optimum influence.

Proposition 1. *For each $k \geq 1$, there exists a directed tree T such that $k - 2$ applications of the switch operator with 0 increment are required to move from a greedy labeling of dimension k (provided by Algorithm 1) to the optimal labeling of the same dimension.*

Proof. For a given k, let us consider the directed tree depicted in Fig. 5 having $|T| = 2(k-1) + 2(k-2) + 2 = 4k - 4$. Algorithm 1 outputs the left labeling of dimension k whose nonoptimal influence is $I = 2k - 3$.

The only way to further increase I is to apply $k-2$ times the operator *switch* and reach the labeling on the right, whose influence is still $I = 2k-3$. After that, $TrySwitch$ applies a further *switch*, indicated with the red arrow in Fig. 5, on the right, and it increases the influence by one unit, leading to the optimal one $I = 2k-2$. □

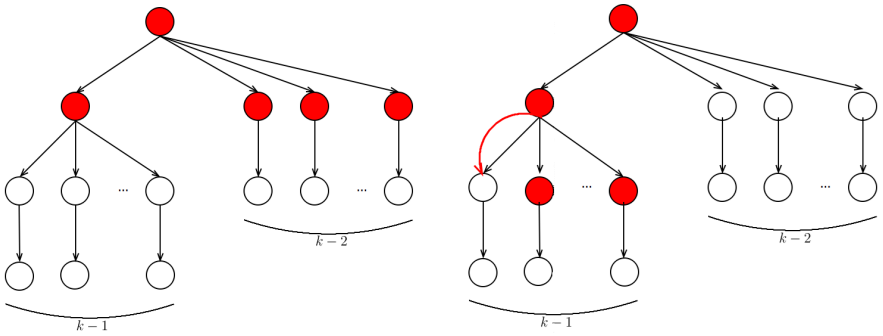


Figure 5: A greedy labeling of a directed tree that requires $k-2$ applications of *switch* before finding a way to obtain the optimal label.

4 Dynamic programming solution

In this section, we show that $CIMP(T,k)$ can be solved in polynomial time with respect to the number of nodes of the tree through a dynamic programming approach, when the maximum degree of the nodes and k are constants. Let T be a directed tree and v one of its nodes. We indicate as T_v the (directed) subtree of T whose root is v, and with $CIMP_0(T_v,k)$ [resp. $CIMP_1(T_v,k)$] a solution l of $CIMP(T_v,k)$ with the constraint that $l_v = 0$ [resp. $l_v = 1$].

We associate to the node v two $(k+1)$-length vectors, $S_{0,v}$ and $S_{1,v}$ whose positions vary from 0 to k and whose i-th entries $S_{0,v}(i)$ and $S_{1,v}(i)$ are defined as $I(T_v,l)$, with $l = CIMP_0(T_v,i)$ and $l = CIMP_1(T_v,i)$, respectively. We assume the values $S_{0,v}(i)$ with $i > |V(T_v)|$ are set to 0, and the same holds for the values $S_{1,v}(i)$. So, if v is a leaf, it turns out to be $S_{0,v} = S_{1,v} = \overline{\mathbf{0}}$, the zero constant vector.

The vector $S_{0,v}$ stores the maximum influence values obtained in the subtree T_v by the labels whose cardinalities range from 0 to k, when the node v is a 0-node, while the vector $S_{1,v}$ stores the same values when the node v is a 1-node.

By definition, if v is the root of T, then $\max\{S_{0,v}(i), S_{1,v}(i)\}$ stores the cardinality of one of the labeling of $CIMP(T,i)$, and its k-th entry provides the required solution to $CIMP(T,k)$.

Now, we provide the recurrence to compute the two vectors $S_{0,v}$ and $S_{1,v}$, for each non-leaf node v in T: to lighten the notation, we denote $d = d(v)$ and

let $v_1, \ldots, v_d$ be the children of v

$$\begin{cases} S_{0,v}(i) = \max\limits_{i_1+i_2+\cdots+i_t=i} \{\max\{S_{0,v_1}(i_1), S_{1,v_1}(i_1)\} + \cdots + \max\{S_{0,v_d}(i_d), S_{1,v_d}(i_d)\}\} \\ S_{1,v}(i) = \left(\max\limits_{i_1+i_2+\cdots+i_t=i-1} \{\max\{S_{0,v_1}(i_1), S_{1,v_1}(i_1)\} + \cdots + \max\{S_{0,v_d}(i_t), V_{1,v_d}(i_d)\}\} \right) + d_0(v) \end{cases} \tag{3}$$

where $d_0(v)$ is computed using the number of $S_{0,v_i}, 1 \leq i \leq d$ vectors involved in the sum.

In words, the maximum is taken over all indices satisfying $i_1 + \ldots + i_d = i$ [resp. $i_1 + \ldots + i_d = i - 1$] because the solution must contain exactly i [resp. $i - 1$] 1−nodes. The solution for T_v is constructed from the solutions obtained for each subtree rooted in each of the children of v, considering the optimal case, i.e. if the root must be included or not in each case.

Example 4. *Let us consider the directed tree in Fig. 6. Setting $k = 3$, we list the vectors S_0 and S_1 for the ten nodes. First of all, note that:*

1. $S_{m,v} = (0, 0, 0, 0)$ *for* $m = \{0, 1\}$,$v = \{5, 6, 7, 8, 9, 10\}$;
2. $S_{0,3} = S_{0,4} = (0, 0, 0, 0)$;
3. $S_{1,3} = (0, 3, 2, 1)$;
4. $S_{1,4} = (0, 2, 1, 0)$.

Concerning node 2, $S_{0,2}(1) = 3$ is obtained by choosing, as an example, the sequence $S_{1,3}(1) = 3, S_{0,4}(0) = 0, S_{0,5}(0) = 0$ related to the vectors of its children. Similarly, we get $S_{0,2}(2) = S_{0,2}(3) = 5$. Similarly, we obtain the vector $S_{1,2} = (0, 3, 5, 6)$.

To understand its computation, let's consider, as an example, $S_{1,2}(3) = 6$. That value is obtained by summing the sequence of values $S_{1,3}(1) = 3$, $S_{1,4}(1) = 2$, $S_{1,5}(0) = 0$ and $d_0(1) = 1$. Finally, we reach node 1 that has vectors $S_{0,1} = (0, 3, 5, 6)$ and $S_{1,1} = (0, 1, 4, 6)$, so providing the two solutions in Fig. 6 with maximal influence six.

The following theorem provides the correctness of the recurrence:

Lemma 1. *For each node v of a directed tree T, the values $S_{0,v}(i)$ and $S_{1,v}(i)$, store the influences of the labels that are solutions of $CIMP_0(T_v, i)$ and $CIMP_1(T_v, i)$, respectively.*

Proof. Let us proceed by induction on the height of the tree T_v.

Base: if v is a leaf, then the maximal influence of any of the two possible labels $l_v = 1$ or $l_v = 0$ is 0, and both the vectors $V_{0,n}$ and $V_{1,n}$ are the 0-constant vector;

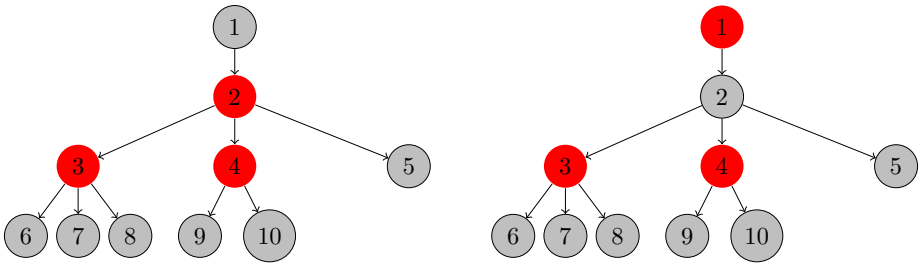


Figure 6: The two solutions of $CIMP(T, 3)$. The labels of the nodes can be obtained from the computation of maximum values of the vectors $V_{0,1}$ and $V_{1,1}$, being node 1 the root of the tree.

Inductive step: let $v_1, \dots v_d$ be the children of v. By inductive hypothesis, the vectors S_{0,v_j} and S_{1,v_j} store the influences of the solutions of $CIMP_0(T_{v_j}, i)$ and $CIMP_1(T_{v_j}, i)$, with $0 \leq i \leq k$ and $1 \leq j \leq d$, respectively. Let us focus on the vector $S_{0,v}$ whose i-th entry is computed using the maximal (according to the influence value) sequence of indexes $i_1 + i_2 + \dots + i_d = i$. Assume by contradiction that

$$I(T_v, l) > S_{0,v}(i), \tag{4}$$

with $l = CIMP_0(T_v, i)$.

In that case, the label l produces $l^{(1)}, \dots, l^{(d)}$ labels of each subtree $T_{v_1}, \dots, T_{v_d}$ such that $|l^{(1)}| + \dots + |l^{(d)}| = i$. So, if (4) holds, it means that one of the labels V_{0,v_j} and V_{1,v_j} were not optimal, contradicting their definitions. A similar reasoning holds for $S_{1,v}$, reaching the thesis. □

Theorem 1. *Given a directed tree $T = (V, E)$ and an integer $0 \leq k \leq |T|$, the problem $CIMP(T, k)$ can be solved in $O(|V|(k + \bar{d})^{\min\{k,\bar{d}\}})$ time, with $\bar{d} = \max_{v \in V} d(v)$.*

Proof. Lemma 1 assures that the value $\max\{S_{0,v}(k), S_{1,v}(k)\}$ is the influence of one of the labeling l that is a solution of $CIMP(T, k)$, when n is the root of T. Furthermore, the sequences of indices used to compute such a value recursively provide l.

The complexity of the computation of l follows from the definition of the vectors $S_{0,v}$ and $S_{1,v}$. In particular, the heaviest computational step concerns the search of the indexes' sequences $i_1, \dots, i_d$ that provide the maximum influences for each internal node v of T, and for each cardinality of the labels (recall that $d = d(v)$). The computations of the entries $S_{0,v}(i)$ and $S_{1,v}(i)$ require to scan all the $\binom{i+d-1}{i}$ indexes' sequences of the vectors S_{0,v_i} and S_{1,v_i}, with v_i ranging among the children of v, to find those indexes summing to i and producing the maximum influence. To hit the aim, the final complexity reaches $O(|V|(k + \bar{d})^{\min\{k,\bar{d}\}})$. □

The following example shows how the vectors S_0 and S_1 lead to the solutions of $CIMP(T,k)$ related to the tree T in Fig. 5 and provided as a counterexample to the greedy heuristic with a bounded number of switch operations to the problem.

Example 5. *Let us consider the problem $CIMP(T,5)$, where T is the directed tree in Fig. 5. Let us number the nodes of T starting from node 1, the root, and proceeding to its children from left to right, using the labels $2,\dots,5$. We compute the vectors $S_{0,2} = (0,1,2,3,4,3)$, $S_{1,2} = (0,4,4,4,4,4)$, $S_{0,i} = \overline{\mathbf{0}}$ and $S_{1,i} = (0,1,0,0,0,0)$, with $i \in \{3,4,5\}$. The computation of the vectors related to node 1 rely on their values obtaining $S_{0,1} = (0,4,5,6,7,7)$, and $S_{1,1} = (0,4,7,7,7,8)$. The maximal value $S_{1,1}(5) = 8$ is the (unique) solution of $CIMP(T,5)$ and it is obtained by the sequence $S_{0,2}(4) = 4$, $S_{0,i}(0) = 0$, with $i \in \{3,4,5\}$ and $d_0(v) = 4$.*

5 Conclusions

In this work, we introduced and analysed the Constrained Influence Maximization Problem ($CIMP$) on directed tree structures, extending the classical formulation of the influence maximization framework.

After presenting possible heuristics, we introduced a dynamic programming algorithm that computes the exact solution to CIMP in polynomial time, provided that k and the maximum degree of the tree are fixed. Our approach systematically builds optimal labelings for subtrees, storing influence values under the constraint of fixed cardinality and root label.

However, there is still room for improvement of the proposed algorithms. In fact, our approach led to a combinatorial explosion in k and d. Therefore, future works will focus on different strategies to reduce its complexity and, if possible, to extend these techniques to broader classes of graphs, incorporate probabilistic influence models, or consider additional constraints and real-world applications.

References

[1] Andrei Mouravski. *Influence maximization on families of graphs*. Rochester Institute of Technology, 2011.

[2] Rahul Kumar Gautam, Anjeneya Swami Kare, and S Durga Bhavani. Heuristics for influence maximization with tiered influence and activation thresholds. In *International Conference on Advanced Network Technologies and Intelligent Computing*, pages 134–148. Springer, 2023.

[3] Maksim Kitsak, Lazaros K. Gallos, Shlomo Havlin, Fredrik Liljeros, Lev Muchnik, H. Eugene Stanley, and Hernán A. Makse. Identification of influential spreaders in complex networks. *Nature Physics*, 6(11):888–893, August 2010.

[4] Niccolò Di Marco, Sara Brunetti, Matteo Cinelli, and Walter Quattrociocchi. Post-hoc evaluation of nodes influence in information cascades: The case of coordinated accounts. *ACM Trans. Web*, October 2024. Just Accepted.

[5] Chengcheng Shao, Giovanni Luca Ciampaglia, Onur Varol, Kai-Cheng Yang, Alessandro Flammini, and Filippo Menczer. The spread of low-credibility content by social bots. *Nature communications*, 9(1):1–9, 2018.

[6] 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.

[7] Mehrdad Agha Mohammad Ali Kermani, Reza Ghesmati, and Mir Saman Pishvaee. A robust optimization model for influence maximization in social networks with heterogeneous nodes. *Computational Social Networks*, 8(1):17, 2021.

[8] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD03. ACM, August 2003.

[9] Elchanan Mossel and Sebastien Roch. On the submodularity of influence in social networks. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, STOC07, page 128–134. ACM, June 2007.