

Beyond Local Influence: Improved Bounds, Exact Models and Efficient Algorithms for Component-Based Diffusion in Networks

Matteo Bergamaschi · Marta Pozzi · Francesco
Rinaldi · Laura Sanità

Received: date / Accepted: date

Abstract Diffusion in networks is often modeled as a local neighbor-based process: a node becomes active as a result of influence exerted by its active neighbors. However, in many technological and communication settings, adoption is driven not only by local influence, but by access to sufficiently large active connected regions. This paper studies such component-based diffusion dynamics, focusing on the Technology Diffusion (TD) problem and on a related Influence Maximization (IM) problem described by the same propagation process.

We first analyze how some connectivity restrictions affect component-based diffusion and show that their impact can be bounded more tightly than earlier results in the literature suggested. We then propose exact integer programming formulations that are not based on the connected-activation-sequence paradigm, showing the importance of considering disconnected active sets. On the algorithmic side, we develop a zeroth-order approach, namely a component-aware direct search that does not require the diffusion trajectory to be a connected activation sequence. Computational experiments on a standard benchmark show that the proposed method improves solution quality over state-of-the-art approaches, and it scales to significantly larger instances than exact/approximation methods.

1 Introduction

Diffusion processes on networks describe how individual adoption decisions can generate large-scale propagation. They arise in applications such as information dissemination, viral marketing, epidemic spreading, and the adoption of communication technologies. Most classical diffusion models are local: whether a node becomes active depends on the states of its immediate neighbors. Prominent examples in this context are influence maximization models formalized by Kempe, Kleinberg, and Tardos [20], which rapidly became one of the central problems in network science and diffusion theory. Given a graph $G = (V, E)$ and a diffusion model, the objective is to identify a subset of initially active nodes capable of maximizing the spread of influence throughout the network. The literature on influence maximization mainly focuses on two classical propagation schemes: the Linear Threshold Model and the Independent Cascade Model [20]. In both cases, the diffusion mechanism is local, in the sense that the activation of a node depends exclusively on the state of its immediate neighbors. In the Linear Threshold Model, a node activates when the cumulative influence received from active adjacent nodes exceeds a prescribed threshold. In the Independent Cascade Model, instead, active nodes attempt to activate their inactive neighbors according to probabilistic propagation rules. A substantial portion of the literature has focused on exploiting structural properties of these local diffusion models in order to derive efficient approximation algorithms.

Local influence, however, does not capture every form of network-driven adoption. In many technological and communication settings, the benefit of adopting a technology depends on the size of the active network that becomes accessible after adoption. A user may therefore be influenced not only by the

M. Bergamaschi, F. Rinaldi
Department of Mathematics “Tullio Levi-Civita”, University of Padova,
Via Trieste 63, 35121 Padova, Italy

M. Pozzi, L. Sanità
Department of Computing Sciences, Bocconi University,
Via Roentgen 1, 20136 Milano, Italy

number of active neighbors, but also by the sizes of the active connected regions to which those neighbors belong. This observation motivated Goldberg and Liu [17] to define diffusion models driven by connected components rather than exclusively by pairwise interactions. In particular, they introduce the Technology Diffusion (TD) problem, in which this type of propagation is formalized. Each node u is assigned a threshold $\theta(u)$. Starting from a set of initially active nodes, called seeds, an inactive node becomes active when the total size of the active connected components adjacent to it is at least $\theta(u) - 1$. Once activated, the node may connect several previously disjoint active components, thereby enabling further activations. The objective of TD is to find a minimum-cardinality seed set that eventually activates the entire network.

Goldberg and Liu [17] propose an approximation algorithm for the TD problem, as well as an IP formulation that was used to benchmark some heuristics. The approximation factor has been subsequently improved by [21], and additionally reduced by [13] for some special classes of graphs. However, all of these results make use of a central structural assumption, namely, the activation sequence is required to be connected, meaning that the set of active nodes must induce a connected subgraph at every stage of the process. Nevertheless, connectedness is not an intrinsic requirement of the original diffusion dynamics. In an unrestricted process, several active components may grow independently and merge only at a later stage. Requiring the active set to remain connected may therefore exclude better seed sets.

Furthermore, the component-based propagation scheme at the heart of the TD problem also gives rise to a natural Influence Maximization (IM) variant: Given a fixed budget on the number of initial seeds, maximize the final number of activated nodes. However, influence maximization under this component-based propagation scheme has received little attention so far.

1.1 Our contribution

Our first contribution is a refined investigation of the structural role of connected activation sequences for TD. Goldberg and Liu proved that restricting the search to connected activation sequences yields at most a multiplicative factor-2 loss with respect to the optimal solution. However, this multiplicative guarantee relies essentially on the assumption that every node has threshold at least two. This assumption rules out nodes that adopt spontaneously, without requiring any previously active node. Such nodes have a natural interpretation in technology-adoption settings: they represent users who are willing to adopt a technology independently of the behavior of others. We here show that, when threshold-one nodes are allowed, no multiplicative comparison between connected and unrestricted activation sequences is possible. Indeed, an unrestricted process may activate the entire network without using any seed, while every connected activation sequence may still require additional seeds. On the positive side, we derive an additive bound depending on the number of distinct threshold values appearing in the instance. Our additive bound works without imposing any assumption on the thresholds, and it is tight.

Besides clarifying the theoretical cost of connected activation sequences, these results also motivate the study of the unrestricted problem directly. We therefore introduce the first exact Integer Programming (IP) formulation for the TD Problem, without imposing connectivity restrictions on the activation process. We also show that our modeling framework can be adapted to the corresponding IM variant. We compare our formulation with the IP formulation of Goldberg and Liu, which instead relies on the connected activation sequence assumption. Our computational results confirm that allowing disconnected activation sequences leads to smaller seed sets in practice.

This latter observation motivates us to develop heuristic approaches which do not insist on searching for connected activation sequences. We design a Component-Aware Direct Search method, denoted by COaDS. For a fixed seed budget, COaDS treats the diffusion process as an evaluation oracle and searches directly over candidate seed sets. Its neighborhoods and move-selection rules exploit both the topology of the graph and the component-based activation mechanism. We first formulate COaDS for the component-based IM problem and then embed it in a heuristic binary-search procedure over the seed-set cardinality to address TD. Experiments on preferential-attachment networks following the benchmark framework of Goldberg and Liu [17] show that COaDS consistently produces smaller seed sets than the tested heuristic, approximation, and integer-programming baselines. It also remains effective on networks with thousands of nodes, where the exact formulations become impractical and the approximation algorithm may fail to return a feasible solution within the available computational budget.

1.2 Related work

Technology diffusion. As mentioned, the TD problem was introduced by Goldberg and Liu [17] to model technology adoption over communication networks. Unlike neighbor-based diffusion models, TD makes the activation of a node depend on the size of the active connected region that becomes accessible through that node. Goldberg and Liu established hardness results relating TD to Set Cover and proposed an $O(r\ell \log n)$ -approximation algorithm, where r is the diameter of the graph and ℓ is the number of distinct threshold values, all of which are greater than 1. Könemann, Sadeghian, and Sanità [21] subsequently developed simpler combinatorial algorithms with approximation ratio $O(\min\{r, \ell\} \log n)$, using reductions to Submodular Set Cover and the quota node-weighted Steiner Tree problem. Esfandiari et al. [13] gave an $O(\log \ell)$ -approximation in the restricted case where the underlying graph is a spider (that is, a tree with at most one node of degree larger than 2). Sharma [32] shows that TD and the corresponding IM variant are polynomial-time solvable on graphs that are spiders with a constant number of legs.

Local influence maximization. One of the most important observations of [20] is that the expected influence function is monotone and submodular under both the Linear Threshold and Independent Cascade dynamics. This property immediately yields a $(1 - 1/e)$ -approximation through a greedy strategy. Since such a greedy algorithm requires many repeated evaluations of the expected spread, several works have proposed lazy evaluation and simulation-based accelerations, such as the CELF optimization of Leskovec et al. [23], as well as scalable heuristics for the Independent Cascade model [7, 8].

Subsequent works concentrated on improving the scalability of these approaches, mainly due to the computational complexity of repeatedly evaluating the diffusion process. In particular, several algorithmic frameworks based on sampling techniques, reverse reachable sets, and sketching methods have been proposed in order to obtain near-optimal solutions on large-scale social networks. Important examples include the reverse-reachable-set framework of Borgs et al. [6], the TIM/TIM+ algorithms [34], the IMM algorithm [33], and sketch-based approaches for influence estimation and maximization [9]. A comprehensive survey can be found in [24]. Mathematical programming approaches for (interesting variants of) influence maximization problems can be found, e.g., in [14, 15, 18, 19, 29, 35, 37].

Despite the vast literature on IM, the overwhelming majority of existing models rely on neighbor-based propagation dynamics. In other words, influence spreads through direct local interactions between adjacent nodes. The component-based diffusion mechanism introduced by Goldberg and Liu for the TD Problem [17] suggests a substantially different perspective. Motivated by this observation, in this work we study the variant of the IM problem based on the diffusion dynamics of the TD framework.

Direct search methods for diffusion problems. Direct search methods represent a classical and widely used family of derivative-free optimization algorithms that iteratively evaluate the objective function at a finite set of trial points around the current incumbent solution and accept a new point whenever a sufficient improvement is found. Broader treatments appear in [3, 10], while a recent survey can be found in [12]. For discrete and mixed-integer problems, relevant foundational works include mixed-variable pattern search [2], filter and mesh-adaptive methods [1], with extensions to granular variables [4]. Derivative-free line-search frameworks have addressed bound-constrained [26], constrained mixed-integer [27], and pure-integer black-box problems through primitive directions [25]; related ideas were extended to mixed-integer nonsmooth constrained optimization in [16]. Complementary approaches include model-based trust-region methods [30], radial-basis-function-based strategies [11], surrogate-based schemes [28], and black-box integer global optimization strategies [22]; see [31] for a systematic review and comparison.

These methods are particularly suitable for diffusion problems on networks, where the search space is inherently combinatorial and the objective value is often obtained through a simulation procedure. This is the case, for instance, in several IM formulations, where candidate seed sets are discrete subsets of nodes whose quality is evaluated by simulating the diffusion process and measuring the final spread. The combination of a discrete feasible region and expensive black-box objective evaluations hence makes direct-search methods appealing, but also requires problem-specific neighborhoods that exploit the network structure to guarantee computational effectiveness. A first application of direct search to diffusion problems was proposed in [36], which formulated IM as a mixed-integer nonlinear problem with a black-box objective and introduced a Customized Direct Search method for exploring the space of feasible seed sets. In [5], the authors developed a Network-aware Direct Search approach that exploits graph structure in the neighborhood construction, reducing function evaluations while maintaining solution quality.

These works show the potential of direct-search methods for diffusion problems and motivate their application to the framework studied here. In this setting, the diffusion process is indeed evaluated through simulation, while the evolution of active connected components provides structural information that can be exploited through neighborhoods tailored to both the network topology and the TD dynamics.

1.3 Formal definition of the problem

We close the section with a formal description of the TD problem and dynamics. Let $G = (V, E)$ be an undirected graph, where V is the set of nodes, with $|V| = n$, and E is the set of edges, with $|E| = m$. Each node $u \in V$ can be in one of two states: inactive or active. We are given a threshold function $\theta : V \rightarrow \{1, \dots, |V|\}$, which associates with each node u a value $\theta(u)$, representing the minimum number of active nodes, including u itself, that have to be reachable from u through a path of active nodes to activate.

The process starts from activating a certain subset of nodes, called seeds, whose set is denoted by $S \subseteq V$. Once activated, a node remains active for all subsequent steps. Formally: given a seed set $S \subseteq V$, let $A_t(S) \subseteq V$ denote the set of active nodes at time t . The diffusion process is initialized as $A_0(S) = S$. For every $t \geq 0$, let $\mathcal{C}(A_t(S))$ be the set of connected components of the subgraph induced by $A_t(S)$. Then a node $u \notin A_t(S)$ becomes active at time $t + 1$ if the total size of the active connected components adjacent to u is at least $\theta(u) - 1$, namely if

$$\sum_{\substack{C \in \mathcal{C}(A_t(S)) : \\ N(u) \cap C \neq \emptyset}} |C| \geq \theta(u) - 1,$$

where $N(u)$ denotes the set of neighbors of u . Hence,

$$A_{t+1}(S) = A_t(S) \cup \left\{ u \in V \setminus A_t(S) : \sum_{\substack{C \in \mathcal{C}(A_t(S)) : \\ N(u) \cap C \neq \emptyset}} |C| \geq \theta(u) - 1 \right\}.$$

Since the process is progressive, there exists a finite time at which no further activation is possible. We denote by $\text{Spread}(G, S, \theta)$ the final cardinality of the active-node set obtained by applying the component-based diffusion process on the graph G , with thresholds θ , starting from the seed set S .

The TD Problem consists of determining a seed set $S \subseteq V$ of minimum cardinality such that, starting from S , the diffusion process eventually activates all nodes in V . Therefore, it can be written in compact form as

$$\begin{aligned} \min_{S \subseteq V} \quad & |S| \\ \text{s.t.} \quad & \text{Spread}(G, S, \theta) = |V|. \end{aligned}$$

2 The role of connected activation sequences

2.1 A refined bound

We start with some definitions:

Definition 1 (Activation Sequence [17]) Let $S \subseteq V$ be a seed set. An *activation sequence* is a permutation $T : V \rightarrow \{1, \dots, n\}$, where $n = |V|$, describing the order in which nodes become active during the diffusion process. The value $T(u)$ denotes the activation time of node u .

Seed nodes may activate at an arbitrary time step t . A non-seed node $u \notin S$ can activate at time step $T(u)$ only if, in the subgraph of G induced by

$$\{u\} \cup \{v \in V : T(v) < T(u)\},$$

the connected component containing u has cardinality at least $\theta(u)$.

Definition 2 (Connected Activation Sequence [17]) An activation sequence T is said to be *connected* if, for every time step $t \in \{1, \dots, n\}$, the set

$$\{u \in V : T(u) \leq t\}$$

induces a connected subgraph of G .

The work of [17] provides a first approximation guarantee for studying connected activation sequences.

Theorem 1 (Goldberg–Liu [17]) *Let $G = (V, E)$ be a connected graph with node thresholds ≥ 2 . The size of the smallest seed set that induces a connected activation sequence is at most twice the size of an optimal seed set.*

We remark that the bound provided by the former result does not hold if we allow nodes to have unit thresholds in the graph. This type of node does not need to be connected to any other component to activate as a non-seed node.

Example 1: No multiplicative bound is possible. An easy example is given by a small chain composed of three nodes, where the central node u has threshold $\theta(u) = 3$, while the two nodes v, v' on the side have threshold $\theta(v) = \theta(v') = 1$. The TD dynamics allows the nodes on the side to activate for free, and the central one to activate afterwards, leading to an optimal seed set $OPT = \emptyset$ of size 0. However, if we want to turn the activation sequence into a connected one, we would need to seed the central node, getting an optimum for the connected process of cardinality 1. Hence, no multiplicative bound can hold.

We now derive an *additive* bound that depends on the number of distinct threshold values. We remark that our bound holds for any instance, independently of whether or not some nodes have threshold 1.

Theorem 2 *Let $G = (V, E)$ be a connected graph with thresholds*

$$\theta_1 < \theta_2 < \dots < \theta_\ell.$$

Let T^* be an optimal activation sequence for G , and let OPT denote the corresponding optimal seed set. Then there exists a connected activation sequence whose associated seed set $\widehat{OPT}$ satisfies

$$|\widehat{OPT}| \leq |OPT| + \ell.$$

Proof We prove a slightly stronger statement. Let $U \subseteq V$ be an active connected component at some point during the activation sequence T^* , and define $q(U)$ as the number of distinct threshold values attained by the non-seed vertices of U . That is,

$$q(U) = |\{\theta(v) : v \in U \setminus OPT\}|.$$

We claim that the vertices of U admit a connected activation sequence whose seed set is $(OPT \cap U) \cup X_U$, for some set of nodes $X_U \subseteq U : |X_U| \leq q(U)$. We prove this claim by induction on $q(U)$.

Base case. If $q(U) = 0$, then every vertex of U belongs to OPT . Since $G[U]$ is connected, its vertices can be ordered according to a traversal of a spanning tree of $G[U]$. Every vertex after the first is adjacent to a previously activated vertex, and all vertices are seeds. Hence, the resulting activation sequence is connected and no additional seed is required.

Induction step. Suppose now that $q(U) > 0$, and assume that the claim holds for every active connected component containing fewer than $q(U)$ distinct threshold values among its non-seed vertices. Let $\mu = \max \{\theta(v) : v \in U \setminus OPT\}$, and among the vertices $v \in U \setminus OPT$ satisfying $\theta(v) = \mu$, let u be the first one activated in T^* .

Immediately before the activation of u , let $C_1, C_2, \dots, C_k$ be the active connected components adjacent to u , indexed so that $|C_1| \geq |C_2| \geq \dots \geq |C_k|$. Since u is not a seed, the feasibility of its activation implies

$$1 + \sum_{j=1}^k |C_j| \geq \mu. \tag{1}$$

First, assume that $k \geq 1$. By the choice of u , none of the components $C_1, \dots, C_k$ contains a non-seed vertex with threshold μ . Indeed, every vertex in these components was activated before u , whereas u is the first non-seed vertex of threshold μ . Consequently, $q(C_1) \leq q(U) - 1$. The induction hypothesis can therefore be applied to C_1 . Hence, the vertices of C_1 admit a connected activation sequence using the original seeds in $OPT \cap C_1$ and at most $q(U) - 1$ additional seeds.

After constructing C_1 , we place u next and add it to the seed set. Since u is adjacent to C_1 , this preserves the connectivity of the activation sequence. Note that this step introduces one additional seed.

We next append the components $C_2, \dots, C_k$, one at a time. For each $j \geq 2$, choose a vertex of C_j adjacent to u , and order the vertices of C_j according to a traversal of a spanning tree of the component C_j rooted at that vertex. This preserves connectivity.

To verify feasibility, consider a non-seed vertex $v \in C_j$. At the time when v was activated in T^* , the active component witnessing its activation was contained in C_j . This follows from the fact that active connected components may merge during the diffusion process but never split. Therefore, $\theta(v) \leq |C_j|$. Moreover, since C_1 was chosen as a largest component, $|C_1| \geq |C_j|$. Before the vertices of C_j are appended, the connected active set already contains $C_1 \cup \{u\}$ and thus has cardinality at least $\theta(v)$. It follows that every non-seed vertex of C_j satisfies its activation threshold when it is appended. Hence, all the components $C_2, \dots, C_k$ can be appended without introducing any additional seeds.

After u and all the components $C_1, \dots, C_k$ have been included, the connected active set has cardinality $1 + \sum_{j=1}^k |C_j| \geq \mu$ by (1). Every remaining non-seed vertex $v \in U$ satisfies $\theta(v) \leq \mu$ by the definition of μ . Since $G[U]$ is connected, the remaining vertices can be appended in an order in which every new vertex is adjacent to the current connected active set. Their thresholds are therefore satisfied, and no further seeds are required.

The total number of additional seeds used in this construction is at most $q(U)$.

Now consider the case $k = 0$. In this case, u has no active neighbor immediately before its activation. Since u is not a seed, the feasibility of its activation implies $\mu = 1$. We place u first in the new activation sequence. Since $\theta(u) = 1$, the component consisting only of u is large enough to activate u without declaring it a seed. Since $G[U]$ is connected, the remaining vertices of U can then be appended according to a connected tree rooted at u . Every remaining non-seed vertex has threshold at most $\mu = 1$, and every original seed can be activated unconditionally. Hence, no additional seed is required.

This completes the induction and proves the claim.

We now apply the claim to $U = V$. At the end of T^* , all vertices are active and, since G is connected, V forms one active connected component. Furthermore, $q(V) \leq \ell$. Therefore, there exists a connected activation sequence with associated seed set $\widehat{\text{OPT}}$ such that

$$|\widehat{\text{OPT}}| \leq |\text{OPT}| + q(V) \leq |\text{OPT}| + \ell.$$

The next example shows that our additive factor is essentially tight, and hence the dependency on the parameter ℓ cannot be avoided.

Example 2: Additive $\Omega(\ell)$ factor is needed. We will construct a rooted tree starting from the leaves, and recursively adding layers on top. We fix $\ell \geq 2$. Let G_0 consist of a single node with threshold 1. This node is considered a root of a trivial tree. For $h = 1, \dots, \ell - 1$, construct G_h . We create ℓ disjoint copies of G_{h-1} and add a new root r_h , adjacent to the root of each copy. Preserve all existing thresholds and set

$$\theta(r_h) = \ell |V(G_{h-1})| + 1 = |V(G_h)|.$$

If we allow disconnected sequences, all vertices activate without seeds by proceeding from the leaves towards the root, so $|\text{OPT}| = 0$. If instead we require the activation sequence to remain connected, each level forces an additional seed.

Combining our bound with the already known bound from [17], we get the following refined bound for instances with minimum threshold ≥ 2 .

Theorem 3 *Let $G = (V, E)$ be a connected graph with thresholds*

$$2 \leq \theta_1 < \theta_2 < \dots < \theta_\ell.$$

Let OPT denote the minimum size of a feasible seed set, and let $\widehat{\text{OPT}}$ denote the minimum size of a seed set inducing a connected activation sequence. Then

$$|\widehat{\text{OPT}}| \leq \min\{2|\text{OPT}|, |\text{OPT}| + \ell\}.$$

2.2 Formulations

Goldberg and Liu [17] develop an Integer Programming formulation based on the connectivity assumption. The key observation behind this restriction is that, in a connected activation sequence, the activation time of a node coincides with the size of the active connected component at the moment of activation. Indeed, if the active nodes always induce a connected subgraph, then at time step t exactly t nodes are active, and therefore a node u can activate at time t if and only if $t \geq \theta(u)$ and u is adjacent to the active component.

This property allows the original diffusion dynamics to be encoded using local connectivity constraints. In particular, the diffusion process can be represented through an ordering of the vertices satisfying suitable adjacency conditions.

Let $x_{i,t}$ be a binary variable such that

$$x_{i,t} = \begin{cases} 1 & \text{if node } v_i \text{ activates at time step } t, \\ 0 & \text{otherwise.} \end{cases}$$

The Integer Programming formulation proposed in [17] is the following:

$$\begin{aligned} \min \quad & \sum_{i \leq n} \sum_{t < \theta(v_i)} x_{i,t} \\ \text{s.t.} \quad & x_{i,t} \in \{0, 1\}, & \forall i, t, \\ & \sum_{t \leq n} x_{i,t} = 1, & \forall i, \\ & \sum_{i \leq n} x_{i,t} = 1, & \forall t, \\ & \sum_{\{v_i, v_{i'}\} \in E} \sum_{t' < t} x_{i',t'} \geq x_{i,t}, & \forall i, t > 1. \end{aligned}$$

The first set of constraints imposes binary feasibility. The second and third sets of constraints enforce that the variables define a permutation of the vertices, namely that each node activates exactly once and exactly one node activates at each time step. Note that these constraints are not necessary for the true process, but they are fundamental to maintain the correspondence between the size of the connected component and the activation time. Finally, the connectivity constraints guarantee that every non-initial activation is adjacent to at least one previously active node, ensuring that the activation sequence remains connected throughout the diffusion process.

We provide a formulation that, in contrast to the one above, does not impose any connectivity assumption on the set of active nodes throughout the diffusion process. Here, we denote by N the cardinality of V . We allow all the seeds to activate together at the beginning of the process, and start their own active connected components. These components can later merge while new nodes become active. Our decision variables are the following:

$$x_{i,t} \in \{0, 1\} \quad (i \in V, t \geq 0),$$

where $x_{i,t} = 1$ if node i activates at time t . The equality $x_{i,0} = 1$ indicates that node i is initialized at time zero. The nodes that activate at time 0 with threshold greater than one are counted as seeds, whereas nodes with threshold one may be initialized without cost.

$$y_{i,r,t} \in \{0, 1\}, \quad n_{i,r,t} \in \mathbb{Z}_{\geq 0} \quad (i \in V, r \in \{1, \dots, N\}, t \geq 0),$$

are the additional variables needed to model the diffusion process, taking into account the fact that different connected components can grow independently over time. In particular, $y_{i,r,t} = 1$ if node i belongs to an active connected component labeled by r at time t , while $n_{i,r,t}$ represents the size of the active component labeled by r adjacent to the node i at time t .

The formulation is the following:

$$\min \sum_{\substack{i \in V \\ \theta_i > 1}} x_{i,0} \quad (2)$$

$$\text{s.t.} \quad \sum_{t \geq 0} x_{i,t} = 1 \quad \forall i \in V \quad (3)$$

$$\sum_{i \in V} x_{i,t} \leq 1 \quad \forall t \geq 1 \quad (4)$$

$$\theta_i x_{i,t} \leq \sum_{r=1}^N n_{i,r,t} \quad \forall i \in V, \forall t \geq 1 \quad (5)$$

$$n_{i,r,t} \leq \sum_{j \in V} y_{j,r,t} \quad \forall i \in V, \forall r = 1, \dots, N, \forall t \geq 1 \quad (6)$$

$$n_{i,r,t} \leq N \sum_{j \in N(i)} y_{j,r,t} \quad \forall i \in V, \forall r = 1, \dots, N, \forall t \geq 1 \quad (7)$$

$$\sum_{r=1}^N y_{i,r,t} \leq 1 \quad \forall i \in V, \forall t \geq 0 \quad (8)$$

$$y_{i,r,t} \leq \sum_{t'=0}^t x_{i,t'} \quad \forall i \in V, \forall r = 1, \dots, N, \forall t \geq 0 \quad (9)$$

$$\sum_{i \in V} y_{i,r,0} \leq 1 \quad \forall r = 1, \dots, N \quad (10)$$

$$y_{i,r,t} \leq y_{i,r,t-1} + \sum_{j \in N(i)} y_{j,r,t-1} \quad \forall i \in V, \forall r = 1, \dots, N, \forall t \geq 1 \quad (11)$$

$$x_{i,t} \in \{0, 1\} \quad \forall i \in V, \forall t \geq 0 \quad (12)$$

$$y_{i,r,t} \in \{0, 1\} \quad \forall i \in V, \forall r = 1, \dots, N, \forall t \geq 0 \quad (13)$$

$$n_{i,r,t} \in \mathbb{Z}_{\geq 0} \quad \forall i \in V, \forall r = 1, \dots, N, \forall t \geq 1 \quad (14)$$

The minimization problem is subject to three classes of constraints: activation constraints, namely constraints (3)–(4), component-labeling constraints, which consist of (8) to (11), diffusion constraints, (5), (6), and the counting constraint (7).

Proposition 1 *The integer programming formulation described above correctly models the Technology Diffusion Problem allowing disconnected activation sequences.*

Proof We first show that any feasible solution to the TD problem can be mapped to a feasible solution of the integer programming formulation. Let $S \subseteq V$ be a feasible seed set for the TD process, and let T be an activation sequence induced by S , where $T(i)$ denotes the activation time of node $i \in V$. Since S is feasible, every node in V is eventually activated.

In order to define the mapping, we first need to align the time horizon of the true activation sequence with the one of the formulation. Indeed, our formulation is constructed over a time horizon of N^2 , to allow the merging of the connected components through successive swaps of the labels of the variables y .

We map each activation time $T(i)$ of the TD sequence to time $NT(i)$ of the extended horizon. This leaves $N - 1$ label-update steps between consecutive activations.

Therefore, the variables are defined as follows:

- We initialize all seeds and all nodes with threshold one at time zero of the formulation. This initialization preserves feasibility: threshold-one nodes activate without any previously active neighbor, and the original TD process starting from S can activate all of them by its first step. For every $i \in V$ and $t \in \{0, \dots, N^2\}$, we set

$$x_{i,t} = \begin{cases} 1, & \text{if } t = 0 \text{ and } (i \in S \text{ or } \theta_i = 1), \\ 1, & \text{if } i \notin S, \theta_i > 1, \text{ and } t = NT(i), \\ 0, & \text{otherwise.} \end{cases}$$

- The variables $y_{i,r,t}$ record the labels assigned to active nodes. We construct the labeling so that, immediately before each positive-time activation, every active connected component carries a single label and distinct components carry distinct labels. When an activation merges several components, their labels are unified during the subsequent label-update steps.

At time $t = 0$, each seed and each node with threshold one is initialized with its own label. Specifically, we set

$$y_{i,r,0} = \begin{cases} 1, & \text{if } r = i \text{ and either } i \in S \text{ or } \theta_i = 1, \\ 0, & \text{otherwise.} \end{cases}$$

At every time, all label variables associated with an inactive node are set to zero.

We first unify the labels within each connected component of the subgraph induced by the nodes initialized at time zero. We process these components one at a time. For each component, choose one of its vertices arbitrarily and let r denote its initial label. As long as some vertex of that component does not carry label r , select, at each label-update step, exactly one such vertex i having a neighbor $j \in N(i)$ with $y_{j,r,t-1} = 1$. Set $y_{i,r,t} = 1$ and $y_{i,r',t} = 0$ for every $r' \neq r$. All other vertices retain their previous label values. Here, $N(i)$ denotes the open neighborhood of i .

Such a vertex always exists until the component is entirely labeled by r , because the component is connected and the selected representative retains label r throughout the procedure. Once the component is entirely labeled by r , proceed to the next component. Consequently, the total number of label changes across all initial components is at most $N - 1$.

Now let i be a node activated at time $NT(i) > 0$, and let $C_1, \dots, C_\ell$ be the active connected components adjacent to i before its activation. Each of these components carries a single, distinct label. We choose one of these components arbitrarily and we denote its label by r . Unless i is the last remaining inactive node, we will propagate this label across the other components to unify all of them. At the activation time of i , we set

$$y_{i,r,NT(i)} = 1, \quad y_{i,r',NT(i)} = 0 \quad \text{for every } r' \neq r.$$

All the other active nodes maintain their labels at this time step. We use the label-update steps, starting at $NT(i) + 1$, to propagate label r throughout the newly formed component. As long as some vertex of this component does not carry label r , select, at each subsequent time step t , exactly one vertex

$$j \in \{i\} \cup \bigcup_{k=1}^{\ell} C_k$$

that does not carry label r at time $t - 1$ and has a neighbor

$$j' \in N(j) \cap \left(\{i\} \cup \bigcup_{k=1}^{\ell} C_k \right)$$

with $y_{j',r,t-1} = 1$. Set $y_{j,r,t} = 1$ and $y_{j,r',t} = 0$ for every $r' \neq r$. All other vertices retain their previous label values.

Such a vertex exists whenever the newly formed component is not yet entirely labeled by r , because this component is connected and the vertices already carrying r retain that label throughout the procedure. Each vertex that does not initially carry r changes label exactly once. Consequently, again, at most $N - 1$ label-update steps are required. Since consecutive positive-time activations are separated by at least N time steps, relabeling is completed before the next activation. Once relabeling is complete, variables can retain the same values until the next activation.

After the last activation, retain all label values until the end of the time horizon.

- For every $i \in V$, $r \in \{1, \dots, N\}$, and $t \geq 1$, we set

$$n_{i,r,t} = \begin{cases} \sum_{j \in V} y_{j,r,t}, & \text{if } \sum_{j \in N(i)} y_{j,r,t} \geq 1, \\ 0, & \text{otherwise.} \end{cases}$$

Thus, $n_{i,r,t}$ is the number of nodes carrying label r whenever the corresponding component is adjacent to i , and is zero otherwise.

We now verify that the variables defined above satisfy all the constraints of the formulation. Since every node is activated exactly once, Constraint (3) is satisfied. Moreover, the activations of the non-seed nodes are ordered so that at most one node is activated at each positive time step. Therefore, Constraint (4) also holds.

By construction, each active node is assigned to exactly one component label. Hence, Constraints (8) and (9) are satisfied. At time zero, distinct labels are assigned to distinct seed nodes, which gives Constraint (10). Furthermore, a node retains its current label or acquires a new one only from a neighboring node carrying that label at the previous time step. This proves that Constraint (11) is satisfied.

By the definition of $n_{i,r,t}$, its value is at most the number of nodes carrying label r , so Constraint (6) holds. Moreover, $n_{i,r,t}$ is positive only if node i is adjacent to at least one node carrying label r . Since every labeled component contains at most N nodes, Constraint (7) follows as well.

To verify Constraint (5), consider a node i activated at time $t \geq 1$, and let $C_1, \dots, C_k$ be the active connected components adjacent to i immediately before its activation. The labeling construction ensures that these components are represented by distinct labels and that i is assigned to one of them upon activation. Therefore,

$$\sum_{r=1}^N n_{i,r,t} = 1 + \sum_{q=1}^k |C_q|.$$

Since the given TD activation sequence is feasible, the right-hand side is at least θ_i , and thus Constraint (5) holds. If $x_{i,t} = 0$, the same constraint is trivially satisfied.

Finally, Constraints (12)–(14) follow directly from the definitions of x , y , and n . Hence, the constructed variables form a feasible solution of the integer programming formulation.

Conversely, let (x, y, n) be an integer feasible solution of the IP formulation. We define the corresponding seed set as

$$S := \{i \in V : x_{i,0} = 1, \theta_i > 1\}.$$

Notice that nodes with threshold one can activate for free without requiring any previously active component.

We first construct an auxiliary activation-time function T' over the extended time horizon, which we then use to define an activation sequence T for the TD process. By Constraint (3), for every $i \in V$ there exists a unique $t \geq 0$ such that $x_{i,t} = 1$. We define

$$T'(i) := t \quad \text{if and only if} \quad x_{i,t} = 1.$$

The values $T'(i)$ do not need to be consecutive, since the extended time horizon also contains label-update steps. Let π be a permutation of V such that

$$\pi(i) \leq \pi(j) \Rightarrow T'(i) \leq T'(j).$$

Ties among seed nodes activated at time zero are broken arbitrarily. We then define the activation sequence T by

$$T(i) := \pi(i) \quad \forall i = 1, \dots, N.$$

Thus, $T(i)$ is the position of node i in the activation order obtained from T' .

We claim that the activation sequence T is feasible for the TD problem. Seed nodes require no activation condition, while nodes with threshold one can always activate for free. Hence, consider a non-seed node i with $\theta_i > 1$, and let $t := T'(i)$ be its activation time in the extended horizon.

For every time step t , define

$$A_t := \{j \in V : T'(j) \leq t\}.$$

At the activation time $t = T'(i) > 0$ we have

$$A_t = \{i\} \cup \{j \in V : T'(j) < T'(i)\}.$$

Let C_i^t denote the connected component containing i in $G[A_t]$. We show that $|C_i^t| \geq \theta_i$.

For every $r \in \{1, \dots, N\}$, and $t \geq 0$, define the set of active nodes in component r at time t ,

$$Y_{r,t} := \{j \in V : y_{j,r,t} = 1\}.$$

All nodes in $Y_{r,t}$ belong to the same connected component of $G[A_t]$. This follows by induction on t . At time zero, Constraint (9) and (10) give $Y_{r,0} \subseteq A_0$ and $|Y_{r,0}| \leq 1$. For $t \geq 1$, Constraint (11) ensures that every node in $Y_{r,t}$ either already belonged to $Y_{r,t-1}$ or is adjacent to a node in $Y_{r,t-1}$. Moreover, Constraint (9) guarantees that all these nodes are active at time t .

Consider now any label r such that $n_{i,r,t} > 0$. Constraint (7) implies that i is adjacent to at least one node in $Y_{r,t}$. Hence, all nodes in $Y_{r,t}$ belong to C_i^t . Furthermore, Constraint (6) gives

$$n_{i,r,t} \leq |Y_{r,t}|.$$

Since Constraint (8) ensures that the sets $Y_{r,t}$ are pairwise disjoint, it follows that

$$\sum_{r=1}^N n_{i,r,t} \leq |C_i^t|.$$

Finally, since $x_{i,t} = 1$, Constraint (5) yields

$$\theta_i \leq \sum_{r=1}^N n_{i,r,t} \leq |C_i^t|.$$

Thus, every non-seed node satisfies the TD activation condition at its activation time. Therefore, the activation sequence defined by T , together with the seed set S , is feasible for the TD problem.

Finally, let (x, y, n) be an optimal solution of the integer programming formulation. The corresponding TD solution has seed set

$$S = \{i \in V : x_{i,0} = 1 \text{ and } \theta_i > 1\},$$

and therefore

$$|S| = \sum_{\substack{i \in V \\ \theta_i > 1}} x_{i,0} = \text{OPT}_{\text{IP}}.$$

Since S is feasible for the TD problem, it follows that

$$\text{OPT}_{\text{TD}} = \text{OPT}_{\text{IP}}.$$

2.3 Derived formulation for influence maximization

Some simple modifications to the formulation that we proposed for TD allow us to introduce the first exact IP formulation for the IM problem with component-based spread.

Formally, the problem is as follows: $G = (V, E)$ is, as before, an undirected graph, and $\theta : V \rightarrow \{1, \dots, |V|\}$ is the same threshold function as in TD.

Given a budget $B \in \mathbb{N}$, the goal is to select at most B initial active nodes, the seeds, in order to maximize the total number of nodes activated by the diffusion process. Using the operator Spread associated with the component-based diffusion dynamics, the problem can be written in compact form as

$$\begin{aligned} \max_{S \subseteq V} \quad & \text{Spread}(G, S, \theta) \\ \text{s.t.} \quad & |S| \leq B. \end{aligned}$$

As in the TD Problem, this compact formulation depends on the black-box operator Spread, whose value is obtained by simulating the component-based diffusion process starting from the selected seed set. Following the same modeling ideas used for TD, our formulation presents the following modifications.

1. **Objective function.** The objective function becomes

$$\max \sum_{i \in V} \sum_{t \geq 0} x_{i,t}$$

representing the total number of nodes that become active during the diffusion process.

2. **Modification of the activation-time constraint.** Because in IM not all the graph is influenced at the end, we modify the constraint that forces all nodes to get activated at exactly one time step (and therefore implies the activation of all the nodes) to

$$\sum_{t \geq 0} x_{i,t} \leq 1$$

allowing some nodes to never activate.

3. **Introduction of a budget constraint on seed nodes.** In the IM problem, the number of initially activated nodes is bounded by a prescribed budget parameter B . This is modeled through the additional constraint

$$\sum_{\substack{i \in V \\ \theta_i > 1}} x_{i,0} \leq B.$$

This constraint guarantees that at most B seed nodes can be selected at the beginning of the diffusion process.

All the remaining constraints stay unchanged, since they continue to describe the evolution of the component-based activation dynamics over time.

2.4 Empirical benefits of allowing disconnected activation sequences

We conclude this section by showing some experiments on small graphs that clearly show that the gap between the connected and the disconnected setting comes out in practice.

Following the experimental baseline of Goldberg and Liu [17], we generate preferential-attachment graphs in which each newly added vertex creates a degree chosen randomly from $\{1, 2, 3, 4\}$. For a fixed parameter c , node thresholds are sampled from

$$\left\{ \max\{2, c\}, 2c, 3c, \dots, \min\left\{\left\lceil \frac{n}{c} \right\rceil c, n\right\} \right\}.$$

We compare our formulation for possibly disconnected activation sequences with Goldberg–Liu’s formulation for connected activation sequences. We also report the theoretical upper bound,

$$\min\{2OPT, OPT + \ell\}.$$

To reduce the size of our new formulation, we introduced an upper bound on the number of labels that are initialized: exploiting the fact that a connected activation sequence always provides an upper bound on the real optimum of the problem, we first solved the formulation from Goldberg and Liu [17], and then used their solution to upper bound the number of labels r in our formulation, significantly reducing the number of variables that the solver would have to manage. Despite this, the substantially larger formulation required to model unrestricted activation sequences forced us to consider instances of relatively small size. Experiments are conducted on graphs with $n \in \{6, 8, 10, 12, 15, 20\}$.

Figure 1 reports the average seed-set size obtained for different graph sizes and threshold parameters c . The red line represents the optimum value of the unrestricted formulation, while the green line corresponds to Goldberg–Liu’s IP formulation. The black line shows the theoretical upper bound.

Even on small instances, the tests confirm that connectivity has a nonzero cost. We can observe that on these instances, the empirical price of connectivity is smaller than the worst-case increase allowed by the theoretical guarantee. The bound appears conservative, but it provides an estimate of the range in which disconnected activation sequences must lie and can be used as a cutoff to restrict the label space of the IP model.

3 Direct search algorithms for IM and TD

In this section, we introduce two algorithms. The first is tailored to explore the network for the IM problem using the TD propagation model. Building upon this framework, the second algorithm extends the approach to directly solve the TD problem.

The use of these zeroth-order procedures is motivated by the size and structure of the exact IP formulations introduced in the previous section. Indeed, although the IP models provide an explicit algebraic description of the component-based diffusion process, they require a large number of variables and constraints. Let $n = |V|$, let τ denote the number of time steps considered in the formulation, and let R be the number of component labels used to track active connected components. The TD formulation contains activation variables $x_{i,t}$, component-label variables $y_{i,r,t}$, and auxiliary variables $n_{i,r,t}$ measuring the size of the component labeled by r that is visible from node i at time t . Therefore, the number of variables is of order $O(nR\tau)$. Since, in the worst case, up to $R = O(n)$ component labels may be needed, and the time horizon may have to be enlarged in order to model the evolution and merging of connected components, the resulting formulation can become very large. In particular, when $\tau = O(n^2)$, the number of variables is of order $O(n^4)$.

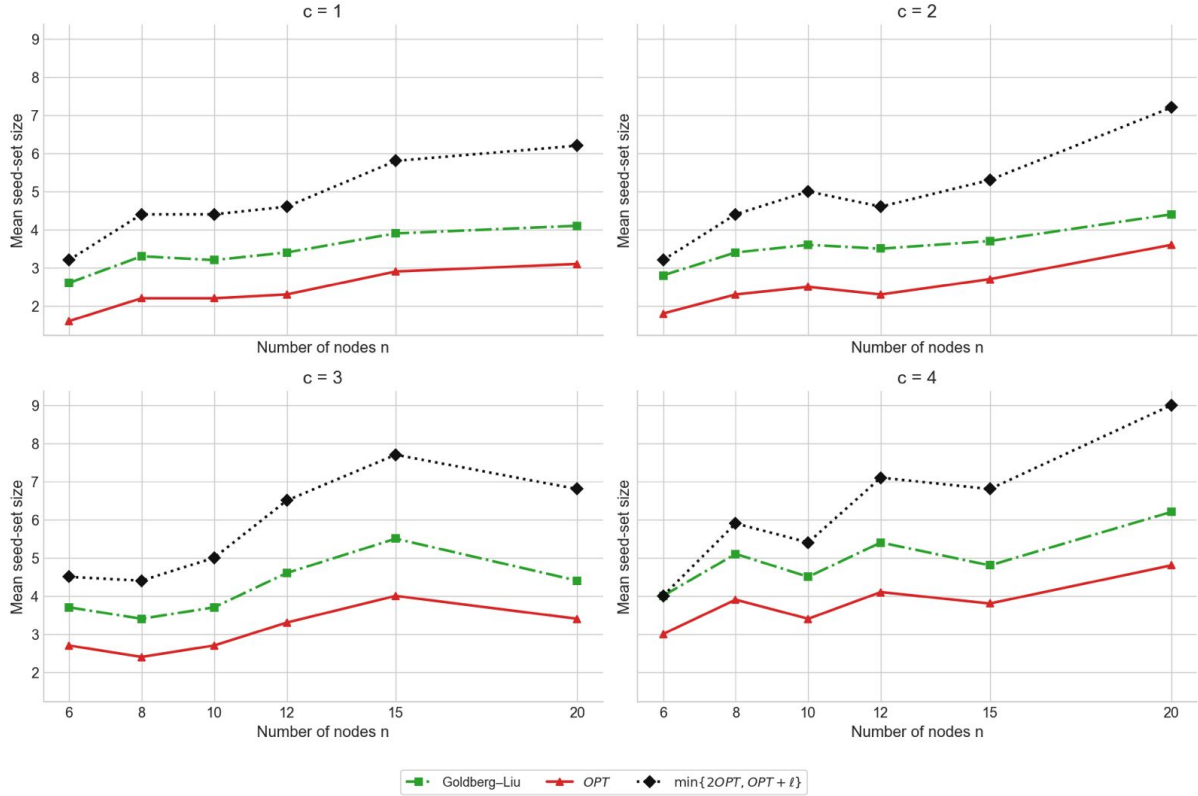


Fig. 1 Average seed-set size obtained by the unrestricted formulation, the Goldberg–Liu formulation for connected activation sequences, and the theoretical upper bound $\min\{2OPT, OPT + \ell\}$, for different graph sizes and threshold step-length parameters c .

A similar estimate holds for the number of constraints. Besides the assignment and activation constraints, the formulation must include constraints enforcing component membership, component visibility, component-size accounting, and the propagation of component labels over time. These constraints are again indexed by nodes, component labels, and time steps, leading to a number of constraints of order $O(nR\tau)$, which can also reach $O(n^4)$ in the worst case.

The exact IP formulation for the IM problem has essentially the same structural complexity. The full-activation constraint of TD is replaced by a budget constraint on the number of seeds, and the objective is changed so as to maximize the total number of activated nodes. However, the model still has to describe the same component-based diffusion dynamics, and therefore it requires the same families of activation, component-label, and component-size variables. Hence, also in the IM case, the number of variables and constraints grows as $O(n^4)$.

These size estimates explain why the exact formulations are mainly useful for modeling purposes and for solving small or moderate-size instances. For larger networks, directly solving the IP models with general-purpose solvers becomes computationally challenging. This motivates the development of zeroth-order approaches that treat the diffusion process as an evaluation oracle. Therefore, to support both algorithms, it is first necessary to develop a procedure capable of simulating the diffusion process according to the rules defined in Section 1.3. This diffusion evaluation algorithm serves as the foundation for assessing the effectiveness of candidate solutions to both the TD and IM problems, and is presented in Algorithm 1.

3.1 Component-Aware Direct Search

Having introduced the diffusion evaluation procedure in Algorithm 1, we now present a direct search approach for the IM problem under the TD model described in Section 1.3. The proposed method, referred to as the *Component-Aware Direct Search* (COaDS) algorithm, is reported in Algorithm 2. Starting from an initial seed set, COaDS iteratively explores candidate solutions belonging to suitably defined neighborhoods of the current incumbent. Each candidate is evaluated by applying the diffusion procedure described in Algorithm 1, and the incumbent is updated whenever a strictly improving candidate is

Algorithm 1 Connected Component Spread

Require: Graph $G = (V, E)$, seed set $S \subseteq V$, thresholds θ

```

1: active[v]  $\leftarrow$  1 if  $v \in S$ , and active[v]  $\leftarrow$  0 otherwise
2: repeat
3:   prev  $\leftarrow$  active
4:   Find the connected components  $C_1, \dots, C_q$  of the active subgraph using DFS
5:   Assign id[u]  $\leftarrow$   $k$  for each  $u \in C_k$ , and record  $|C_k|$ 
6:   for each  $v \in V$  such that prev[v] = 0 do
7:      $K(v) \leftarrow \{\text{id}[u] : u \in N_G(v), \text{prev}[u] = 1\}$ 
8:     if  $\sum_{k \in K(v)} |C_k| \geq \theta_v - 1$  then
9:       active[v]  $\leftarrow$  1
10:    end if
11:  end for
12: until active = prev
13: return active,  $\sum_{v \in V} \text{active}[v]$ 

```

identified. The method therefore operates exclusively through objective-function evaluations, without requiring gradient information or surrogate models. Its distinguishing feature is that both candidate generation and neighborhood exploration exploit the topology of the underlying graph, with the aim of identifying promising modifications while limiting the number of diffusion evaluations.

The procedure begins from an initial seed set $S^{(0)} \subseteq V$ satisfying $|S^{(0)}| = B$. Any constructive heuristic can be used to generate this solution. In the computational experiments, $S^{(0)}$ is obtained using the best-performing heuristic among those considered in Section 4. Its diffusion spread is then computed as $s^{(0)} = \text{Spread}(G, S^{(0)}, \theta)$, and $S^{(0)}$ becomes the first incumbent solution.

COaDS is controlled by the parameters $\zeta, \delta, d, \rho_{\max}, \nu_{\max}$, and $\lambda_{\min}$. The parameter $\zeta > 0$ determines the sufficient-increase threshold used as an opportunistic stopping criterion during neighborhood exploration, while $\delta \in (0, 1)$ controls the progressive reduction of this threshold. The POLL phase explores solutions at L_1 distance at most $2d$ from the incumbent, where $d \in \mathbb{N}$. The parameter $\rho_{\max} \in \mathbb{N}$ limits the maximum graph distance considered during the Marginal Gain Phase, whereas $\nu_{\max} \in \mathbb{N}$ bounds the number of candidates whose diffusion spread is fully evaluated at each search radius.

The parameter $\lambda_{\min} \in 0, \dots, B$ specifies the minimum connectivity requirement imposed on candidate solutions. This parameter is introduced to enforce a prescribed degree of connectivity within the seed set, as preliminary computational tests indicated that such a restriction is beneficial in practice. For a seed set S , $\lambda(S)$ denotes the size of the largest connected component of the subgraph induced by S . A candidate seed set is feasible only if $\lambda(S) \geq \lambda_{\min}$. Setting $\lambda_{\min} = B$ requires the entire seed set to induce a connected subgraph, whereas $\lambda_{\min} = 0$ removes the connectivity requirement.

At each iteration r , before entering the POLL phase, COaDS performs a *Marginal Gain Phase* designed to identify promising modifications of the incumbent solution by exploiting the topology of the graph. Let $S^{(r)}$ denote the current incumbent seed set. For each seed node $u \in S^{(r)}$, the algorithm considers nodes at increasing graph distance ρ from u , with $\rho = 1, \dots, \rho_{\max}$.

For each candidate node $v \notin S^{(r)}$ located at distance ρ from u , the algorithm computes the score

$$\Delta(v) = |\{w \in N_G(v) : \theta_w \leq \rho\}|, \quad (15)$$

where $N_G(v)$ denotes the set of neighbors of v . The score $\Delta(v)$ counts the neighbors of v whose thresholds do not exceed the current exploration radius. Candidates with larger values of $\Delta(v)$ are regarded as more promising, since their selection may facilitate the activation of a larger number of nearby nodes. To limit the number of expensive diffusion simulations, at most the $\nu_{\max}$ highest-ranked candidates are fully evaluated for each value of ρ .

For each selected candidate v , the algorithm identifies a closest seed node $u \in S^{(r)}$ and computes a shortest path $P(v, u)$ connecting them. The nodes of this path that do not already belong to $S^{(r)}$ are inserted into the candidate seed set. To preserve the cardinality constraint, the same number of incumbent seeds outside the newly added path are removed, yielding a candidate set S' satisfying $|S'| = B$. The removals are selected so that the connectivity condition $\lambda(S') \geq \lambda_{\min}$ is preserved. If no seed set satisfying both the cardinality and connectivity requirements can be constructed, the candidate is discarded without evaluating its diffusion spread.

The feasible candidates are evaluated in decreasing order of $\Delta(v)$. The Marginal Gain Phase terminates as soon as a candidate satisfying the sufficient-increase condition is identified. Otherwise, once all candidates up to radius $\rho_{\max}$ have been considered, the algorithm proceeds to the POLL phase. After

every update of the incumbent solution, the Marginal Gain Phase is performed again before the next POLL phase.

If the Marginal Gain Phase does not identify a sufficiently improving candidate, COaDS enters the POLL step, which constitutes the standard local exploration routine of the direct search method. The POLL step is divided into two phases. The first phase explores a small and strongly localized neighborhood of the incumbent and is intended to identify improving solutions at a limited computational cost. The second phase considers a substantially larger neighborhood and is therefore more expensive. In practice, the algorithm is designed to avoid entering this second phase during most iterations; it is mainly used near the end of the search to verify local optimality with respect to the prescribed neighborhood.

To define the two POLL neighborhoods, let $z \in \{0, 1\}^n$ be the incidence vector of the current seed set and define

$$N(z, 2d) = \{y \in \{0, 1\}^n : 0 < \|y - z\|_1 \leq 2d, \sum_{i \in V} y_i = B\}, \quad (16)$$

where $d \in \mathbb{N}$. Since all feasible solutions contain exactly B selected nodes, their mutual L_1 distance is always even. In particular, a distance of at most $2d$ corresponds to replacing at most d incumbent seeds with the same number of non-selected nodes.

The localized component of the search is described by

$$C(z) = \{y \in \{0, 1\}^n : \forall i \in V, y_i = 1 \Rightarrow (z_i = 1 \vee N_G(i) \cap S(z) \neq \emptyset), \sum_{i \in V} y_i = B\}. \quad (17)$$

The neighborhood explored during the first POLL phase is

$$N^1(z) = \{y \in N(z, 2) \cap C(z) : \lambda(S(y)) \geq \lambda_{\min}\}, \quad (18)$$

where $S(y) = \{i \in V : y_i = 1\}$.

If Phase 1 does not produce a sufficiently improving solution, the algorithm enters the second POLL phase and explores

$$N_d^2(z) = \{y \in N(z, 2d) : \lambda(S(y)) \geq \lambda_{\min}\}. \quad (19)$$

Unlike $N^1(z)$, the expanded neighborhood $N_d^2(z)$ is not restricted by $C(z)$ and may therefore include replacements involving nodes that are not adjacent to the incumbent seed set. It contains all feasible solutions obtainable through at most d simultaneous seed substitutions. This broader exploration increases the ability of the method to escape configurations that are locally optimal with respect to one-swap moves, but its size grows rapidly with d . For this reason, Phase 2 is invoked only after both the Marginal Gain Phase and the first POLL phase have failed, and its main role is to provide a final, stronger local-optimality check.

At the end of each iteration, the algorithm performs a termination check based on the candidates examined during the Marginal Gain and POLL phases. Let S' be the best improving candidate identified during the iteration, and let

$$s' = \text{Spread}(G, S', \theta). \quad (20)$$

A candidate is said to provide a sufficient increase if

$$s' > (1 + \zeta)s^{(r)}, \quad (21)$$

where $\zeta > 0$ is the current sufficient-increase parameter. This condition is used throughout the Marginal Gain and POLL phases as an opportunistic stopping rule: as soon as a candidate satisfying (21) is found, the remaining candidates in the current iteration need not be evaluated.

If the iteration produces a candidate satisfying $s' > s^{(r)}$, the incumbent is updated according to

$$S^{(r+1)} = S', \quad s^{(r+1)} = s'. \quad (22)$$

When the improvement is positive but does not satisfy the sufficient-increase condition, the candidate is still accepted, but the requirement is relaxed by setting

$$\zeta \leftarrow \delta\zeta, \quad (23)$$

where $\delta \in (0, 1)$. This reduction progressively weakens the sufficient-increase threshold, allowing smaller improvements to trigger early termination as the search approaches a locally optimal solution. After updating the incumbent, the iteration counter is increased and the algorithm restarts from the Marginal Gain Phase.

If neither the Marginal Gain Phase nor either phase of the POLL step identifies a strictly improving candidate, the current incumbent is locally optimal with respect to the neighborhoods explored by the algorithm, and the search terminates. The procedure also stops when the prescribed computational time limit is reached, in which case the best incumbent solution found so far is returned.

Algorithm 2 Component-Aware Direct Search (COaDS) Method

Require: Graph $G = (V, E)$, thresholds θ , seed-set size B , parameters $\zeta > 0$, $\delta \in (0, 1)$, $d \in \mathbb{N}$, maximum radius $\rho_{\max}$, maximum number of evaluated candidates per radius $\nu_{\max}$, and minimum connectivity requirement $\lambda_{\min}$

- 1: **Initialization:** Let $S^{(0)} \subseteq V$ be a suitably chosen seed set such that $|S^{(0)}| = B$. Set $s^{(0)} \leftarrow \text{Spread}(G, S^{(0)}, \theta)$ and $r \leftarrow 0$.
- 2: **Marginal Gain Phase:** For each radius $\rho = 1, \dots, \rho_{\max}$, rank the candidate nodes at distance ρ from $S^{(r)}$ according to their marginal-gain score $\Delta(v)$. Consider at most the $\nu_{\max}$ highest-ranked candidates and, for each of them, construct a feasible candidate seed set S' . Evaluate $\text{Spread}(G, S', \theta)$ and retain the best improving candidate found. If $\text{Spread}(G, S', \theta) > (1 + \zeta)s^{(r)}$, terminate the phase and go directly to the termination check.
- 3: **POLL step (Phase 1):** Let $z^{(r)}$ be the incidence vector of $S^{(r)}$. Evaluate the seed sets $S(y)$ for $y \in N^1(z^{(r)})$, retaining the best improving candidate found. If a candidate satisfies the sufficient-increase condition, terminate the POLL step and go directly to the termination check. If no such candidate is found after exhausting $N^1(z^{(r)})$, proceed to Phase 2.
- 4: **POLL step (Phase 2):** Evaluate the seed sets $S(y)$ for $y \in N_d^2(z^{(r)})$, retaining the best improving candidate found. Terminate the phase when a candidate satisfying the sufficient-increase condition is found or when the neighborhood has been exhausted.
- 5: **Termination check:** Let S' be the best candidate identified during the current iteration and set $s' \leftarrow \text{Spread}(G, S', \theta)$. If $s' > s^{(r)}$, set $S^{(r+1)} \leftarrow S'$ and $s^{(r+1)} \leftarrow s'$. If $s' \leq (1 + \zeta)s^{(r)}$, also set $\zeta \leftarrow \delta\zeta$. Set $r \leftarrow r + 1$ and return to the Marginal Gain Phase. If no improving candidate has been identified, terminate the search.
- 6: **return** $S^{(r)}$

3.2 COaDS binary search for TD

The TD problem requires finding a minimum-cardinality seed set that activates the whole network. Instead of tackling this minimization problem directly, we exploit the IM version of the component-based diffusion model as a subroutine. The idea is to fix a candidate seed-set cardinality and use COaDS to search for a seed set with maximum spread under that budget. This transforms the TD problem into a sequence of fixed-budget spread-maximization problems.

The detailed scheme of the proposed procedure is reported in Algorithm 3. The algorithm embeds COaDS within a binary-search strategy on the seed-set cardinality. More specifically, for a given cardinality B , COaDS is applied to identify a seed set of size B that maximizes the diffusion spread. The solution returned by COaDS is then evaluated with respect to the TD goal, namely whether it activates the entire graph.

The binary-search strategy is motivated by the monotonicity of the diffusion process with respect to the seed set. Adding nodes to the initial seed set cannot indeed decrease the final number of activated nodes. Therefore, if full activation is achieved with B seeds, then all larger cardinalities are also feasible. Conversely, in an exact setting, if no feasible seed set of cardinality B exists, then no smaller cardinality can be feasible either. Hence, this property makes it possible to search over seed-set cardinalities by updating lower and upper endpoints, rather than testing all possible values sequentially. In our algorithm, however, the second implication has to be interpreted carefully. Since COaDS is a heuristic procedure, its failure to find a feasible seed set of size B does not constitute a formal certificate of infeasibility for that cardinality. Thus, the update of the lower bound is heuristic in nature. Nevertheless, this strategy is effective in practice and allows the method to focus the search on the most promising range of cardinalities.

The procedure is initialized with a feasible seed set S^H generated by the best-performing heuristic. Since S^H activates the whole graph, its cardinality provides an initial upper bound on the optimal TD value. We denote by B_L and B_U the current lower and upper endpoints of the search interval, respectively. Moreover, S^* denotes the best feasible solution found so far, and $B^* = |S^*|$ denotes its cardinality.

At each iteration, the algorithm tests the midpoint of the current search interval, defined as

$$B = \left\lfloor \frac{B_L + B_U}{2} \right\rfloor.$$

A seed set $S^{(0)}$ of cardinality B is first generated by the heuristic and then used as the initial solution for COaDS. Let S_B be the seed set returned by COaDS. If

$$\text{Spread}(G, S_B, \theta) = |V|,$$

then S_B is feasible for the TD problem. In this case, the incumbent solution is updated by replacing S^* with S_B and setting $B^* = B$. Since a feasible solution with cardinality B has been found, the algorithm then continues the search on smaller cardinalities.

If full activation is not achieved, the algorithm exploits the information returned by the diffusion simulation to construct a feasible solution. Let

$$A(G, S, \theta) \subseteq V$$

denote the final set of nodes activated by the diffusion process starting from a seed set S , and let

$$I(G, S, \theta) = V \setminus A(G, S, \theta)$$

be the set of nodes that remain inactive after propagation. Starting from the solution S_B returned by COaDS, a feasible TD solution can always be obtained by adding all inactive nodes directly to the seed set. Accordingly, the algorithm defines

$$S_{\text{ext}} = S_B \cup I(G, S_B, \theta)$$

and

$$B_{\text{ext}} = B + |I(G, S_B, \theta)|.$$

The set S_{ext} is feasible for TD because all nodes that are not activated by S_B are added directly as seeds. Hence, every node is either activated through the diffusion process or selected in the augmented seed set, so B_{ext} provides a valid upper bound on the optimal TD value. If this bound improves the current incumbent, that is, if $B_{\text{ext}} < B^*$, the incumbent solution is replaced by S_{ext} and the incumbent value is updated to B_{ext} . Moreover, since a feasible solution of size B_{ext} is now known, the remaining search interval can be tightened by setting the upper bound to $\min\{B_U, B_{\text{ext}} - 1\}$, since cardinalities at least B_{ext} cannot improve the current best solution. Finally, due to the COaDS failure, the current budget B is discarded and the search continues on larger cardinalities by updating the lower bound to $B + 1$. The procedure terminates once there are no cardinalities left to test, that is, when the lower bound becomes larger than the upper bound. At that point, all relevant budgets have either been discarded or used to improve the incumbent, and the algorithm returns the best feasible seed set identified during the search.

Algorithm 3 COaDS Binary Search for Technology Diffusion

Require: Graph $G = (V, E)$, thresholds θ

```

1: Compute  $s_\emptyset \leftarrow \text{Spread}(G, \emptyset, \theta)$ 
2: if  $s_\emptyset = |V|$  then
3:   return  $\emptyset, 0$ 
4: end if
5: Generate a feasible heuristic solution  $S^H$ 
6:  $S^* \leftarrow S^H$ ,  $B^* \leftarrow |S^H|$ 
7:  $B_L \leftarrow 1$ ,  $B_U \leftarrow B^* - 1$ 
8: while  $B_L \leq B_U$  do
9:    $B \leftarrow \lfloor (B_L + B_U)/2 \rfloor$ 
10:  Generate a heuristic seed set  $S^{(0)}$  such that  $|S^{(0)}| = B$ 
11:  Apply COaDS with budget  $B$ , starting from  $S^{(0)}$ , and let  $S_B$  be the resulting seed set
12:  if  $\text{Spread}(G, S_B, \theta) = |V|$  then
13:     $S^* \leftarrow S_B$ ,  $B^* \leftarrow B$ 
14:     $B_U \leftarrow B - 1$ 
15:  else
16:     $I_B \leftarrow I(G, S_B, \theta)$ 
17:     $S_{\text{ext}} \leftarrow S_B \cup I_B$ 
18:     $B_{\text{ext}} \leftarrow B + |I_B|$ 
19:    if  $B_{\text{ext}} < B^*$  then
20:       $S^* \leftarrow S_{\text{ext}}$ ,  $B^* \leftarrow B_{\text{ext}}$ 
21:    end if
22:     $B_U \leftarrow \min\{B_U, B_{\text{ext}} - 1\}$ 
23:     $B_L \leftarrow B + 1$ 
24:  end if
25: end while
26: return  $S^*, B^*$ 

```

Theorem 4 Let $\{z^{(r)}\}$ and $\{s^{(r)}\}$ be the sequences of incumbent solutions and spread values generated by COaDS for a fixed budget B . Assume $\lambda_{\min} = 0$. Then for any fixed $d \geq 1$, the algorithm produces a d -swap local maximum in finite time, namely a solution $z^{(\bar{r})}$ such that

$$s^{(\bar{r})} \geq \text{Spread}(G, S(y), \theta) \quad \forall y \in N(z^{(\bar{r})}, 2d).$$

Proof Let $z^{(\bar{r})}$ be the output of COaDS. Since $\lambda_{\min} = 0$, the final **POLL step** ensures that every point y in the neighborhood $N(z^{(\bar{r})}, 2d)$ has been evaluated and that no such point has spread greater than

$s^{(\bar{r})}$. This guarantees that $z^{(\bar{r})}$ is a d -swap local maximum. If the algorithm returns upon reaching full spread, the same conclusion follows directly from $s^{(\bar{r})} = n$.

To prove that COaDS produces the sequence $\{z^{(r)}\}$ in finite time, observe that

$$s^{(0)} < s^{(1)} < \dots < s^{(\bar{r})},$$

since every accepted incumbent update strictly increases the spread. Thus, no incumbent solution is visited twice. Furthermore, the fixed-budget domain contains only finitely many seed sets, so the sequence of incumbents is finite. Each iteration examines finitely many candidates, and each diffusion evaluation terminates on the finite graph. Therefore, the algorithm terminates in finite time.

Remark 1 Algorithm 3 uses COaDS as an imperfect oracle for the fixed-budget component-spread subproblem. Although finding full spread certifies that a budget is feasible for TD, failure to find full spread does not prove that the budget is infeasible. Consequently, the binary-search update of the lower endpoint may discard a feasible budget, including the optimal one. The local convergence of COaDS therefore does not guarantee convergence of the outer procedure to a globally optimal TD solution.

4 Numerical results

In this section, we present numerical experiments conducted on the TD problem. The objective is to demonstrate that the proposed algorithm scales significantly better to large networks than traditional approaches, such as greedy algorithms and integer programming formulations solved by commercial optimization solvers. At the same time, we show that it consistently outperforms existing heuristic methods on these larger instances.

All methods and experiments were implemented in Python. The source code for both the algorithms and the experimental framework is publicly available at <https://github.com/Berga53/technology-diffusion>. All experiments in this section were conducted on a machine equipped with an Apple M4 chip (10-core CPU and 16 GB of unified memory).

The numerical tests focused on synthetic instances generated through a preferential attachment mechanism. The choice of this model is motivated by its ability to reproduce highly heterogeneous degree distributions, a characteristic commonly observed in real-world social, communication, and technological networks. Such topological features are particularly relevant in diffusion processes, where highly connected nodes may play a crucial role in the propagation dynamics. Starting from a small initial graph, nodes are added sequentially and connected to existing vertices with probability proportional to their degree. Six network sizes were considered, corresponding to $n \in \{200, 400, 600, 1000, 2000, 4000\}$.

Given a parameter c , threshold values were sampled uniformly from the set

$$\left\{ \max\{2, c\}, 2c, 3c, \dots, \min\left\{\left\lceil \frac{n}{c} \right\rceil c, n\right\} \right\}.$$

This construction yields heterogeneous activation requirements while preserving a consistent scaling with the network size. The parameter c controls the granularity of the threshold distribution and the minimum activation requirement, thereby generating instances with different diffusion characteristics. For the experiments we considered $c \in \{1, 5, 10, 20\}$.

For each combination of n and c , 25 independent instances were generated, resulting in a total of 600 benchmark instances.

The proposed COaDS algorithm is compared against several approaches for the TD problem.

– Heuristic methods:

- **Degree**: selects nodes in decreasing order of degree.
- **Degree Threshold**: selects nodes in decreasing order of the product between degree and threshold.
- **Betweenness**: selects nodes in decreasing order of betweenness centrality.
- **Degree Discount**: selects nodes according to the discounted-degree strategy proposed in [8], where the degree is computed on the subgraph induced by inactive nodes.
- **Degree Connected**: selects nodes in decreasing order of degree among those connected to the currently active nodes.
- **Greedy approximation algorithm**: the approximation algorithm proposed in [21].
- **Goldberg and Liu formulation** [17]: the integer programming formulation proposed by Goldberg and Liu, solved using Gurobi.

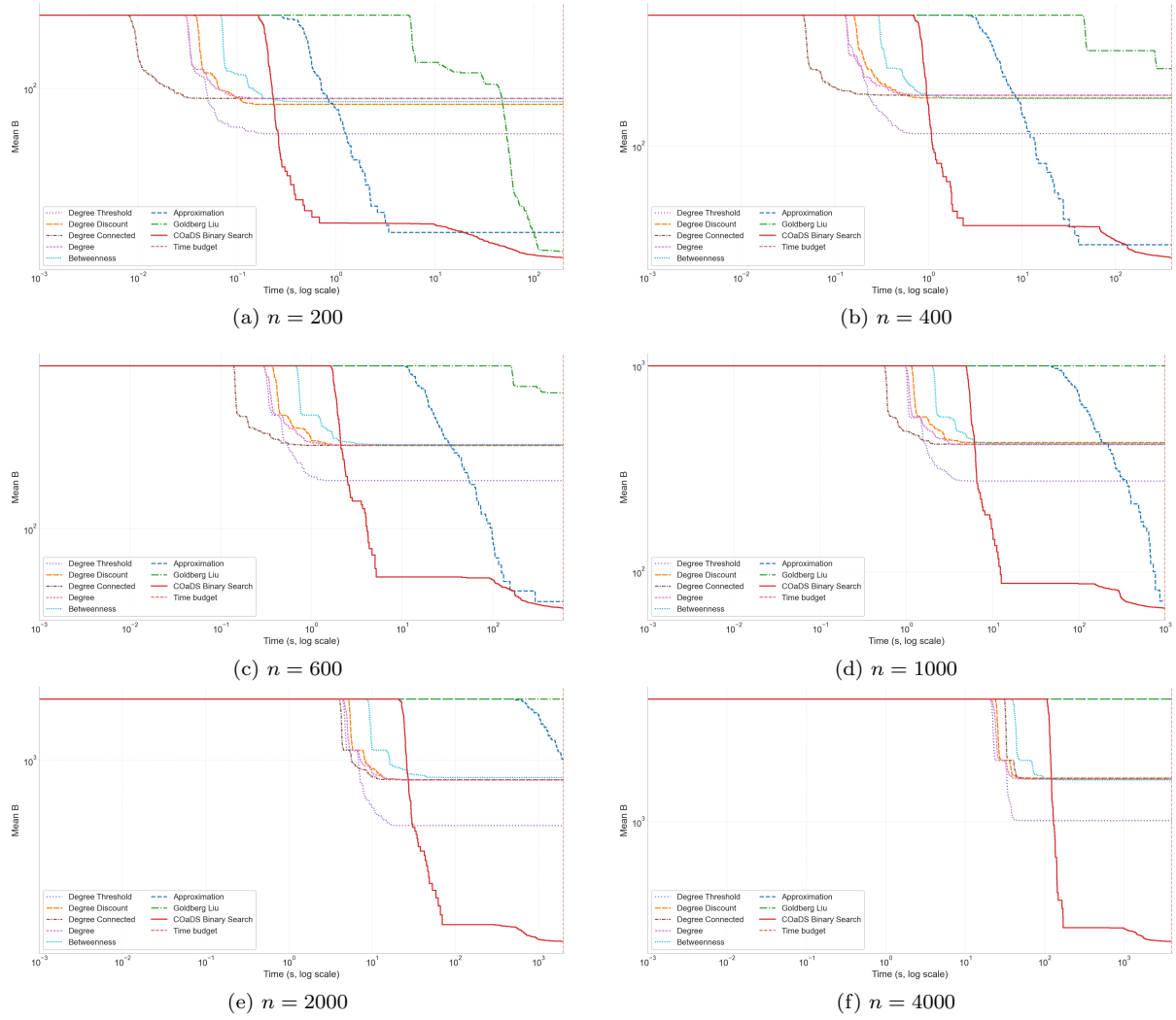


Fig. 2 Evolution of the mean incumbent seed-set cardinality over time for different graph sizes. For each fixed value of n , the curves are averaged over all corresponding benchmark instances. The vertical dashed line denotes the computational time limit.

Performance is evaluated in terms of both solution quality, measured by the cardinality of the resulting seed set, and computational efficiency. To ensure a fair comparison, all methods are executed under the same computational budget, with a time limit equal to n seconds for an instance containing n nodes. When a method does not find a feasible solution within the available computational resources, its solution value is set to $B = n$. This corresponds to the worst possible value and ensures that unsuccessful runs are retained in all aggregate comparisons.

Figure 2 reports the evolution of the mean incumbent value over time for each graph size. For every method, the curve is initialized at n and decreases whenever a better feasible seed set is found on one or more instances. The heuristic methods identify their final solutions very rapidly, generally within the first seconds, but their curves subsequently remain constant. Among them, the degree-threshold heuristic consistently provides the best final solutions, whereas degree, degree-discount, degree-connected, and betweenness produce similar and substantially larger seed sets.

The COaDS-based binary-search procedure exhibits a different anytime behavior. Its first improvements are obtained later than those of the constructive heuristics, since each candidate cardinality requires solving an associated IM problem. Nevertheless, after the initial search phase, its incumbent value decreases sharply and eventually becomes smaller than that of all the heuristic methods for every graph size. COaDS also continues to improve during the remaining computational budget, showing that the additional running time is effectively exploited to refine the solution rather than merely retaining an incumbent found at the beginning of the execution.

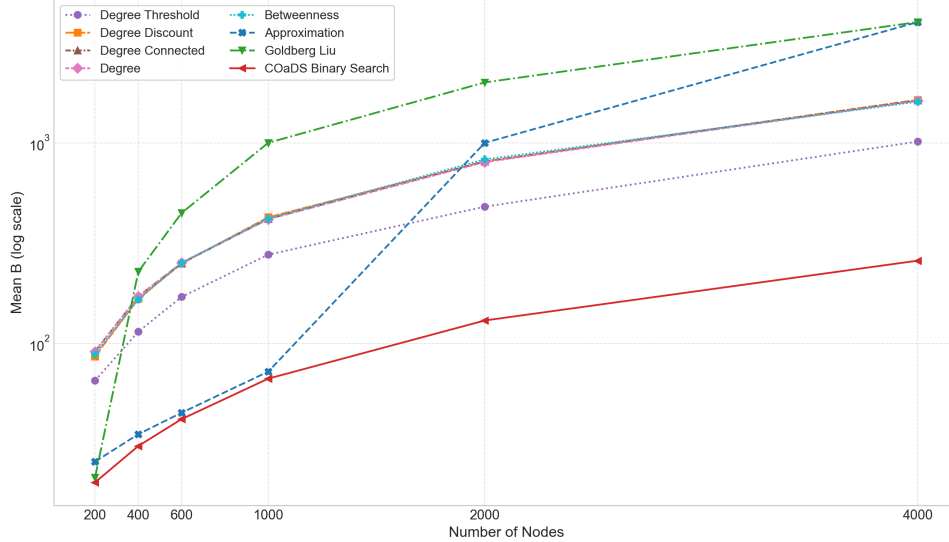


Fig. 3 Mean seed-set cardinality obtained by each method as a function of the number of nodes. The vertical axis is shown on a logarithmic scale.

For the smaller instances, the approximation algorithm is the closest competitor to COaDS. In particular, for $n = 200, 400$, and 600 , it eventually reaches mean seed-set cardinalities comparable to those obtained by COaDS, although it generally requires a substantial portion of the available time. As the graph size increases, however, its performance deteriorates rapidly. For $n = 1000$, COaDS already attains a better final mean value, while for $n = 2000$ the approximation algorithm remains far from the solutions produced by COaDS at the end of the time limit. For $n = 4000$, the approximation algorithm does not find any feasible solution within the available computational budget.

The formulation of Goldberg and Liu is competitive only on the smallest instances. Its computational requirements increase rapidly with the graph size, and the quality of the incumbents obtained within the prescribed time budget deteriorates already for the intermediate sizes. From $n = 1000$ onward, the formulation can no longer be effectively applied because of its excessive computational and memory requirements. This behavior highlights the limited scalability of the exact formulation on the considered benchmark.

The aggregate results in Figure 3 confirm these observations. The mean cardinality returned by all methods increases with the number of nodes, but the rate of deterioration differs considerably. The heuristic methods scale approximately in proportion to the graph size and preserve a large gap from the best solutions. The degree-threshold heuristic remains the strongest heuristic baseline, but its gap with respect to COaDS widens as n increases.

The approximation algorithm performs well on the smaller and medium-sized graphs, but its mean solution value grows sharply for the largest instances. This behavior is partly explained by the more difficult instances associated with small values of c , on which the approximation algorithm becomes increasingly expensive and is unable to complete enough iterations within the available time. Similarly, the Goldberg–Liu formulation cannot be effectively applied from $n = 1000$ onward because of its computational and memory requirements.

By contrast, COaDS maintains the lowest mean cardinality for every tested graph size and exhibits a substantially slower growth in solution size. In particular, it continues to find feasible solutions for $n = 4000$, while both the approximation algorithm and the integer programming formulation are assigned the value $B = n$.

Figure 4 reports the aggregate performance profiles based on seed-set cardinality. This representation compares the algorithms over the entire benchmark without directly averaging cardinalities from instances of different sizes. For each algorithm s and instance i , the performance ratio is defined as

$$r_{s,i} = \frac{B_{s,i}}{\min_{s'} B_{s',i}},$$

and the corresponding performance profile is

$$\rho_s(\tau) = \frac{|\{i \in \mathcal{I} : r_{s,i} \leq \tau\}|}{|\mathcal{I}|}.$$

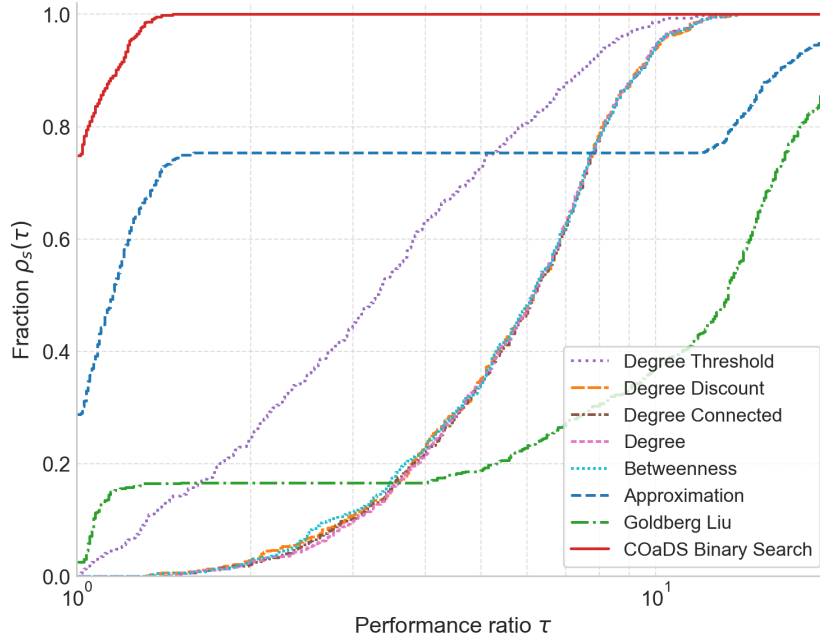


Fig. 4 Aggregate performance profiles based on seed-set cardinality over all benchmark instances. For a performance ratio τ , $\rho_s(\tau)$ denotes the fraction of instances on which algorithm s returns a seed set whose cardinality is at most τ times the best observed cardinality for that instance.

Therefore, $\rho_s(1)$ is the fraction of instances on which algorithm s attains the best observed solution, while a profile closer to the upper-left corner indicates stronger and more robust performance.

COaDS clearly dominates the other methods. It attains the best observed cardinality on a large majority of the instances and reaches $\rho_s(\tau) = 1$ for a performance ratio only slightly larger than one. Thus, even when COaDS does not return the best observed solution, its cardinality remains close to it.

Degree Threshold is the strongest heuristic according to the profile. Nevertheless, it is best only on a small fraction of the instances and requires considerably larger values of τ to cover the complete benchmark. Degree, Degree Discount, Degree Connected, and Betweenness have similar profiles and are generally several times worse than the best observed solution.

The approximation algorithm is competitive on part of the benchmark and attains the best observed solution on a non-negligible fraction of the instances. However, its profile contains a long plateau and reaches the largest values of τ only after a marked increase, reflecting its loss of effectiveness on the larger instances. In particular, on the $n = 4000$ instances it does not find a feasible solution and is therefore assigned the value $B = n$.

The Goldberg and Liu formulation has the weakest overall profile. It is competitive on a small fraction of the smallest instances, but from $n = 1000$ onward it does not find a feasible solution within the available computational resources and is assigned the value $B = n$. Its profile therefore reflects both the solution quality obtained on the smaller graphs and the loss of effectiveness on larger networks.

To examine the dependence of solution quality on both the graph size and the threshold parameter, Figure 5 reports the average relative gap for each pair (n, c) . Let $B_{s,i}$ denote the seed-set cardinality returned by algorithm s on instance i , and define

$$B_i^{\min} = \min_{s'} B_{s',i}.$$

The relative gap of algorithm s on instance i is given by

$$g_{s,i} = 100 \frac{B_{s,i} - B_i^{\min}}{B_i^{\min}}.$$

The heatmap reports the mean value of $g_{s,i}$ over the 25 instances generated for each pair (n, c) . As described above, when a method does not find a feasible solution within the available computational resources, its solution value is set to $B = n$.

COaDS exhibits the smallest average relative gaps in almost all configurations. Its gaps remain close to zero for the smaller graphs and remain substantially below those of the competing methods as the

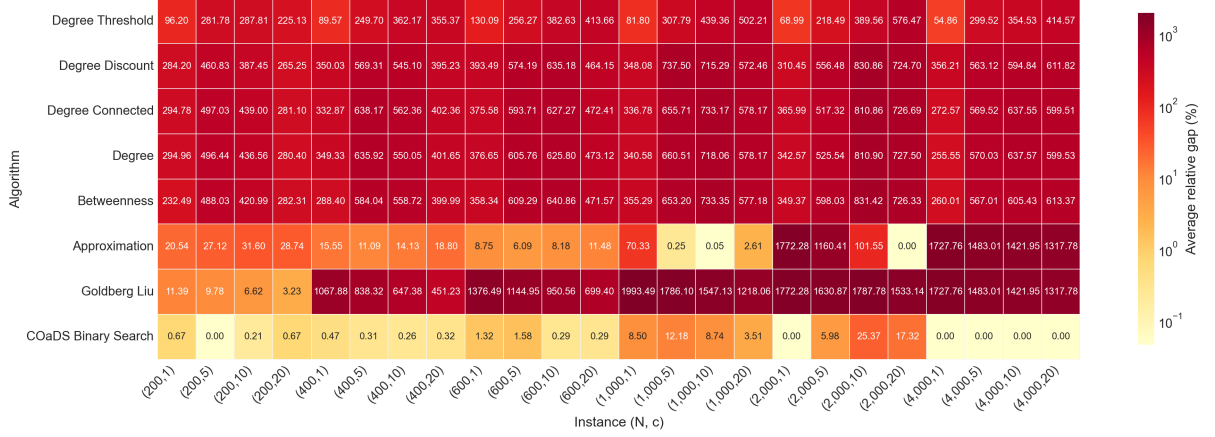


Fig. 5 Average relative gap in seed-set cardinality for each algorithm and each pair (n, c) . For each instance, the gap is computed with respect to the smallest seed-set cardinality obtained by any of the considered methods. Each cell reports the average over the 25 instances associated with the corresponding pair (n, c) . The color scale is logarithmic.

network size increases. For $n = 4000$, COaDS attains a zero average gap for every value of c , meaning that it returns the best observed solution on all corresponding benchmark groups. This does not establish optimality, since the comparison is made with respect to the best solution found by the tested methods, but it shows that COaDS remains effective in the regime where the more computationally demanding competitors fail.

The heuristic methods exhibit considerably larger gaps. Degree Threshold, Degree, Degree Discount, Degree Connected, and Betweenness return solutions several times larger than the best observed ones on most configurations and display very similar behavior across the benchmark.

The heatmap also reveals a marked sensitivity of the approximation algorithm to the threshold parameter c . For larger values of c , it remains competitive on several small and medium-sized configurations and occasionally obtains solutions close to the best observed value. Its gaps are much larger for $c = 1$, however, and increase sharply with the graph size. On the $n = 4000$ instances, the approximation algorithm does not find a feasible solution for any value of c , producing very large relative gaps.

The Goldberg and Liu formulation exhibits an even stronger loss of scalability. It is competitive only on the smallest instances, while from $n = 1000$ onward it does not find a feasible solution within the available computational resources. Its behavior is dominated by the graph size, with large relative gaps across all values of c once the formulation ceases to find feasible solutions.

Overall, the incumbent histories, aggregate mean cardinalities, aggregate performance profile, and relative-gap heatmap lead to consistent conclusions. COaDS provides the best compromise between solution quality and scalability among the considered methods. The heuristic approaches remain computationally inexpensive but return substantially larger seed sets, the approximation algorithm is strongly affected by both the graph size and the threshold parameter, and the Goldberg and Liu formulation becomes ineffective beyond the smallest benchmark instances.

5 Conclusion

In this work, we studied component-based diffusion processes on networks, focusing on the Technology Diffusion problem and on the related Influence Maximization problem induced by the same propagation dynamics. We first investigated the role of connected activation sequences, a structural assumption commonly used in previous studies. We showed that restricting the diffusion process to connected activation sequences can require additional seed nodes compared with unrestricted diffusion processes, and provided a refined theoretical characterization of this gap. Our results clarify the limitations of enforcing connectivity and motivate the study of diffusion processes where active components are allowed to evolve and merge without such constraints.

We then introduced new exact integer programming formulations for the diffusion problems under analysis that do not rely on connected activation sequences. These formulations provide a direct representation of the component-based diffusion dynamics and allow the unrestricted problem to be analyzed through mathematical programming tools. However, the resulting models have a large number of variables and constraints, limiting their applicability to small and moderate-size instances. Computational

experiments on some toy problems, however, confirm that allowing disconnected activation sequences can lead to improved solutions compared with connectivity-based formulations.

Finally, motivated by the computational limitations of existing approaches, we developed a Component-Aware Direct Search method (COaDS) with the goal of addressing larger-scale instances than those previously considered in the literature. Indeed, by treating the diffusion process as a black-box procedure and exploiting both the network structure and the component-based activation mechanism, COaDS efficiently explores the feasible seed sets. Experimental results on preferential-attachment networks showed that the proposed approach obtains high-quality solutions and outperforms the considered integer-programming, heuristic and approximation baselines.

Future work may focus on further improving the scalability of component-aware search strategies, investigating additional structural properties of component-based diffusion, and extending the approach to broader classes of network diffusion models.

References

- [1] Mark A. Abramson, Charles Audet, John W. Chrissis, and J. G. Walston. “Mesh Adaptive Direct Search Algorithms for Mixed Variable Optimization”. In: *Optimization Letters* 3 (2009), pp. 35–47. DOI: 10.1007/s11590-008-0089-2.
- [2] Charles Audet and J. E. Dennis Jr. “Pattern Search Algorithms for Mixed Variable Programming”. In: *SIAM Journal on Optimization* 11.3 (2000), pp. 573–594. DOI: 10.1137/S1052623499352024.
- [3] Charles Audet and Warren Hare. *Derivative-Free and Blackbox Optimization*. Cham: Springer, 2017. DOI: 10.1007/978-3-319-68913-5.
- [4] Charles Audet, Sébastien Le Digabel, and Christophe Tribes. “The Mesh Adaptive Direct Search Algorithm for Granular and Discrete Variables”. In: *SIAM Journal on Optimization* 29.2 (2019), pp. 1164–1189. DOI: 10.1137/18M1175872.
- [5] Matteo Bergamaschi, Sara Venturini, Francesco Tudisco, and Francesco Rinaldi. “An Efficient Network-aware Direct Search Method for Influence Maximization”. In: *Journal of Complex Networks* 13.6 (2025), cnaf042. DOI: 10.1093/comnet/cnaf042.
- [6] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. “Maximizing Social Influence in Nearly Optimal Time”. In: *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Ed. by Chandra Chekuri. Philadelphia: Society for Industrial and Applied Mathematics, 2014, pp. 946–957. DOI: 10.1137/1.9781611973402.70.
- [7] Wei Chen, Chi Wang, and Yajun Wang. “Scalable Influence Maximization for Prevalent Viral Marketing in Large-Scale Social Networks”. In: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York: Association for Computing Machinery, 2010, pp. 1029–1038. DOI: 10.1145/1835804.1835934.
- [8] Wei Chen, Yajun Wang, and Siyu Yang. “Efficient Influence Maximization in Social Networks”. In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York: Association for Computing Machinery, 2009, pp. 199–208. DOI: 10.1145/1557019.1557047.
- [9] Edith Cohen, Daniel Delling, Thomas Pajor, and Renato F. Werneck. “Sketch-based Influence Maximization and Computation: Scaling up with Guarantees”. In: *Proceedings of the 23rd ACM International Conference on Information Management and Knowledge Management*. New York: Association for Computing Machinery, 2014, pp. 629–638. DOI: 10.1145/2661829.2662077.
- [10] Andrew R. Conn, Katya Scheinberg, and Luís N. Vicente. *Introduction to Derivative-Free Optimization*. Philadelphia: Society for Industrial and Applied Mathematics, 2009. ISBN: 9780898716689. DOI: 10.1137/1.9780898718768.
- [11] Alberto Costa and Giacomo Nannicini. “RBFOpt: An Open-source Library for Black-box Optimization with Costly Function Evaluations”. In: *Mathematical Programming Computation* 10 (2018), pp. 597–629. DOI: 10.1007/s12532-018-0144-7.
- [12] K. J. Dzahini, F. Rinaldi, C. W. Royer, and D. Zeffiro. “Direct-search Methods in the Year 2025: Theoretical Guarantees and Algorithmic Paradigms”. In: *EURO Journal on Computational Optimization* 13 (2025), p. 100110. DOI: 10.1016/j.ejco.2025.100110.
- [13] Hossein Esfandiari, MohammadTaghi Hajiaghayi, Jochen Könnemann, Hamid Mahini, David L. Malec, and Laura Sanità. “Approximate Deadline-Scheduling with Precedence Constraints”. In: *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 7-9, 2015, Proceedings*. Vol. 9294. Lecture Notes in Computer Science. Springer, 2015, pp. 483–495. DOI: 10.1007/978-3-662-48350-3_41.

- [14] Vinicius Ferreira, Artur Pessoa, and Thibaut Vidal. “Influence optimization in networks: New formulations and valid inequalities”. In: *Computers & Operations Research* 173 (2025), p. 106857. DOI: 10.1016/j.cor.2024.106857.
- [15] Matteo Fischetti, Michael Kahr, Markus Leitner, Michele Monaci, and Mario Ruthmair. “Least cost influence propagation in (social) networks”. In: *Mathematical Programming* 170 (2018), pp. 293–325. DOI: 10.1007/s10107-018-1288-y.
- [16] Tommaso Giovannelli, Giampaolo Liuzzi, Stefano Lucidi, and Francesco Rinaldi. “Derivative-free Methods for Mixed-integer Nonsmooth Constrained Optimization”. In: *Computational Optimization and Applications* 82 (2022), pp. 293–327. DOI: 10.1007/s10589-022-00363-1.
- [17] Sharon Goldberg and Zhenming Liu. “The Diffusion of Networking Technologies”. In: *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Ed. by Sanjeev Khanna. Philadelphia: Society for Industrial and Applied Mathematics, 2013, pp. 1577–1594. DOI: 10.1137/1.9781611973105.113.
- [18] Evren Güney, Markus Leitner, Mario Ruthmair, and Markus Sinnl. “Large-scale influence maximization via maximal covering location”. In: *European Journal of Operational Research* 289.1 (2021), pp. 144–164. DOI: 10.1016/j.ejor.2020.06.028.
- [19] Dilek Güneş, S. Raghavan, and Rui Zhang. “Least-cost influence maximization on social networks”. In: *INFORMS Journal on Computing* 32.2 (2020), pp. 289–302. DOI: 10.1287/ijoc.2019.0886.
- [20] 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*. New York: Association for Computing Machinery, 2003, pp. 137–146. DOI: 10.1145/956750.956769.
- [21] Jochen Könemann, Sina Sadeghian, and Laura Sanità. “Better Approximation Algorithms for Technology Diffusion”. In: *Algorithms-ESA 2013*. Ed. by Hans L. Bodlaender and Giuseppe F. Italiano. Vol. 8125. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2013, pp. 637–646. DOI: 10.1007/978-3-642-40450-4_54.
- [22] Jeffrey Larson, Sven Leyffer, Prashant Palkar, and Stefan M. Wild. “A Method for Convex Black-box Integer Global Optimization”. In: *Journal of Global Optimization* 80 (2021), pp. 439–477. DOI: 10.1007/s10898-020-00978-w.
- [23] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. “Cost-effective Outbreak Detection in Networks”. In: *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York: Association for Computing Machinery, 2007, pp. 420–429. DOI: 10.1145/1281192.1281239.
- [24] Yuchen Li, Ju Fan, Yanhao Wang, and Kian-Lee Tan. “Influence Maximization on Social Graphs: A Survey”. In: *IEEE Transactions on Knowledge and Data Engineering* 30.10 (2018), pp. 1852–1872. DOI: 10.1109/TKDE.2018.2807843.
- [25] Giampaolo Liuzzi, Stefano Lucidi, and Francesco Rinaldi. “An Algorithmic Framework Based on Primitive Directions and Nonmonotone Line Searches for Black-box Optimization Problems with Integer Variables”. In: *Mathematical Programming Computation* 12 (2020), pp. 673–702. DOI: 10.1007/s12532-020-00182-7.
- [26] Giampaolo Liuzzi, Stefano Lucidi, and Francesco Rinaldi. “Derivative-free Methods for Bound Constrained Mixed-integer Optimization”. In: *Computational Optimization and Applications* 53 (2012), pp. 505–526. DOI: 10.1007/s10589-011-9405-3.
- [27] Giampaolo Liuzzi, Stefano Lucidi, and Francesco Rinaldi. “Derivative-free Methods for Mixed-integer Constrained Optimization Problems”. In: *Journal of Optimization Theory and Applications* 164 (2015), pp. 933–965. DOI: 10.1007/s10957-014-0617-4.
- [28] Juliane Müller. “MISO: Mixed-integer Surrogate Optimization Framework”. In: *Optimization and Engineering* 17 (2016), pp. 177–203. DOI: 10.1007/s11081-015-9281-2.
- [29] Giacomo Nannicini, Giorgio Sartor, Emiliano Traversi, and Roberto Wolfler Calvo. “An exact algorithm for robust influence maximization”. In: *Mathematical Programming* 183.1 (2020), pp. 419–453. DOI: 10.1007/s10107-020-01507-z.
- [30] Eric Newby and M. M. Ali. “A Trust-region-based Derivative Free Algorithm for Mixed Integer Programming”. In: *Computational Optimization and Applications* 60 (2015), pp. 199–229. DOI: 10.1007/s10589-014-9660-1.
- [31] Nikolaos Ploskas and Nikolaos V. Sahinidis. “Review and Comparison of Algorithms and Software for Mixed-integer Derivative-free Optimization”. In: *Journal of Global Optimization* 82 (2022), pp. 433–462. DOI: 10.1007/s10898-021-01085-0.

- [32] Charupriya Sharma. “Technology Diffusion on Spiders”. In: *Master’s Thesis, University of Waterloo (Canada)* (2017).
- [33] Youze Tang, Yanchen Shi, and Xiaokui Xiao. “Influence Maximization in Near-Linear Time: A Martingale Approach”. In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. New York: Association for Computing Machinery, 2015, pp. 1539–1554. DOI: 10.1145/2723372.2723734.
- [34] Youze Tang, Xiaokui Xiao, and Yanchen Shi. “Influence Maximization: Near-Optimal Time Complexity Meets Practical Efficiency”. In: *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*. New York: Association for Computing Machinery, 2014, pp. 75–86. DOI: 10.1145/2588555.2593670.
- [35] Kübra Tanınmış, Necati Aras, and İ. K. Altınel. “Influence maximization with deactivation in social networks”. In: *European Journal of Operational Research* 278.1 (2019), pp. 105–119. DOI: 10.1016/j.ejor.2019.04.010.
- [36] Yu Tian and Renaud Lambiotte. “Unifying Information Propagation Models on Networks and Influence Maximization”. In: *Physical Review E* 106.3 (2022), p. 034316. DOI: 10.1103/PhysRevE.106.034316.
- [37] Hao-Hsiang Wu and Simge Küçükyavuz. “A two-stage stochastic programming approach for influence maximization in social networks”. In: *Computational Optimization and Applications* 69.3 (2018), pp. 563–595. DOI: 10.1007/s10589-017-9958-x.