Computational aspects of optimal strategic network diffusion

Marcin Waniek, Khaled Elbassioni, Flávio L Pinheiro, César A. Hidalgo, Aamena Alshamsi

Theoretical Computer Science 814 (2020) 153–168

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Computational aspects of optimal strategic network diffusion

Marcin Waniek [a,b,*], Khaled Elbassioni [b], Flávio L. Pinheiro [c,d], César A. Hidalgo [d], Aamena Alshamsi [b]

[a] Computer Science, New York University Abu Dhabi, Abu Dhabi, United Arab Emirates
[b] Khalifa University of Science and Technology, Abu Dhabi, United Arab Emirates
[c] Nova Information Management School (NOVA IMS), Universidade Nova de Lisboa, Lisboa, Portugal
[d] MIT Media Lab, Massachusetts Institute of Technology, Cambridge, USA

ARTICLE INFO

Article history:
Received 5 July 2019
Received in revised form 19 December 2019
Accepted 26 January 2020
Available online 30 January 2020
Communicated by L.M. Kirousis

Keywords:
Strategic diffusion
Network contagion
Complex networks
Influence maximization

ABSTRACT

Diffusion on complex networks is often modeled as a stochastic process. Yet, recent work on strategic diffusion emphasizes the decision power of agents [1] and treats diffusion as a strategic problem. Here we study the computational aspects of strategic diffusion, *i.e.*, finding the optimal sequence of nodes to activate a network in the minimum time. We prove that finding an optimal solution to this problem is NP-complete in a general case. To overcome this computational difficulty, we present an algorithm to compute an optimal solution based on a dynamic programming technique. We also show that the problem is fixed parameter-tractable when parametrized by the product of the treewidth and maximum degree. We analyze the possibility of developing an efficient approximation algorithm and show that two heuristic algorithms proposed so far cannot have better than a logarithmic approximation guarantee. Finally, we prove that the problem does not admit better than a logarithmic approximation, unless P=NP.

1. Introduction

Diffusion in social networks has been a widely studied application in problems such as epidemic outbreaks and the adoption of behaviors and innovations [2–7]. The approach to model diffusion can be roughly divided between simple and complex contagion processes. Under simple contagion transmission requires contact with one individual at a constant rate such as transmission of infectious diseases [8]. An example of simple contagion process is the independent cascade model [9]. In contrast, complex contagion requires reinforcement from multiple sources. An example of complex contagion process is the linear threshold model [9]. Complex contagion has been widely studied in the context of cascade phenomena, with particular applications to viral marketing campaigns [10–12]. In that context, the problem under analysis is that of finding the initial set of seeds that would maximize diffusion. Different approaches to solving the seed selection problem include using centrality measures [13], network percolation [14], and propagation traces [15], among many others [16–18]. Some works in the literature take an adaptive approach to the topic, basing the selection of seeds on additional knowledge gathered during the process, be it either the current state of a dynamic network topology [19], or expanded pool of potential seeds [20,21]. An alternative way of increasing the diffusion coverage is spreading available seeds over time [22,23]. Despite the strategic aspects of different methods of seed selection, the diffusion process itself is purely stochastic.

* Corresponding author at: Computer Science, New York University Abu Dhabi, Abu Dhabi, United Arab Emirates.
E-mail address: mjwaniek@nyu.edu (M. Waniek).

https://doi.org/10.1016/j.tcs.2020.01.027
0304-3975/

However, there are cases in which diffusion takes place according to the same rules of complex contagion, but that are not purely stochastic (*i.e.*, they have a strategic component) and in which the network itself is not, necessarily, a social network. Take for instance how regions develop new economic activities [24] or start production in new research fields [25]. In these cases agents are not embedded in the network, instead they are taking actions over a networked system, which captures the relatedness between economic or academic activities. More importantly, not only the choice of the initial state is strategic, but also the whole process of diffusion is strategic. Stochasticity in this context is not at the pairwise interactions between agents, but in the success rate of agents in entering new activities. In case of the economic development, the probability of success while entering a new activity (represented by a node in the network of related economic activities) increases with the number of other already developed activities that are connected to it. Thus, agents can fail or succeed depending on the particular context of each strategic action.

Alshamsi et al. [1] proposed to study the diversification of economic activities through the lenses of a strategic diffusion model. In this model the entire process of diffusion is guided by a strategic agent. The agent's actions concern the selection of a node in a network to be targeted at each time step. Hence, the node can become activated in the next step of the action sequence of the agent, or not. Following existing empirical findings [24,26], the model assumes that the activation probability (*i.e.*, the probability of success of the agent's action) is captured by a complex contagion process. The goal is to activate the entire network, starting from a single active node, in the minimum time possible. While Alshamsi et al. [1] showed for a wide range of network topologies that activating a network in an optimal manner requires a balance between exploitation and exploration strategies, many questions regarding the computational feasibility of finding optimal strategies remained open. Here, we explore several theoretical computational considerations of strategic diffusion processes.

1.1. Results and methods

We start by exploring the computational feasibility of finding an optimal strategy that minimizes the total diffusion time, *i.e.*, answering the question of what is the best sequence of activating a given number of nodes in the network (starting from a seed node) in the shortest expected time, while taking into consideration that the time necessary to activate each node is proportional to the number of active neighbors.

We show that the decision version of the problem is NP-complete (Theorem 3). We prove it using a reduction from the Set Cover problem. This observation indicates that developing a polynomial algorithm for finding an optimal sequence of strategic diffusion is not realistic, unless P=NP.

Given this difficulty, we propose an algorithm for computing an optimal way of activating a given number of nodes in the shortest time possible (Algorithm 1). While the algorithm takes exponential time to find the solution, it still outperforms simple exhaustive search and can be used to compute an optimal solution for networks of moderate size. The algorithm utilizes the dynamic programming technique to find the fastest way to reach every possible state of network activation.

We also present two algorithms finding an optimal solution for a more restrictive class of networks, *i.e.*, networks with bounded treewidth and bounded maximal degree. They traverse a tree decomposition of a network and find an optimal way of activating either an entire network or a given number of nodes, respectively. As a consequence, we show that the problem is fixed parameter-tractable when parametrized by the sum of the treewidth and maximum degree.

As finding an optimal solution proves to be computationally demanding, we turn our attention to assessing the possibility of obtaining an efficient approximation algorithm. We first investigate whether two effective heuristic algorithms proposed by Alshamsi et al. [1] have constant approximation ratios. One of them is the greedy algorithm, *i.e.*, always targeting the node with the highest probability of activation. The other is the majority algorithm, *i.e.*, targeting the node with the highest number of active neighbors. We show that the approximation ratio for both of these algorithms is $\Omega(\log n)$ (Theorems 7 and 8). The proofs are based on constructing a sequence of networks where the ratio between total expected time of activation of the solution obtained using the heuristic algorithm and the optimal expected time of activation goes to infinity.

Finally, we show that, unless P=NP, there is no way to approximate the problem within a ratio better than $\ln n$, in particular it is impossible to construct an r-approximation algorithm for a constant r (Theorem 9). We prove this claim by showing a reduction from the Minimum Set Cover problem and using the fact that Minimum Set Cover cannot be approximated within a ratio of $(1-\epsilon)\ln n$ for any $\epsilon > 0$, unless P=NP [27].

Organization of the manuscript. The remainder of the article is organized as follows. Section 2 describes the notations and computational problems used in our reductions. Section 3 presents a formal definition of the strategic diffusion process and the main problem considered in our study. In Section 4 we present our hardness result for the decision version of the problem and we describe a dynamic programming algorithm to find an optimal way of strategic diffusion. Section 4.4 introduces algorithms computing optimal solution for networks with bounded treewidth and maximal degree. Section 4.5 describes our results concerning approximation of the optimal solution. Section 5 presents conclusions and potential ideas for future work.

2. Preliminaries & notation

In this section, we present notations and concepts that will be used throughout the paper.

2.1. Basic network notation

Let $G = (V, E, W)$ denote a network with weighted edges, where $V = \{1, \ldots, n\}$ denotes the set of n nodes, $E \subseteq V \times V$ denotes the set of edges and $W \in \mathbb{R}^{n \times n}$ denotes the matrix with weights of edges. We denote an edge between nodes i and j by ij. In this work we consider networks that are *undirected*, *i.e.*, we do not discern between edges ij and ji. We also assume that networks do not contain self-loops, *i.e.*, $\forall_{i \in V}\, ii \notin E$. We denote by $N_G(i)$ the set of *neighbors* of i in G, *i.e.*, $N_G(i) = \{j \in V : ij \in E\}$. We denote by $d_G(i)$ the *degree* of i in G, *i.e.*, $d_G(i) = |N_G(i)|$.

We consider networks with weighted edges. We denote by $w_{ij} \in \mathbb{R}$ the weight of the connection from i to j, we will call this value *influence* that i has on j. We do not assume that the relation of influence is symmetric, *i.e.*, it is possible that for $ij \in E$ we have $w_{ij} \neq w_{ji}$. Unless stated otherwise, we will assume that $\forall_{i,j \in V}\, w_{ij} \geq 0$. We also assume that if $ij \notin E$ then $w_{ij} = 0$. We denote by w_i the sum of influence on node i, *i.e.*, $w_i = \sum_{j \in N(i)} w_{ji}$. We will typically assume that $\forall_{i \in V}\, w_i > 0$.

Let $\Gamma(V)$ denote the set of all *ordered sequences* of elements from V without repetitions. Let γ_i denote the i-th element (node) of sequence $\gamma \in \Gamma(V)$, and $|\gamma|$ denote the number of elements in $\gamma \in \Gamma(V)$. Finally, we call $\gamma \in \Gamma(V)$ a *full* sequence if $|\gamma| = |V|$ and we denote the set of full sequences by $\Gamma^*(V)$.

To make the notation more readable, we will often omit the network itself from the notation when it is clear from the context, *e.g.*, by writing $N(i)$ instead of $N_G(i)$. We sometimes treat sequences as sets, when the order is not important. We use $\oplus$ to denote the concatenation operation over sequences.

2.2. Computational problems

In our reductions we will use two standard versions of the Set Cover problem, decision and combinatorial optimization.

Definition 1 *(Set Cover [28]).* An instance of the Set Cover problem is defined by a universe $U = \{u_1, \ldots, u_{|U|}\}$, a collection of sets $\mathcal{S} = \{S_1, \ldots, S_{|\mathcal{S}|}\}$ such that $\forall_j S_j \subset U$, and an integer $k \leq |\mathcal{S}|$. The goal is to determine whether there exist k elements of $\mathcal{S}$ the union of which equals U.

Set Cover is one of the classic 21 Karp's NP-complete problems.

Theorem 1 *([28]). Set Cover problem is NP-complete.*

For the proof of the approximation hardness we will use the minimization version of the problem.

Definition 2 *(Minimum Set Cover).* An instance of the Minimum Set Cover problem is defined by a universe $U = \{u_1, \ldots, u_{|U|}\}$ and a collection of sets $S = \{S_1, \ldots, S_{|S|}\}$ such that $\forall_j S_j \subset U$. The goal is to find subset $S^* \subseteq S$ such that the union of S^* equals U and the size of S^* is minimal.

For $\alpha \geq 1$, an α-approximation for a given instance of a minimization problem is a feasible solution whose objective is within a factor of α of any optimal solution. We will use the fact that Minimum Set Cover problem is hard to approximate.

Theorem 2 *([27]). For any fixed $\epsilon > 0$ Minimum Set Cover cannot be approximated to within $(1 - \epsilon) \ln n$, unless P=NP.*

We now move to defining the model of strategic diffusion and the main optimization problem of our study.

3. Problem definition

In this section we describe the process of strategic network diffusion, which is the main focus of our study, as well as the computational problem concerning it.

Most diffusion models are purely stochastic [10,9]. In this work however, we focus on the strategic model of diffusion to account for cases in which the interconnections between targets affect their activation time and therefore choosing the order in which nodes will be targeted for activation is strategically planned.

In this model, the process of diffusion is driven by a strategic agent. At the beginning of the process only one chosen node of the network, the *seed node* i_S, is active. Then, the agent chooses a sequence $\gamma \in \Gamma(V)$ that provides the order in which the nodes will be activated. The probability of successful activation of a node i in one attempt is given by:

$$p(i) = \beta \left(\frac{\sum_{j \in N(i) \cap A} w_{ji}}{w_i} \right)^{\alpha}$$

where A is the set of currently active nodes (at the beginning of the process it consists only of the seed), and $\alpha, \beta \in [0, 1]$ are constants. Unless stated otherwise, we will assume that $\alpha = \beta = 1$. Notice that the expected time of activation of node

i with non-zero activation probability is $\tau(i) = \frac{1}{p(i)}$. By $\tau(\gamma)$ we will denote the expected time of activation of all nodes in the sequence γ.

This model was proposed by Alshamsi et al. [1] for undirected networks with unweighted edges. If we assume that $w_{ij} = 1 \iff ij \in E$ then our model is exactly equivalent to the model proposed by Alshamsi et al. [1].

We now define the main computational problem of our study.

Definition 3 *(Optimal Partial Diffusion Sequence).* This problem is defined by a tuple (G, i_S, z), where $G = (V, E, W)$ is a given network with weighted edges, $i_S \in V$ is the seed node and $z \leq n$ in the number of nodes to activate. The goal is to identify $\gamma^* \in \Gamma(V)$ such that $\gamma^*_1 = i_S$, $|\gamma^*| = z$ and $\tau(\gamma^*)$ is minimal.

In other words, we intend to find the fastest way to activate z nodes in the network. When $z = |V|$ in the above definition, the problem is simply called Optimal (Full) Diffusion Sequence.

Remark 1. In the case of integer weights of *polynomial length*, for any instance of the Optimal Diffusion Sequence problem there exists a polynomially equivalent instance with binary weights.

Proof. Let $(G, i_S, |V|)$ be a given instance of the Optimal Diffusion Sequence. In order to construct an equivalent instance with 0/1-weights we replace every edge ij with a set of w_{ij} parallel 2-paths $\{\langle i, k_{i,j,1}, j\rangle, \ldots, \langle i, k_{i,j,w_{ij}}, j\rangle\}$, and we set weights of the new edges to $w_{ik_{i,j,r}} = w_{k_{i,j,r}j} = 1$, $w_{k_{i,j,r}i} = w_{jk_{i,j,r}} = 0$, for $r = 1, \ldots, w_{ij}$.

Given a feasible solution γ to the original instance, we obtain a feasible solution γ' to the constructed instance by activating all nodes $k_{i,j,r}$ immediately after activating node i in the original sequence (notice that the expected time of activation of every such node $k_{i,j,r}$ is 1). Thus, if the value of the solution γ is t, then the value of the γ' is $t + \sum_{ij \in E} \left(w_{ij} + w_{ji}\right)$.

Given a feasible solution γ' to the constructed instance of the binary version of the problem, we can obtain a solution with a lesser or equal value by activating all nodes $k_{i,j,r}$ in one block immediately after activating node i (again, the expected time of activation of node $k_{i,j,r}$ is always 1, while it contributes to the time of activation of node j). For such sequence, we can obtain corresponding solution γ to the original instance of the problem by removing from it all nodes $k_{i,j,r}$. If the value of the solution γ' is t, then the value of the γ is $t - \sum_{ij \in E} \left(w_{ij} + w_{ji}\right)$.

Note that this reduction does not work in general for the Optimal Partial Diffusion Sequence if $z < |V|$. □

4. Computing an optimal solution

We now describe our results on finding an efficient way to compute an exact solution to the Optimal Diffusion Sequence problem.

4.1. Hardness of finding an optimal solution

First, we show that the decision version of Optimal Partial Diffusion Sequence problem is NP-complete in a general case.

Theorem 3. *Optimal Partial Diffusion Sequence problem is NP-complete, even if all weights are in* $\{0, 1\}$.

Proof. The decision version of the Optimal Partial Diffusion Sequence problem is the following: given a network $G = (V, E, W)$, the seed node i_S, the number of nodes to activate z, and a value $t^* \in \mathbb{R}^+$, does there exist a sequence of nodes $\gamma^* \in \Gamma(V)$ starting with i_S such that $|\gamma^*| = z$ and $\tau(\gamma^*) \leq t^*$?

This problem clearly is in NP, as given a solution, *i.e.*, a sequence of nodes $\gamma^* \in \Gamma(V)$, we can compute its expected time of activation in polynomial time.

To prove the NP-hardness of the problem we will show a reduction from the NP-complete Set Cover problem.

Let $(U, \mathcal{S}, k)$ be an instance of the Set Cover problem. We define network G as follows (an example of such network is presented in Fig. 1):

- **The set of nodes:** For every $S_i \in \mathcal{S}$, we create three nodes, denoted by S_i, q_i and q'_i. For every $u_i \in U$, we create a single node u_i. Additionally, we create a single node i_S.
- **The set of edges:** For every node S_i, we create an edge $S_i i_S$. For every node q_i, we create edges $q_i S_i$ and $q_i q'_i$. Finally, for every node u_i and every S_j such that $u_i \in S_j$ we create an edge $u_i S_j$.
- **The weight matrix:** For every edge $u_i S_j$ we set weights to $w_{u_i S_j} = 0$ and $w_{S_j u_i} = 1$. For every edge $S_i q_i$ we set weights to $w_{S_i q_i} = 0$ and $w_{q_i S_i} = |U||\mathcal{S}|$. For all other pairs of nodes connected with an edge we set the weights to 1.

Let $z = k + |U| + 1$ (we intend that the solution sequence will activate node i_S, k nodes in $\mathcal{S}$ and all nodes in U), and a value $t^* = k(|U||\mathcal{S}| + 1) + |U||\mathcal{S}|$. Now, consider an instance of the Optimal Partial Diffusion Sequence problem (G, i_S, z).

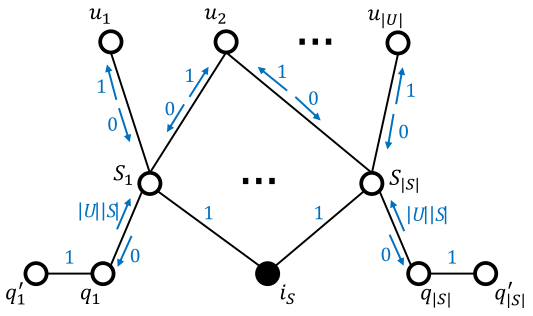

Fig. 1. Network G constructed for the proof of Theorem 3. Blue numbers next to edges express their weights. If there are no arrows next to edge ij then $w_{ij} = w_{ji}$. Otherwise weight w_{ij} is denoted next to the arrow pointing towards node j, and weight w_{ji} is denoted next to the arrow pointing towards node i. (For interpretation of the colors in the figures, the reader is referred to the web version of this article.)

We will now show that a solution to this instance of value at most t^* corresponds to a solution to the given instance of the Set Cover problem.

Notice that the expected time of activation of every node S_i is always $|U||\mathcal{S}|+1$, while for the expected time of activation of a node u_i we have $\tau(u_i) \leq |\{S_j : u_i \in S_j\}| \leq |\mathcal{S}|$. Notice also that neither any of the nodes q_i nor any of the nodes q'_i can be activated when i_S is the seed node, as the influence of S_i on q_i is zero.

First, we will show that if there exists a solution $\mathcal{S}^*$ to the given instance of the Set Cover problem, then there also exists a solution to the constructed instance of the Optimal Partial Diffusion Sequence problem of value at most t^*. We can construct such a solution by first activating every node $S_i \in \mathcal{S}^*$ (k nodes activated in expected time $|U||\mathcal{S}| + 1$ each) and then activating all nodes u_i ($|U|$ nodes activated in expected time not exceeding $|\mathcal{S}|$ each). Such γ^* activates $k + |U|$ nodes in expected time not exceeding $k(|U||\mathcal{S}| + 1) + |U||\mathcal{S}|$, therefore it is a solution to the constructed instance of the Optimal Partial Diffusion Sequence problem of value at most t^*.

Second, to complete the proof of the NP-hardness, we have to show that if there exists a solution γ^* to the constructed instance of the Optimal Partial Diffusion Sequence problem of value at most t^*, then there also exists a solution to the given instance of the Set Cover problem. Such a solution is $\mathcal{S}^* = \gamma^* \cap \mathcal{S}$, *i.e.*, choosing sets S_i corresponding to nodes S_i occurring in sequence γ^*. Notice that there cannot be more than k such nodes, as activating $k + 1$ nodes S_i has expected time $(k + 1)(|U||\mathcal{S}| + 1) > t^*$. Since this is the case, in order to activate $k + |U|$ nodes other than i_S, sequence γ^* has to activate all nodes in U. However, to activate a node u_i, we first have to activate at least one of its neighbors, *i.e.*, node S_j such that $u_i \in S_j$. Therefore, for every node u_i there must exist at least one node $S_j \in \gamma^* \cap S$ such that $u_i \in S_j$. Hence, $S^* = \gamma^* \cap \mathcal{S}$ is a valid solution to the given instance of the Set Cover problem.

Finally, we can use the construction in Remark 1 to replace every edge $q_i S_i$ by a set of parallel paths with edge weights in $\{0, 1\}$. Note that such a replacement does not change the value of the objective as the nodes q_i, and hence the intermediate nodes added on the parallel paths, are never activated. This concludes the proof. □

Therefore, there exists no polynomial algorithm finding the optimal way to activate a given number of nodes in the process of strategic diffusion, unless P=NP. However, we now propose an exponential algorithm based on dynamic programming technique.

4.2. Dynamic programming algorithm

We present an algorithm for computing an optimal solution to the problem using the dynamic programming technique [29]. The main idea behind dynamic programming is breaking the main problem into a number of smaller, easier to solve sub-problems in a recursive manner. However, unlike in standard recursion where often the same computation is repeated multiple times, in dynamic programming the solution to each sub-problem is computed only once and stored in memory for future use. Over the years the dynamic programming techniques were developed in various directions [30–33], however our algorithm is based on the original version of the technique.

Notice that in our setting the activation probability of any given node depends on the set of active nodes in the network, however it does not depend on the order in which these nodes were activated. Hence, the solution for each set of active nodes have to be computed only once. Moreover, solving the sub-problem of finding an optimal way of activating k nodes in the network allows to efficiently find the solution for $k + 1$ nodes, as the total expected time of activation can be expressed in a recursive manner as a sum of time of activation of the initial k nodes and the time of activation of the last node. The idea of using solutions for smaller sub-problems to solve a larger problem is the core concept behind the dynamic programming, hence we find it a suitable optimization method for solving the Optimal Partial Diffusion Sequence problem.

The algorithm is based on the following recurrence relation, where $\tau^*(C)$ is the minimal expected time of activation of a set of nodes C:

Algorithm 1: Dynamic programming algorithm for strategic diffusion.

Input: A weighted network (V, E, W), a seed node $i_S \in V$, and the number of nodes to activate z.
Output: Sequence of activation of z nodes starting with i_S with minimal expected time of activation.

```
1  for C ⊆ V do
2      τ*[C] ← ∞
3  τ*[{i_S}] ← 0
4  γ*[{i_S}] ← ⟨i_S⟩
5  for k = 1, ..., z − 1 do
6      for C ⊂ V : (|C| = k) ∧ (τ*[C] < ∞) do
7          for i ∈ V : (i ∉ C) ∧ (N(i) ∩ C ≠ ∅) do
8              Δτ ← w_i / Σ_{j∈N(i)∩C} w_ji
9              if τ*[C] + Δτ < τ*[C ∪ {i}] then
10                 τ*[C ∪ {i}] ← τ*[C] + Δτ
11                 γ*[C ∪ {i}] ← γ*[C] ⊕ ⟨i⟩
12 return γ*[arg min_{C⊆V:|C|=z} τ*[C]]
```

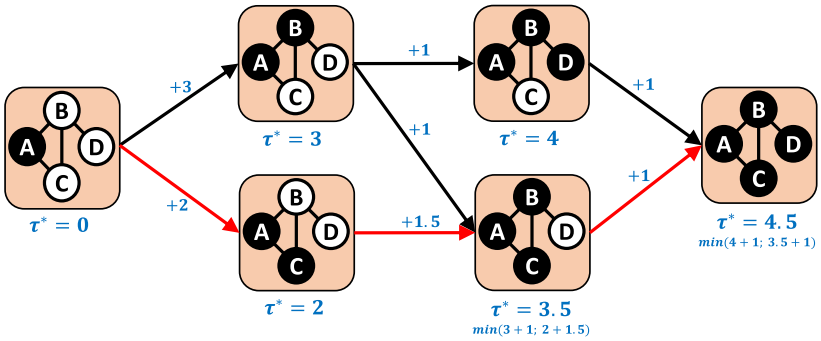

Fig. 2. Example of using dynamic programming. Each large orange state represents a possible state of activation of the network, with black nodes representing active nodes, and white nodes representing inactive nodes. The value of τ^* indicates the minimal expected time required to reach this state of activation. Value on each arrow represents the cost of activation of a single node, required to move between states. Red arrows represent optimal sequence of activation, which corresponds to the least expensive path from the initial state to the state where entire network is activated.

$$\tau^*(C) = \begin{cases} 0 & \text{if } C = \{i_S\} \\ \min\limits_{i \in C} \tau^*(C \setminus \{i\}) + \frac{w_i}{\sum_{j \in N(i) \cap C} w_{ji}} & \text{if } |C| > 1 \\ \infty & \text{otherwise} \end{cases}$$

where we assume that summation over empty set results in zero and division by zero results in ∞.

Out of sets of size one, only the set consisting of the seed node has finite time of activation. For sets that are larger than one we divide the problem into computing the optimal time of activation of all nodes but the last one, and then computing the time of activation of the last node. Since in the strategic diffusion process the nodes are activated sequentially, one of the nodes i from the set C has to be activated last, and we are able to compute its time of activation based on the information of which other nodes of the network are already active. Notice that attempting to activate a node without any active neighbors (or with sum of influences from these neighbors equal to zero) will result in the value of the formula equal to ∞.

Pseudocode of the dynamic programming algorithm is presented as Algorithm 1, while Fig. 2 presents the intuition behind the algorithm by showing how it works on a specific graph structure.

In entry $\tau^*[C]$ we compute the minimal expected time necessary to activate all nodes in set C, while in entry $\gamma^*[C]$ we keep a sequence of activation allowing us to achieve this optimal expected time. In the k-th execution of the loop in line 5 we compute values of entries in τ^* and γ^* for sets C such that $|C| = k + 1$. We do it by iterating (in loop in line 6) over all sets of nodes of size k that can be activated when we start the process from the seed node i_S and then iterate (in loop in line 7) over all possible nodes that can be targeted, *i.e.*, nodes with non-zero probability of activation, when the set of active nodes is C. In lines 8-11 we update the best way of activating nodes in set $C \cup \{i\}$ when activating nodes in C first and activating node i afterwards has lower expected time than currently known fastest way to activate nodes in $C \cup \{i\}$.

As for the implementation details, τ^* and γ^* can be implemented as hash tables. In this case checking whether $\tau^*[C] < \infty$ is equivalent to checking whether τ^* contains the entry for C and therefore lines 1-2 can be omitted.

Notice also that during the k-th execution of the loop in line 5 we only need entries in tables τ^* and γ^* for sets C such that $|C| = k$. Hence, to reduce memory requirements, we can only keep in memory entries from tables τ^* and γ^* for two sizes of sets: the ones for size k, computed in the previous execution of the loop (or initialized in lines 3-4 in case of the first execution) and the ones for size $k + 1$, being computed in the current execution of the loop.

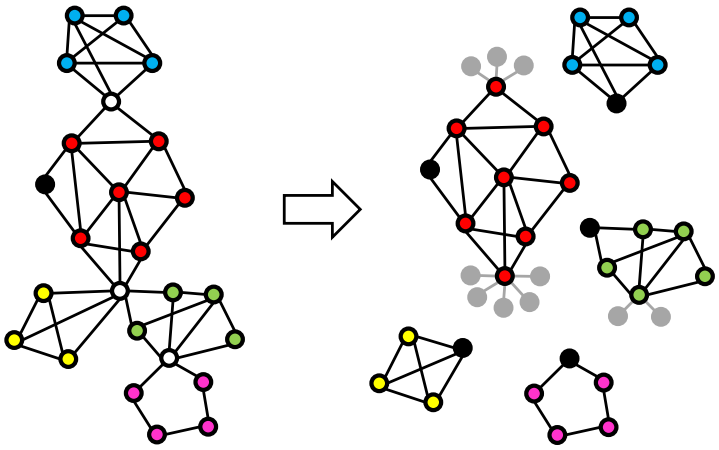

Fig. 3. Decomposition of graph into biconnected components. Each biconnected component is marked with different color, with cut nodes marked white and seed nodes marked black. Gray nodes mark stubs added to provide proper influence sum for original cut nodes.

Despite these optimization possibilities, the dynamic programming algorithm remains exponential, as it considers all subsets of nodes possible to be activated. If the number of nodes to be activated is constant then the algorithm is polynomial.

Theorem 4. *Algorithm 1 solves the Optimal Partial Diffusion Sequence problem in time* $\mathcal{O}(|V|^{z+1}|E|)$.

In Section 4.4, we give a dynamic program that works more efficiently when the network has both bounded degree and bounded treewidth. Note that it is common to use dynamic programming for solving optimization problems on graphs with bounded treewidth, see, *e.g.*, [34].

An alternative approach to developing an algorithm of finding an effective way of activating the network would be to use the reinforcement learning techniques [35]. Reinforcement learning is based on the idea of making a sequence of decisions by finding a balance between exploitation (performing the currently best action to obtain short-term profits) and exploration (investigating other actions in the hope of a long-term gain). In case of strategic diffusion, the exploitation would be activating a node with the highest activation probability, while the exploration would be activating a node with lower activation probability in order to make its neighbors easier to activate. Nevertheless, an algorithm based on reinforcement learning would not guarantee that identified activation sequence is the optimal solution, unlike the presented dynamic programming algorithm, which offers such certainty. Hence, we leave the development of an algorithm based on reinforcement learning as a potential future work.

We now discuss other ways of optimization for more restricted network structures.

4.3. Decomposition into biconnected components

We will now show that looking for the optimal way of activating all nodes of the network can be made simpler by using decomposition into biconnected components.

Cut node in a network (also called an articulation point) is a node the removal of which causes network to fall into two or more connected components. By separating the network in cut nodes we obtain a decomposition into biconnected components. Biconnected component is a maximal sub network (in terms of inclusion) such that removing any node from it will not disconnect it. An example of a decomposition into biconnected components is shown in Fig. 3. Notice that a copy of a cut node appears in every biconnected component it belongs to.

The strategic diffusion process in every biconnected component can be considered separately. This is because every biconnected components has only one possible starting point of the diffusion (being it either the seed node in case of biconnected component containing it or the cut node closest to the seed node in case of all other components). This is indicated by black color of a node in Fig. 3. Once this starting point of a given component C is activated, the activation state in other biconnected components of the network does not affect activation is C.

This is because activation probability of a node is only affected by the state of activation of its neighbors and the value of w_i (to remind the reader, $w_i = \sum_{j \in N(i)} w_{ji}$). Only cut nodes have neighbors in other biconnected components, hence for all non-cut nodes this observation is trivial. As for any cut node i, notice that its neighbors in other biconnected components can only be activated after i is activated (as all the paths between them and the seed node run through the cut node). Therefore the only way in which neighbors of a cut node i in other biconnected components affect the activation probability of i is adding to the value of w_i. To account for this fact, we create additional stubs connected to copies of i that are not starting points of activation during the biconnected component decomposition (marked gray on Fig. 3).

To remind the reader, in this work we consider undirected networks (notice that the definition of biconnected components is different for directed networks). However, we do not assume that the relation of influence is symmetric, *i.e.*, it is possible that for $ij \in E$ we have $w_{ij} \neq w_{ji}$. Since we consider the decomposition into disconnected components in the context of a given seed node, there is never a disambiguation in terms of which edge weights have to be used in case of the cut nodes. To compute the activation probability of a cut node i, for each neighbor j only weight w_{ji} is taken into consideration. What is more, only neighbors of i that are closer to the seed node than i can have a positive contribution into its activation probability, as the neighbors of i that are further away from the seed node can only be activated after i is activated. At the same time, i affects the activation time of its neighbor j through the weight w_{ij}, and it can have a positive contribution both when j is closer and when its further away to the source node than i.

4.4. Optimal solution for networks with bounded treewidth

We will now show a polynomial algorithm that finds an optimal way to activate a network with both treewidth and degree bounded by constants.

Let $G = (V, E, W)$ be an arbitrary network. Let (T, F) be a network where every node contains a subset of nodes from V, *i.e.*, $\forall_{t \in T} t \subseteq V$ (we will call each such subset a *bag*). Such (T, F) is a tree decomposition of G (see, *e.g.*, [36]) if and only if the following conditions are met:

- Every node from V is contained in at least one bag, *i.e.*, $\bigcup_{t \in T} t = V$.
- For every edge $e \in E$ there exists a bag that contains both ends of e, *i.e.*, $\forall_{ij \in E} \exists_{t \in T} \{i, j\} \subseteq T$.
- For every node $v \in V$ subnetwork of (T, F) induced by bags containing v is connected.

The treewidth θ of a decomposition is the size of its largest bag minus one, *i.e.*, $\theta = \max_{t \in T} |t| - 1$. The treewidth of a network is the minimum over treewidths of all its tree decompositions. A problem is said to be *fixed-parameter tractable* [37], with respect to parameter k, if any instance of the problem of size N can be solved in time $f(k) \cdot N^{O(1)}$, for some computable function $f(\cdot)$. We show that the Optimal Strategic Diffusion problem is fixed-parameter tractable with respect to the sum of treewidth and maximum degree.

4.4.1. Full diffusion

To better explain the idea, we first give an algorithm for the full diffusion case.

Theorem 5. *Let $G = (V, E, W)$ be a network with maximal degree δ. There exists an algorithm that, given a tree decomposition of treewidth θ, finds an optimal way of activating nodes in V in time $\mathcal{O}(\theta\delta(\theta\delta)!^2 n)$.*

In what follows, we assume that the tree decomposition is rooted in some node t_R, the same for bottom-up and top-down order. We use the following notation:

- $c(t)$ denotes the sequence of children of t;
- $\gamma|_X$ denotes the subsequence of γ consisting only of the nodes in set X;
- $\gamma_{\triangleleft x}$ denotes the subsequence of γ consisting of all elements preceding x;
- $\gamma_{\blacktriangleright x}$ denotes the subsequence of γ consisting of x as well as all elements following x;
- $T_{topdown}$ denotes sequence of nodes in T in a top-down order.

In what follows we will call two permutations γ and γ' of nodes from G *compatible*, denoted $\gamma \sim \gamma'$, if and only if they activate the nodes that they have in common in the same order, *i.e.*, $\gamma|_{\gamma'} = \gamma'|_{\gamma}$.

For each node of t the tree decomposition T we compute a record consisting of the following fields:

- $\Upsilon[t]$ stores all permutations of nodes in bag t and their neighbors that can be a part of a valid solution to the problem;
- $\tau[t, \gamma, i]$ stores the expected time of activation of every node in $i \in t$ when activated in the order given by γ;
- $\tau^*[t, \gamma]$ stores the smallest expected time to activate all nodes in the subnetwork of G induced by the nodes in the subtree of T rooted at t, when the nodes in t and their neighbors are activated in the order given by γ.

To compute the records we traverse the tree decomposition in a bottom-up order (given the root t_R) and for every node t we perform the following steps:

1. Compute:

$$\Upsilon[t] = \left\{ \gamma \in \Gamma^*(t \cup \bigcup_{i \in t} N(i)) : (i_S \notin t \vee \gamma_1 = i_S) \wedge \left(\forall_{\substack{\gamma_i \in t: \exists_{j<i} \gamma_j \in N(\gamma_i) \\ \gamma_i \neq i_S}} \right) \wedge \left(\forall_{t' \in c(t)} \exists_{\gamma' \in \Upsilon[t']} \gamma' \sim \gamma \right) \right\};$$

Algorithm 2: Algorithm reconstructing the optimal way of activating all nodes of a network with bounded treewidth and degree.

Input: Tree decomposition (T, F) of a weighted network (V, E, W), with computed values of τ^*.
Output: Sequence of activation of nodes in V starting with i_S with minimal expected time of activation.

```
1  γ* ← ⟨⟩
2  for t ∈ T_topdown do
3      γ ← arg min_{γ'∈Υ[t]:γ'∼γ*} τ*[t, γ']
4      γ' ← ⟨⟩
5      for γ_i ∈ γ do
6          if γ_i ∉ γ* then
7              γ' ← γ' ⊕ ⟨γ_i⟩
8          else
9              γ* ← γ*_{◁γ_i*} ⊕ γ' ⊕ γ*_{▶γ_i*}
10             γ' ← ⟨⟩
11     γ* ← γ* ⊕ γ'
12 return γ*
```

2. For every $\gamma \in \Upsilon[t]$ and every $\gamma_i \in \gamma|_t$ compute:

$$\tau[t, \gamma, \gamma_i] = \frac{w_{\gamma_i}}{\sum_{j<i} w_{\gamma_j \gamma_i}};$$

3. For every $\gamma \in \Upsilon[t]$ compute:

$$\tau^*[t, \gamma] = \sum_{\gamma_i \in \gamma|_t} \tau[t, \gamma, \gamma_i] + \sum_{t' \in c(t)} \min_{\gamma' \in \Upsilon[t']:\gamma' \sim \gamma} \left(\tau^*[t', \gamma'] - \sum_{\gamma_i' \in \gamma'|_t} \tau[t', \gamma', \gamma_i'] \right).$$

After traversing the tree in this fashion, the minimal time required to activated entire network G starting with i_S can be identified as $\min_{\gamma \in \Upsilon[t_R]} \tau^*[t_R, \gamma]$. In order to reconstruct the sequence allowing to activate entire network in the optimal time, we can run Algorithm 2.

Let us now comment on the procedure of filling the records.

In step 1 we gather all permutations of nodes in bag t and their neighbors that can be a part of a valid solution to the problem, according to three conditions. The first condition asserts that if the permutation contains the seed node, it is activated as the very first node. The second condition assures that every node in bag t other than the source node has at least one active neighbor at the moment of activation (notice that for every node in t all of its neighbors are present in the permutation). The third condition provides that the permutation γ allows to activate all nodes in the subtree of t, *i.e.*, that for every child of t in the tree decomposition there exists at least one valid permutation γ' that activates nodes from γ in the same order as γ.

In step 2 we simply compute the time of activation of every node in bag t, according to the definition given in Section 3, and store it in table τ. Again, notice that for every node in t all of its neighbors are present in the permutation, hence we have enough information to compute its expected time of activation.

Finally, in step 3 we compute the optimal time of activation of all nodes in the subtree of t while using permutation γ and store it in table τ^*. The expression consists of a sum of time of activation of the nodes in bag t and a sum over all children of t, where for each of them we compute the minimum over all compatible permutations and subtract the time of activation of nodes in t. Notice that since (T, F) is the tree decomposition, the only possible overlap between nodes in the subtrees of different children of t are nodes in bag t. Otherwise the graph induced by bags containing such overlapping nodes would not be connected, and hence one of the conditions of being the tree decomposition would not be met for (T, F).

After having filled tables Υ, τ^* and τ, we traverse the tree decomposition again, this time in a top-down order using Algorithm 2. We construct an optimal solution to the problem on variable γ^*. In line 3 we select as γ the permutation with minimal time necessary to activate all nodes in the subtree of t, that is compatible with the solution constructed so far. In lines 4-11 we merge γ into sequence γ^* using auxiliary variable γ'.

As for the time complexity of the algorithm, the most costly operations (both of which are equally expensive) are validating existence of a compatible sequence for every child of t in the third condition in step 1 and computing the time necessary to activate the nodes in the subtrees of all children of t in the second sum in step 3. For assessing the time complexity of the algorithm we will focus on the cost in step 3. Since every node t' is a child of at most one other node in the tree decomposition, for every $t' \in T$ the computation of minimum is executed $\mathcal{O}((\theta\delta)!)$ times (the number of different permutations $\gamma \in \Upsilon[t]$). Since there are $\mathcal{O}(n)$ nodes in the tree decomposition, the computation of minimum is executed $\mathcal{O}((\theta\delta)!n)$ times. The cost of executing it once is $\mathcal{O}(\theta\delta(\theta\delta)!)$, since we have to check $\mathcal{O}((\theta\delta)!)$ many permutations in $\Upsilon[t']$ and for each of them check whether $\gamma'|_\gamma = \gamma|_{\gamma'}$ in time $\mathcal{O}(\theta\delta)$. Hence, the total time complexity of the algorithm is $\mathcal{O}(\theta\delta(\theta\delta)!^2 n)$.

Algorithm 3: Algorithm computing the value of $\tau^*[t, \gamma, k]$.

Input: Tree decomposition (T, F) of a weighted network (V, E, W), with values of τ^* computed so far, node t with sequence of children $c(t) = \langle t_1, \ldots, t_{|c(t)|} \rangle$.
Output: The value of $\tau^*[t, \gamma, k]$.

for $t_i \in \langle t_1, \ldots, t_{|c(t)|} \rangle$ **do**
  for $m = 0, \ldots, k - |\gamma|_t|$ **do**
    if $i = 1$ **then**
      $\tau'[t, 1, \gamma, m] \leftarrow \min_{\substack{\gamma' \in \Upsilon[t_1]: \\ \gamma' \sim \gamma}} \left(\tau^*[t_1, \gamma', m + |\gamma'|_t|] - \sum_{\gamma'_i \in \gamma'|_t} \tau[t_1, \gamma', \gamma'_i] \right)$
    else
      $\tau'[t, i, \gamma, m] \leftarrow \min_{m' \in \{0, \ldots, m\}} \left(\tau'[t, i-1, \gamma, m'] + \min_{\substack{\gamma' \in \Upsilon[t_i]: \\ \gamma' \sim \gamma}} \left(\tau^*[t_i, \gamma', m - m' + |\gamma'|_t|] - \sum_{\gamma'_i \in \gamma'|_t} \tau[t_i, \gamma', \gamma'_i] \right) \right)$
return $\sum_{\gamma_1 \in \gamma|_t} \tau[t, \gamma, \gamma_1] + \tau'\left[t, |c(t)|, \gamma, k - |\gamma|_t|\right]$

4.4.2. Partial diffusion

We next present an algorithm for partial diffusion in networks with bounded treewidth and bounded degree.

Theorem 6. *Let $G = (V, E, W)$ be a network with maximal degree δ. There exists an algorithm that, given a tree decomposition of treewidth θ, finds an optimal way of activating z nodes in V in time $\mathcal{O}(\theta\delta(\theta\delta)!^2 z^2 n)$.*

We use essentially the same notation as in previous section. However, in what follows we will call two permutations $\gamma \in \Gamma(X)$ and $\gamma' \in \Gamma(X')$ of nodes from G *compatible*, denoted $\gamma \sim \gamma'$, if and only if they activate the nodes that they have in common in the same order, *i.e.*, $\gamma|_{\gamma'} = \gamma'|_{\gamma}$, and agree on the non-activated nodes, *i.e.*, $\gamma|_{X' \setminus \gamma'} = \gamma'|_{X \setminus \gamma} = \langle\rangle$ (this difference is the result of considering also permutations that are not full, *i.e.*, permutations $\gamma \in \Gamma(X)$ such that $\gamma < |X|$).

The record that we compute for each node of t the tree decomposition T consisting now of the following fields:

- $\Upsilon[t]$ stores all permutations of nodes in bag t and their neighbors that can be a part of a valid solution to the problem;
- $\tau[t, \gamma, i]$ stores the expected time of activation of every node in $i \in t$ when activated in the order given by γ;
- $\tau^*[t, \gamma, k]$ denotes the smallest expected time to activate k nodes in the subnetwork of G induced by the nodes in the subtree of T rooted at t, when the nodes in t and their neighbors are activated in the order given by γ.

Hence, the only difference in comparison to the full diffusion version is that table τ^* is additionally indexed with the number of nodes to activate.

To compute the records we traverse the tree decomposition in a bottom-up order (given the root t_R) and for every node t we perform the following steps:

1. Compute:
$$\Upsilon[t] = \left\{ \gamma \in \Gamma(t \cup \bigcup_{i \in t} N(i)) : (i_S \notin t \vee \gamma_1 = i_S) \wedge \left(\forall_{\substack{\gamma_i \in t: \\ \gamma_i \neq i_S}} \exists_{j<i} \gamma_j \in N(\gamma_i) \right) \wedge \left(\forall_{t' \in c(t)} \exists_{\gamma' \in \Upsilon[t']} \gamma' \sim \gamma \right) \right\};$$
2. For every $\gamma \in \Upsilon[t]$ and every $\gamma_i \in \gamma|_t$ compute:
$$\tau[t, \gamma, \gamma_i] = \frac{w_{\gamma_i}}{\sum_{j<i} w_{\gamma_j \gamma_i}};$$
3. For every $\gamma \in \Upsilon[t]$ and for $k = |\gamma|_t|, \ldots, z$ compute the value of $\tau^*[t, \gamma, k]$ using Algorithm 3.

After traversing the tree in this fashion, the minimal time required to activate z nodes in network G starting with i_S can be identified as $\min_{\gamma \in \Upsilon[t_R]} \tau^*[t_R, \gamma, z]$. In order to reconstruct the sequence allowing to activate z nodes in the optimal time, we can run Algorithm 4.

Let us now comment on the procedure of filling the records.

Steps 1 and 2 are almost identical to those of the algorithm for full diffusion, with notable difference being that this time we consider sequences that are not necessarily full.

In step 3 we call Algorithm 3 to compute the value of $\tau^*[t, \gamma, k]$, *i.e.*, the optimal time of activating k nodes in the subtree of t while using permutation γ. To this end, we visit all the children of t, in the order $t_1, \ldots, t_{|c(t)|}$. For $m \in \{0, \ldots, k - |\gamma|_t|\}$ we compute $\tau'[t, i, \gamma, m]$, the minimum expected time required to activate m nodes *other than the ones activated in* t, in the subnetwork induced by union of the subtrees rooted at $t_1, \ldots, t_i$. We do this using the values of τ' computed for children of t preceding t_i, *i.e.*, for $t_1, \ldots, t_{i-1}$, as well as the values of τ^* computed for t_i. In the selection process m' denotes the number of nodes activated in the subnetwork induced by union of $t_1, \ldots, t_{i-1}$, while the rest of the m nodes is activated

Algorithm 4: ReconstructPartial(t, γ^*, k): algorithm reconstructing the optimal way of activating k nodes in a network with bounded treewidth and degree.

Input: Tree decomposition (T, F) of a weighted network (V, E, W), with computed values of τ^* and τ', a bag t, a sequence $\gamma^* \in \Gamma(V)$ and an integer $k \leq n$.
Output: Sequence of activation of z nodes in V starting with i_S with minimal expected time of activation.

1 **if** $k > 0$ **then**
2 $\quad\gamma \leftarrow \arg\min_{\gamma' \in \Upsilon[t]:\gamma' \sim \gamma^*} \tau^*[t, \gamma', k]$
3 $\quad\gamma' \leftarrow \langle\rangle$
4 $\quad$**for** $\gamma_i \in \gamma$ **do**
5 $\quad\quad$**if** $\gamma_i \notin \gamma^*$ **then**
6 $\quad\quad\quad\gamma' \leftarrow \gamma' \oplus \langle\gamma_i\rangle$
7 $\quad\quad$**else**
8 $\quad\quad\quad\gamma^* \leftarrow \gamma^*_{\lhd\gamma_i^*} \oplus \gamma' \oplus \gamma^*_{\blacktriangleright\gamma_i^*}$
9 $\quad\quad\quad\gamma' \leftarrow \langle\rangle$
10 $\quad\gamma^* \leftarrow \gamma^* \oplus \gamma'$
11 $\quad k' \leftarrow k - |\gamma|_t|$
12 $\quad$**for** $t_i \in \langle t_{|c(t)|}, \ldots, t_1\rangle$ **do**
13 $\quad\quad m \leftarrow \arg\min_{m' \in \{0,\ldots,k'\}} \left(\tau'[t, i-1, \gamma, m'] + \min_{\substack{\gamma' \in \Upsilon[t_i]: \\ \gamma' \sim \gamma}} \left(\tau^*[t_i, \gamma', k' - m' + |\gamma'|_t|] - \sum_{\gamma_i' \in \gamma'|_t} \tau[t_i, \gamma', \gamma_i']\right)\right)$
14 $\quad\quad\gamma^* \leftarrow$ ReconstructPartial$(t_i, \gamma^*, k' - m + |\gamma'|_t|)$
15 $\quad\quad k' \leftarrow m$
16 **return** γ^*

in the subtree rooted in t_i. The returned value of the minimal time of activation of k nodes in the subtree of t while using permutation γ is computed as the sum of the time of activation of nodes from t and the optimal time of activating the remaining $k - |\gamma|_t|$ nodes in all $|c(k)|$ children of t.

The time complexity of the algorithm is similar to the full diffusion case, except that we have now the two additional loops, one over k in step 3 and the other over m in line 6 of Algorithm 3, which contribute an additional factor of $\mathcal{O}(z^2)$ to the running time.

We are unable to compute an optimal solution in polynomial time in general case. We may however hope to find a polynomial approximation algorithm. We now move to the analysis of possible ways of approximating the optimal solution.

We now move to assessing the possibility of approximating the optimal solution to the Optimal Diffusion Sequence Problem.

4.5. Lower bounds on heuristic algorithms

Alshamsi et al. [1] suggest two heuristic strategies for activating a network in a process of strategic diffusion: the greedy strategy (always target node with the highest probability of activation) and the majority strategy (always target node with the highest number of active neighbors).

We now show that neither of these strategies approximates an optimal solution to within a constant ratio.

Theorem 7. *There exists no constant $r > 1$ such that choosing the sequence of activation using the greedy strategy is an r-approximation algorithm. In particular, the approximation ratio of the greedy strategy is $\Omega(\log n)$.*

Proof. We will now show a series of networks where the approximation ratio of the greedy algorithm goes to infinity.

For a given $k \in \mathbb{N}$ we construct a network $G(k)$, where the set of nodes is $V = \{i_S, a_1, \ldots, a_{k^2}, b_1, \ldots, b_{k-1}\}$ and where the set of edges is:

$$E = \bigcup_{i=1}^{k^2} \{i_S a_i\} \cup \bigcup_{i=1}^{k^2} \bigcup_{j=1}^{k-1} \{a_i b_j\}.$$

We assume that the weight of every edge is 1. The structure of network $G(k)$ is presented in Fig. 4.

Let x be the number of currently activated a_i nodes and let y be the number of currently activated b_i nodes. We have that $\tau(a_i) = \frac{k}{y+1}$ and we have that $\tau(b_i) = \frac{k^2}{x}$.

We will now analyze two different ways of activating all nodes in network $G(k)$. First, let us consider the greedy algorithm. Let the indices of nodes a_i, as well as the indices of nodes b_j, be ordered according to the order of activation, *i.e.*, node a_1 is the first activated node a_i, while node b_1 is the first activated node b_i. Notice that at least ik nodes a has to be activated before the activation of node b_i. We will prove this claim by contradiction. Assume that b_i can have $j < ik$ active neighbors at the moment of activation. Take node a_{j+1} (the first node a activated after b_i). At the moment of activation of b_i its probability of activation is $\frac{j}{k^2}$, while the probability of activation of a_{j+1} is $\frac{i}{k}$. Since we are using greedy algorithm

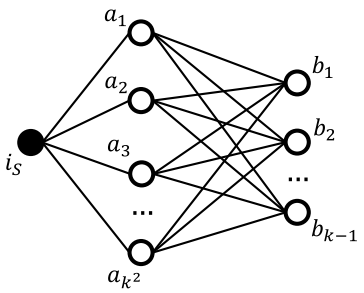

Fig. 4. Network $G(k)$ constructed for the proofs of Theorems 7 and 8.

and b_i was activated before a_{j+1}, we have to have $\frac{j}{k^2} \geq \frac{i}{k}$. However, this is not true since $j < ik$. We have a contradiction, therefore at least ik nodes a has to be activated before the activation of node b_i.

Because of this at least k nodes a_i has to be activated with only one neighbor active, then at least k nodes a_i has to be activated with only two neighbors active, and so on. Focusing only on the time of activation of nodes a_i we have that the total expected time of activation for the greedy algorithm is:

$$\tau_G(G(k)) \geq \sum_{i=1}^{k^2} \tau(a_i) \geq \sum_{j=1}^{k} k\frac{k}{j} = k^2 H_k$$

where H_k is the k-th harmonic number.

Now, let us consider an algorithm A, where we first activate k nodes a_i, then all nodes b_i and finally the remaining nodes a_i. Time of activation of the entire network is then:

$$\tau_A(G(k)) = k^2 + (k-1)k + (k^2 - k) = 3k^2 - 2k$$

since each of the first k nodes a_i is activated in expected time k (as it only has one active neighbor, namely i_S), each node b_i is activated in expected time k (as it has degree k^2 and k active neighbors at the moment of activation) and each of the remaining $k^2 - k$ nodes a_i is activated in expected time 1.

Consider sequence of networks $G(k)$. For this sequence we have

$$\lim_{k\to\infty} \frac{\tau_G(G(k))}{\tau^*(G(k))} \geq \lim_{k\to\infty} \frac{\tau_G(G(k))}{\tau_A(G(k))} \geq \lim_{k\to\infty} \frac{H_k}{3} = \infty.$$

Therefore, solution provided by the greedy algorithm can be arbitrarily worse than the optimal solution. □

We also show analogical result for the majority strategy.

Theorem 8. *There exists no constant $r > 1$ such that choosing the sequence of activation using the majority strategy is an r-approximation algorithm. In particular, the approximation ratio of the majority strategy is $\Omega(\log n)$.*

Proof. We will now show that for the series of networks constructed in the proof of Theorem 7 the approximation ratio of the majority algorithm also goes to infinity.

Let x be the number of currently activated a_i nodes and let y be the number of currently activated b_i nodes. We have that $\tau(a_i) = \frac{k}{y+1}$ and we have that $\tau(b_i) = \frac{k^2}{x}$.

Let us consider the expected activation time of an entire network $G(k)$ using majority algorithm. Let the indices of nodes a_i, as well as the indices of nodes b_j, be ordered according to the order of activation, *i.e.*, node a_1 is the first activated node a_i, while node b_1 is the first activated node b_i. Notice that node b_i at the moment of activation has at most $i+1$ active neighbors. We will prove this claim by contradiction. Assume that b_i can have $j > i+1$ active neighbors at the moment of activation. Consider number of active neighbors at the moment of activation of node a_j (the last node a activated before b_i). Since we are using the majority algorithm, it has to have at least $j-1$ active neighbors (otherwise node b_i, having at the moment $j-1$ active neighbors, would have been chosen before node a_j). However, node a_j can have at most $i < j-1$ active neighbors, namely nodes $b_1, \ldots, b_{i-1}$ and node i_S (as node b_i is not active yet). We have a contradiction, therefore b_i at the moment of activation has at most $i+1$ active neighbors.

Focusing only on the time of activation of nodes b_i we have that the total expected time of activation for the majority algorithm is:

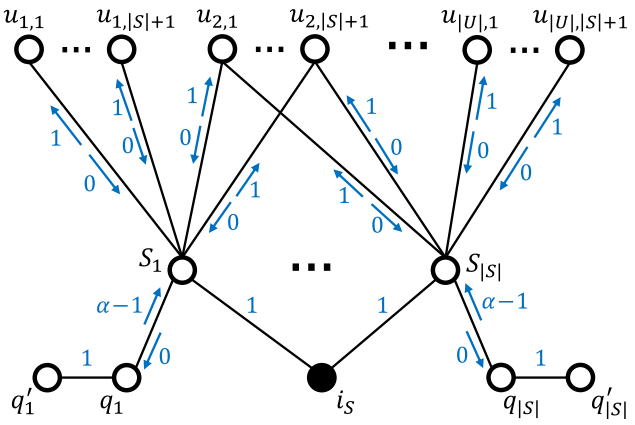

Fig. 5. Network $G(X, r)$ constructed for the proof of Theorem 9. Blue numbers next to edges express their weights. If there are no arrows next to edge ij then $w_{ij} = w_{ji}$. Otherwise weight w_{ij} is denoted next to the arrow pointing towards node j, and weight w_{ji} is denoted next to the arrow pointing towards node i.

$$\tau_C(G(k)) \geq \sum_{i=1}^{k-1} \tau(b_i) \geq \sum_{i=1}^{k-1} \frac{k^2}{i+1} = k^2(H_k - 1)$$

where H_k is the k-th harmonic number.

Let A be the alternative algorithm described in the proof of Theorem 7. Consider sequence of networks $G(k)$. For this sequence we have

$$\lim_{k\to\infty} \frac{\tau_C(G(k))}{\tau^*(G(k))} \geq \lim_{k\to\infty} \frac{\tau_C(G(k))}{\tau_A(G(k))} \geq \lim_{k\to\infty} \frac{H_k}{3} = \infty$$

Therefore, solution provided by the majority algorithm can be arbitrarily worse than the optimal solution. □

4.6. Inapproximability

Finally, we show that in fact the problem cannot be approximated within a ratio of $(1-\epsilon)\ln n$ for any $\epsilon > 0$, unless $P = NP$.

Theorem 9. *The Optimal Partial Diffusion Sequence problem cannot be approximated within a ratio of* $(1-\epsilon)\ln n$ *for any* $\epsilon > 0$, *unless* $P = NP$.

Proof. In order to prove the theorem, we will use the result by Dinur and Steurer [27] that the Minimum Set Cover problem cannot be approximated within a ratio of $(1-\epsilon)\ln n$ for any $\epsilon > 0$, unless $P = NP$.

Let $X = (U, S)$ be an instance of the Minimum Set Cover problem. To remind the reader, U is the universe $\{u_1, \ldots, u_{|U|}\}$, while S is a collection $\{S_1, \ldots, S_{|S|}\}$ of subsets of U. The goal here is to find subset $S^* \subseteq S$ such that the union of S^* equals U and the size of S^* is minimal.

First, we will show a function $f(X)$ that based on an instance of the Minimum Set Cover problem X constructs an instance of the Optimal Partial Diffusion Sequence problem.

Let network $G(X)$ be defined as follows (an example of such network is presented in Fig. 5):

- **The set of nodes:** For every $S_i \in S$, we create three nodes, denoted by S_i, q_i and q'_i. For every $u_i \in U$, we create $|S|+1$ nodes, denoted by $u_{i,1}, \ldots, u_{i,|S|+1}$. Additionally, we create a single node i_S.
- **The set of edges:** For every node S_i, we create an edge $S_i i_S$. For every node q_i, we create edges $q_i S_i$ and $q_i q'_i$. Finally, for every node $u_{i,i'}$ and every S_j such that $u_i \in S_j$ we create an edge $u_{i,i'} S_j$.
- **The weight matrix:** For every edge $S_i q_i$ we set its weights to $w_{S_i q_i} = 0$ and $w_{q_i S_i} = \alpha - 1$, where $\alpha = z|S|^{\lambda+1}$ for some $\lambda > 0$. For every edge $u_{i,i'} S_j$ we set its weights to $w_{u_{i,i'} S_j} = 0$ and $w_{S_j u_{i,i'}} = 1$. For all other pairs of nodes connected with an edge we set their weights to 1.

To complete the constructed instance of the Optimal Partial Diffusion Sequence problem, we set the seed node to be i_S and we set the number of nodes to be activated to $z = |U|(|S|+1)+1$. Hence, the formula of function f is $f(X) = (G(X), i_S, z)$.

Let γ be the solution to the constructed instance of the Optimal Partial Diffusion Sequence problem. The function g computing corresponding solution to the instance X of the Minimum Set Cover problem is now $g(X, \gamma^*) = S \cap \gamma^*$, *i.e.*, S^* is the set of all sets S_i such that their corresponding nodes S_i appear in sequence γ^*.

Now, we will show that $g(X, \gamma^*, r)$ is indeed a correct solution to X, *i.e.*, that it covers the entire universe. Notice that none of the nodes q_i nor q_i' can get activated while i_S is the seed node, since the influence of S_i on q_i is 0. Hence, sequence γ^* can consist only of i_S and nodes in $S \cup U$. Moreover, since we have to activate $|U|(|S|+1)$ of the nodes $S \cup U$, at least one node from each group $u_{i,1}, \ldots, u_{i,|S|+1}$ have to be activated. The only way to do it is to activate a node S_j such that $u_i \in S_j$. Therefore, for every $u_i \in U$ there exists a node $S_j \in \gamma^*$ such that $u_i \in S_j$ and $g(X, \gamma)$ has to be a set cover of U.

We will now prove three lemmas concerning functions f and g.

Lemma 1. *Size of the solution to the given instance of the Minimum Set Cover problem returned by function g is lesser or equal than the expected time of activation of the corresponding solution to the constructed instance of the Optimal Partial Diffusion Sequence problem γ divided by α, i.e., $|g(X, \gamma)| \leq \frac{\tau(\gamma)}{\alpha}$.*

Proof. Let κ be the number of nodes S_i in sequence γ. Since the expected time of activation of a node S_i is always α, we have $\tau(\gamma) \geq \kappa\alpha$. From the definition of g we have $|g(X, \gamma)| = \kappa$, hence we have $|g(X, \gamma)| \leq \frac{\tau(\gamma)}{\alpha}$. □

Lemma 2. *For sequence γ, the solution of $f(X)$, and the corresponding solution to X returned by function g we have $\tau(\gamma) \leq |g(X, \gamma)|\alpha\left(1 + \frac{1}{|S|^\lambda}\right)$.*

Proof. Let κ be the number of nodes S_i in sequence γ. Since the expected time of activation of a node S_i is always α and the expected time of activation of a node $u_{i,j}$ is smaller or equal than $|S|$, we have $\tau(\gamma) \leq \kappa\alpha + (z - \kappa - 1)|S| \leq \kappa\alpha + z|S| \leq \kappa\alpha + \kappa z|S| = \kappa\alpha + \kappa\frac{\alpha}{|S|^\lambda}$. From the definition of g we have $|g(X, \gamma)| = \kappa$, hence we have $\tau(\gamma) \leq |g(X, \gamma)|\alpha\left(1 + \frac{1}{|S|^\lambda}\right)$. □

Lemma 3. *An optimal solution to the constructed instance of the Optimal Partial Diffusion Sequence problem γ^* corresponds to an optimal solution to the given instance of the Minimum Set Cover problem S^*, i.e., $S^* = g(X, \gamma^*)$.*

Proof. As noted above, if γ is a solution to $f(X)$, then $g(X, \gamma)$ is also a correct solution to X, *i.e.*, $g(X, \gamma)$ covers entire universe. Since the expected time of activation of every node S_i is α, the expected time of activation of every node $u_{i,j}$ is smaller or equal than $|S|$, and $|S| < \alpha$, the optimal solution to $f(X)$ is the one that minimizes the number of nodes S_i is γ, hence, the one that minimizes $|g(X, \gamma)|$. □

Now, assume that there exists an r-approximation algorithm for the Optimal Partial Diffusion Sequence problem where $r = (1 - \epsilon)\ln n$ for some $\epsilon > 0$. Let us use this algorithm to solve the constructed instance $f(X)$ and consider solution $g(X, \gamma)$ to the given instance of the Minimum Set Cover problem.

We then have:

$$|g(X, \gamma)| \leq \frac{\tau(\gamma)}{\alpha} \leq \frac{r\tau(\gamma^*)}{\alpha} \leq r\left(1 + \frac{1}{|S|^\lambda}\right)|g(X, \gamma^*)| = r\left(1 + \frac{1}{|S|^\lambda}\right)|S^*|,$$

where first inequality comes from Lemma 1, second inequality comes from the fact that we consider an r-approximation algorithm, third inequality comes from Lemma 2, and the final equality comes from Lemma 3. Hence, we obtained an $r\left(1 + \frac{1}{|S|^\lambda}\right)$-approximation algorithm for the Minimum Set Cover problem, the ratio of which is lower than $\ln(n)$ for large enough λ. However, as shown by Dinur and Steurer [27], it is not possible unless $P = NP$. Therefore, there cannot exist such r-approximation algorithm for the Optimal Partial Diffusion Sequence problem. This concludes the proof of Theorem 9. □

We now move to describing the conclusions and possible ideas for future work.

5. Conclusions & future work

In this article we investigate the computational aspects of the strategic diffusion problem previously introduced by Alshamsi et al. [1]. We show that the partial diffusion problem is NP-complete in the general case, hence finding a polynomial algorithm is impossible, unless P=NP. Given this difficulty, we considered the problem from the parametrized complexity point of view and showed that the problem is fixed parameter-tractable when parametrized by the sum of the treewidth and maximum degree. On the negative side, we showed that two previously proposed heuristic algorithms for solving the problem, *i.e.*, the greedy and the majority solutions, cannot have better than a logarithmic approximation guarantee, even in the full diffusion case. Finally, we proved that the partial diffusion problem does not admit better than a logarithmic approximation, unless P=NP.

As for the potential ideas for future work, determining the complexity of the full diffusion problem is an interesting open question. Furthermore, our results concerning approximation algorithms are negative. Developing a polynomial algorithm with small approximation ratio would make the task of finding an efficient way of spreading the strategic diffusion much simpler. Another possible way of extending our work is showing effective algorithms finding the optimal solution in more restricted classes of networks, *e.g.*, networks with bounded treewidth or bounded pathwidth (without the degree restriction).

The model of strategic diffusion presented in this work is based on a complex contagion process, *i.e.*, multiple sources of exposure to the process greatly increase the probability of activation for a given node. It would be possible to devise a similar model, but based on a simple contagion process, *e.g.*, taking inspiration from the independent cascade model [9]. Since simple contagion does not take multiple exposures into consideration, nodes activated earlier in the sequence would not affect the probability of activation in a given moment. This could lead to optimal strategies significantly different than for the strategic diffusion based on complex contagion.

Declaration of competing interest

Authors declare no conflicts of interest.

Acknowledgements

This work was funded by the Cooperative Agreement between the Masdar Institute of Science and Technology (Masdar Institute), Abu Dhabi, UAE and the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA—Reference 02/MI/MIT/CP/11/07633/GEN/G/00. CAH and FLP acknowledge the support from the MIT Media Lab consortia.

References

[1] A. Alshamsi, F.L. Pinheiro, C.A. Hidalgo, Optimal diversification strategies in the networks of related products and of related research areas, Nat. Commun. 9 (1) (2018) 1328.
[2] T.W. Valente, Network Models of the Diffusion of Innovations, Hampton Press, Cresskill, New Jersey, 1995.
[3] E.M. Rogers, Diffusion of Innovations, Simon and Schuster, 2010.
[4] N.T. Bailey, et al., The Mathematical Theory of Infectious Diseases and Its Applications, Charles Griffin & Company Ltd, 5a Crendon Street, High Wycombe, Bucks HP13 6LE, 1975.
[5] R. Pastor-Satorras, A. Vespignani, Epidemic spreading in scale-free networks, Phys. Rev. Lett. 86 (14) (2001) 3200.
[6] S. Aral, C. Nicolaides, Exercise contagion in a global social network, Nat. Commun. 8 (2017) 14753.
[7] V.V. Vasconcelos, S.A. Levin, F.L. Pinheiro, Consensus and polarization in competing complex contagion processes, J. R. Soc. Interface 16 (155) (2019) 20190196.
[8] W.O. Kermack, A.G. McKendrick, A contribution to the mathematical theory of epidemics, Proc. R. Soc. Lond. Ser. A, Contain. Pap. Math. Phys. Character 115 (772) (1927) 700–721.
[9] D. Kempe, J. Kleinberg, É. 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, ACM, 2003, pp. 137–146.
[10] J. Goldenberg, B. Libai, E. Muller, Using complex systems analysis to advance marketing theory development: modeling heterogeneity effects on new product growth through stochastic cellular automata, Acad. Mark. Sci. Rev. 2001 (2001) 1.
[11] J. Leskovec, L.A. Adamic, B.A. Huberman, The dynamics of viral marketing, ACM Trans. Web 1 (1) (2007) 5.
[12] P. Domingos, M. Richardson, Mining the network value of customers, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2001, pp. 57–66.
[13] C. Kiss, M. Bichler, Identification of influencers—measuring influence in customer networks, Decis. Support Syst. 46 (1) (2008) 233–253.
[14] F. Morone, H.A. Makse, Influence maximization in complex networks through optimal percolation, Nature 524 (7563) (2015) 65.
[15] A. Goyal, F. Bonchi, L.V. Lakshmanan, A data-based approach to social influence maximization, Proc. VLDB Endow. 5 (1) (2011) 73–84.
[16] W. Chen, Y. Wang, S. Yang, Efficient influence maximization in social networks, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2009, pp. 199–208.
[17] O. Hinz, B. Skiera, C. Barrot, J.U. Becker, Seeding strategies for viral marketing: an empirical comparison, J. Mark. 75 (6) (2011) 55–71.
[18] B. Libai, E. Muller, R. Peres, Decomposing the value of word-of-mouth seeding programs: acceleration versus expansion, J. Mark. Res. 50 (2) (2013) 161–176.
[19] G. Tong, W. Wu, S. Tang, D.-Z. Du, Adaptive influence maximization in dynamic social networks, IEEE/ACM Trans. Netw. 25 (1) (2017) 112–125.
[20] L. Seeman, Y. Singer, Adaptive seeding in social networks, in: 2013 IEEE 54th Annual Symposium on Foundations of Computer Science, IEEE, 2013, pp. 459–468.
[21] T. Horel, Y. Singer, Scalable methods for adaptively seeding a social network, in: Proceedings of the 24th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2015, pp. 441–451.
[22] J. Jankowski, P. Bródka, P. Kazienko, B.K. Szymanski, R. Michalski, T. Kajdanowicz, Balancing speed and coverage by sequential seeding in complex networks, Sci. Rep. 7 (1) (2017) 891.
[23] J. Jankowski, M. Waniek, A. Alshamsi, P. Bródka, R. Michalski, Strategic distribution of seeds to support diffusion in complex networks, PLoS ONE 13 (10) (2018) e0205130.
[24] C.A. Hidalgo, B. Klinger, A.-L. Barabási, R. Hausmann, The product space conditions the development of nations, Science 317 (5837) (2007) 482–487.
[25] M.R. Guevara, D. Hartmann, M. Aristarán, M. Mendoza, C.A. Hidalgo, The research space: using career paths to predict the evolution of the research output of individuals, institutions, and nations, Scientometrics 109 (3) (2016) 1695–1709.
[26] F.L. Pinheiro, A. Alshamsi, D. Hartmann, R. Boschma, C. Hidalgo, Shooting low or high: do countries benefit from entering unrelated activities?, arXiv preprint, arXiv:1801.05352.
[27] I. Dinur, D. Steurer, Analytical approach to parallel repetition, in: Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing, ACM, 2014, pp. 624–633.
[28] R.M. Karp, Reducibility among combinatorial problems, in: Complexity of Computer Computations, Springer, 1972, pp. 85–103.

[29] R. Bellman, et al., The theory of dynamic programming, Bull. Am. Math. Soc. 60 (6) (1954) 503–515.
[30] D.P. Bertsekas, Dynamic Programming and Optimal Control, vol. 1, Athena Scientific, Belmont, MA, 1995.
[31] D.P. Bertsekas, J.N. Tsitsiklis, Neuro-dynamic Programming, vol. 5, Athena Scientific, Belmont, MA, 1996.
[32] J. Rust, Dynamic programming, in: The New Palgrave Dictionary of Economics, 2016, pp. 1–26.
[33] R.E. Bellman, S.E. Dreyfus, Applied Dynamic Programming, vol. 2050, Princeton University Press, 2015.
[34] H.L. Bodlaender, Dynamic programming on graphs with bounded treewidth, in: Automata, Languages and Programming, Proceedings of the 15th International Colloquium, ICALP88, Tampere, Finland, July 11–15, 1988, 1988, pp. 105–118.
[35] L.P. Kaelbling, M.L. Littman, A.W. Moore, Reinforcement learning: a survey, J. Artif. Intell. Res. 4 (1996) 237–285.
[36] N. Robertson, P. Seymour, Graph minors. III. Planar tree-width, J. Comb. Theory, Ser. B 36 (1) (1984) 49–64.
[37] R.R.G. Downey, M. Fellows, Parameterized Complexity, Monographs in Computer Science, Springer Verlag, 1999, https://books.google.co.jp/books?id=pt5QAAAAMAAJ.