RESEARCH ARTICLE

MATI: An efficient algorithm for influence maximization in social networks

Maria-Evgenia G. Rossi[1]*, **Bowen Shi**[1], **Nikolaos Tziortziotis**[1], **Fragkiskos D. Malliaros**[2], **Christos Giatsidis**[1], **Michalis Vazirgiannis**[1]

1 École Polytechnique, Palaiseau, France, **2** CentraleSupélec, University of Paris-Saclay and Inria Saclay, Gif-sur-Yvette, France

* maria.rossi@polytechnique.edu

Abstract

Influence maximization has attracted a lot of attention due to its numerous applications, including diffusion of social movements, the spread of news, viral marketing and outbreak of diseases. The objective is to discover a group of users that are able to maximize the spread of influence across a network. The greedy algorithm gives a solution to the Influence Maximization problem while having a good approximation ratio. Nevertheless it does not scale well for large scale datasets. In this paper, we propose Matrix Influence, MATI, an efficient algorithm that can be used under both the Linear Threshold and Independent Cascade diffusion models. MATI is based on the precalculation of the influence by taking advantage of the simple paths in the node's neighborhood. An extensive empirical analysis has been performed on multiple real-world datasets showing that MATI has competitive performance when compared to other well-known algorithms with regards to running time and expected influence spread.

Citation: Rossi M-EG, Shi B, Tziortziotis N, Malliaros FD, Giatsidis C, Vazirgiannis M (2018) MATI: An efficient algorithm for influence maximization in social networks. PLoS ONE 13(11): e0206318. https://doi.org/10.1371/journal.pone.0206318

Editor: Irene Sendiña-Nadal, Universidad Rey Juan Carlos, SPAIN

Received: March 14, 2018

Accepted: October 10, 2018

Published: November 1, 2018

Data Availability Statement: All the datasets used in our paper are freely available in http://snap.stanford.edu/data/.

Funding: This work was partially supported by a DigiCosme Ph.D. Fellowship and the Ecole Polytechnique AXA Chair (DaSciS).

Competing interests: The authors have declared that no competing interests exist.

Introduction

During the last decade, online social networking sites (e.g., Facebook, Twitter, LinkedIn, Tumblr etc.) and relevant smartphone applications have caused a remarkable growth of research on social networks. This led to the development of many applications of social networks of which a rich body of studies has been classified as *the analysis of influence or information diffusion in social networks*.

Influence propagation studies have found applications in various fields. From studying human, animal or even plant epidemics [1–3] to viral marketing [4], social media analytics [5], spread of rumors [6], expert finding [7], recommendation systems [8–10], etc. A key task in order to understand information and influence diffusion is the identification of vital nodes that play a significant role in these cases. Such nodes may allow us to control the spread of an epidemic [11, 12], to predict successful scientists and scientific publications based on co-authorship and citation networks [13–15], to design influential advertisements for new products [4, 16], etc.

For example, in the case of virus propagation, such as influenza, the transmission of the disease mainly depends on the extend of contacts of the infected person to the susceptible population; thus, being able to locate and vaccinate individuals with good spreading properties can prevent from a potential outbreak of the disease, leading to efficient strategies of epidemic control. In a similar way, suppose that our goal is to promote an idea or a product in order to be adopted by a large fraction of individuals in the network. A key idea behind viral marketing is the word-of-mouth effect [17]; individuals that have already adopted the product, recommend it to their friends who in turn do the same to their own social circle, forming a cascade of recommendations [18]. The basic question here is how to target a few initial individuals (e.g., by giving them free samples of the product or explaining them the idea), that can maximize the spread of influence in the network, leading to a successful promotion campaign.

Nevertheless, locating these users in a network is not a trivial task and numerous research has been conducted to solve the problem in the area [19]. It has been of significant importance to identify these nodes that will maximize the influence and information diffusion at the end of a respective phenomenon in a network. The problem is actually split into two subtopics: i) Identification of *individual influential nodes* that have good spreading properties [20–23] and ii) Identification of *a group of nodes* that by acting all together will maximize the total spread of influence in a network [24–30], or as usually called *Influence Maximization* (IM). Indeed, the two tasks greatly differ, as finding a ranking of the nodes that by acting individually can influence a great part of the network cannot directly be used to discover the set of nodes that will—by acting at the same time—maximize the spreading of information in a graph. Of course, this is justified by the fact that putting some of the most influential spreaders together will not result in a most influential set of these spreaders, because their respective influences may be, and is usually, largely overlapped.

Identification of a group of influential spreaders

In this work we will be studying the specific problem that concerns the identification of a group of influential spreaders. Influence maximization can formally be described as follows: given a *social network* where the relations among users are revealed, a *diffusion model* that simulates how information propagates through the network and a parameter k, the goal is to locate those k users (represented as nodes in the graph) that maximize the spread of influence.

Kempe et al. [29] formulated the problem in the aforementioned manner and proved that it is NP-hard while adopting two diffusion models borrowed from mathematical sociology: the *Linear Threshold* (LT) and the *Independent Cascade* (IC) model. According to both, at any discrete time step a user can be either active or inactive (i.e., has adopted the product or not) and the information propagates until no more users can be activated. The authors proved that the function of the influence spread (i.e., the function that gives the number of the nodes that will be active at the end of the spreading process) under both LT and IC models is monotone and submodular in respect to the number of the seed nodes that trigger the information spreading. By exploiting these properties, they presented a greedy algorithm that achieves $(1 - 1/e)$ approximation ratio.

However, as the greedy algorithm repeatedly selects in every iteration the node with the maximum marginal gain by running Monte Carlo simulations, we are lead towards great performance downsides. For that reason the majority of the literature has been providing approximate rather than the exact solution.

Related work

Following the seminal work by Kempe et al. [29], a series of algorithms have been proposed in order to: (i) reduce the number of influence spread evaluations, (ii) make batch computations of the influence spread, and (iii) design scalable heuristics towards computing the respective spread.

Leskovec et al. [30] proposed the CELF algorithm, based on a "lazy-forward" optimization scheme, that finds near-optimal solutions guaranteed to achieve at least $1/2(1 - 1/e)$ of the optimal ones by being 700 times faster than the greedy algorithm. The fact that the marginal gain of a node cannot be greater than the one achieved in previous iterations is taken into account. A table which stores every node and its marginal gain (with regards to the influence achieved by the so far selected candidates) is stored in decreasing order. Only the marginal gain of the most profitable (top) node is re-evaluated when it is needed and the table is sorted again. The node remaining at the top of the table is selected as the next seed node. Chen et al. [31] designed new algorithms that improve the greedy one and combine techniques from the aforementioned CELF algorithm towards faster greedy algorithms. Goyal et al. [27] introduced an algorithm that further optimized CELF by 35–55% in terms of running time, called CELF++. By exploiting the submodularity property of the spread function for the diffusion models (e.g., LT and IC) it avoids the unnecessary re-computations of the marginal gains.

Chen et al. [24] proposed the LDAG algorithm which is tailored for the LT model and achieves to produce results by being orders of magnitude faster than the greedy algorithm. They firstly show that the computation of influence in directed acyclic graphs (DAGs) can be done in linear time. Based on that, they construct a local DAG for every node of the network and restrict the influence of the node in this local area. After constructing the DAGs the greedy seed selection approach is applied together with an accelerated solution for updating the incremental influence spread of each node.

Jiang et al. [32] used simulated annealing to solve the IM problem and proposed the SAEDV algorithm which runs 2-3 times faster than the greedy while its accuracy is also improved. The MIA algorithm, proposed by Chen et al. [25], is a maximum influence arborescence model based on the assumption that information diffusion occurs according to the IC model. They succeed in proposing a scalable algorithm which produces results that outperform the other so far proposed heuristics by 100–120% in terms of running time. They propose i) a best-effort algorithm that estimates the upper bounds of location-aware influence spread and prunes users having small influences thus achieving an approximation ratio of $(1 - 1/e)$ and ii) a topic-materialization-based algorithm that estimates the bounds of influence spread and avoids computation of the actual influence of least influential users while finally achieving an approximation of $\epsilon(1 - 1/e)$ for any for any $\epsilon \in (0, 1]$.

Jung et al. [33] proposed IRIE which incorporates fast iterative ranking algorithm (IR) with a fast influence estimation (IE) method. By avoiding to store and compute local data structures they result in compelling savings in memory usage and running time. Goyal et al. [28] proposed SimPath, an algorithm tailored for the LT model which computes the influence spread by enumerating simple paths within a small neighborhood. With the help of a parameter, a balance between running time and quality of the solution can be achieved.

Ohsaka et al. [34] developed a method that takes advantage of privileged nodes in social networks to spead up any breadth first search performed to test a node's reachability, without loss of solution quality. Their algorithm has comparable running times with the aforementioned IRIE and SAEDV algorithms and is more robust when compared to PMIA. Tang et al. [35] designed the Two-phase Influence Maximization (TIM) algorithm which runs in near-linear time while also returning $(1 - 1/e - 1/\epsilon)$-approximate solutions. In the first phase, a lower-bound of the maximum spread is calculated in order to derive a parameter θ. In the second phase, the parameter θ is used so that random reverse reachable (RR) sets (as defined by Borg et al. [36]) are sampled. The k-sized node set that covers a large number of RR sets is the final result. TIM supports the *triggering model* which is a general diffusion model incorporating both the LT and IC models. The same authors that proposed TIM, proposed the Influence Maximization via Martingales (IMM) algorithm [37]. The basis of this algorithm is a set of estimation techniques based on the *martingales* statistical tool [38]. IMM also adopts the two-phase paradigm of TIM but incorporates a drastically different parameter estimation phase and achieves to derive a lower bound of the maximum expected influence of a node-set of any size (OPT) no less than $OPT \cdot (1 - 1/e)/(1 + \epsilon')^2$ where ϵ is a tunable parameter.

Another interesting approach is the one proposed by Cohen et al. [26]. They designed a SKetch-based Influence Maximization (SKIM) algorithm which uses per-node summary structures called *combined reachability sketches* representing the node's influence coverage [39]. They introduced *influence oracles* which can answer *influence queries* in an efficient way. This algorithm is designed based on the IC diffusion model. Goyal et al. [40] proposed a new probability model, the *credit distribution model*, which directly estimates influence spread by exploiting historical data. This makes the need for knowing the influence probabilities and making Monte Carlo simulations to compute the respective influence redundant—thus avoiding costly computations.

In this paper, we propose MATI, an efficient influence maximization algorithm under both the LT and IC diffusion models. By taking advantage of the possible paths that are created in each node's neighborhood, we have designed an algorithm that succeeds in locating the users that can maximize the influence in a social network while also being scalable for large datasets. Specifically we take advantage of the fact that when we want to calculate the influence gain that a node will add to the influence of a group of nodes, we just need to subtract the influence of the common paths of the new node and those of our already existing seeds from this new node's individual influence to the network. Our algorithm can be seen as an extension of the SimPath algorithm in what concerns its formulation for the LT model. To the best of our knowledge, taking advantage of the possible paths created and pre-calculating the influence gain of a node for the IC model though has not beed proposed, in the best of our knowledge, by any related work. The methods used to precalculate each node's potential influence depends on the creation of matrices which may on one hand increase the memory consumption of the algorithm while at the same time facilitating the re-computation of the seeds in the case that some nodes and edges are deleted. In order to limit the computation of the possible paths and the respective probabilities of them being "active", we use a pruning threshold θ that acts as a trade-off between the running time of the algorithm and the accuracy of the influence computation. Extensive experiments show that for both the cases of the LT and IC models, MATI performs better than the baseline methods both in terms of influence and computation time.

Preliminaries

A social network is typically modeled as a directed graph $G = (V, E)$, with each node $u \in V$ to represent a user, and the edges $E \subseteq V \times V$ to reflect the relationships between users. An influence weight $p_{u,v} \in [0, 1]$ is also associated with each directed edge $(u, v) \in E$, and represents the probability of node u to influence node v. Let us denote as $\mathcal{T}(u) = \{\tau_1, \tau_2, \ldots, \tau_M\}$ the set of all possible unique paths (M in total) that exist in the graph starting from node u, with $\tau_i \not\subset \tau_j, \forall j \in M$ and $j \neq i$. Roughly speaking, any path $\tau_i \in \mathcal{T}(u)$ cannot be presented as a subpath of any other path $\tau_j \in \mathcal{T}(u), j \neq i$ presented in the same set of paths. It should be also noticed that our paths are not allowed to contain cycles. For all the above reasons, we use the Depth-first search (DFS) algorithm for the generation of the paths that start from node u, with the node u to be the root of the *tree*. Each path $\tau_i \in \mathcal{T}(u)$ consists of a sequence of nodes $\tau_i = \{n_{i1}, n_{i2}, \ldots, n_{iN}\}$, where N is the length of the path. M and N can obviously be different for every node u and every path τ_i, respectively, but they are defined as such for the sake of the simplicity of the model.

Let $p^{\tau_i}_{\ell,\ell+1}$, $1 \leq \ell \leq N-1$, represent the influence weight (probability) between two successive nodes ($n_{i\ell}$ and $n_{i(\ell+1)}$) in path τ_i. Then, $\mathcal{F}(\tau_i) = \{f_{i1}, f_{i2}, \ldots, f_{iN}\}$ represents the probabilities for every subpath in τ_i starting from node u to be *a*ctive (i.e., a path is considered active if each one of its edges is active). The f_{ij} corresponds to the probability that the subpath $\{n_{i1}, n_{i2}, \ldots, n_{ij}\}$ is active, and is defined as follows:

$$f_{ij} = \begin{cases} \prod_{\ell=1}^{j-1} p^{\tau_i}_{\ell,\ell+1} & \text{if} \quad j > 1, \\ 1 & \text{otherwise.} \end{cases} \tag{1}$$

Let us now define as $\Psi(u, v) = \{\psi_1, \psi_2, \ldots, \psi_L\}$ the set of all possible (unique) paths from a node u to a node v. ψ_i represents each possible path and L is the number of all possible paths between nodes u and v. Each path ψ_i consists of a sequence of nodes $\psi_i = \{n_{i1}, n_{i2}, \ldots, n_{iN}\}$ where N represents again the number of nodes of path ψ_i. Obviously $L \leq M$ and again L and N can be different for every set of paths between two nodes and every path ψ_i, respectively. We can now respectively define as $\Phi(\psi_i) = \{\phi_{i1}, \phi_{i2}, \ldots, \phi_{iN}\}$ the probability for every path ψ_i between two nodes u and v which is calculated in the same way as f_{ij} (see Eq (1)).

In this point, we will present the above notations by using a toy example. For this purpose, we consider the graph illustrated in Fig 1 that consists of $|V| = 8$ nodes and $|E| = 9$ edges. The set of paths starting from node u is defined as $\mathcal{T}(u) = \{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6\}$, where:

$$\begin{aligned} \tau_1 &= \{u, a, v, c, f\}, & N &= 5 \\ \tau_2 &= \{u, a, v, c, e\}, & N &= 5 \\ \tau_3 &= \{u, a, v, d, e\}, & N &= 5 \\ \tau_4 &= \{u, a, b, v, c, f\}, & N &= 6 \\ \tau_5 &= \{u, a, b, v, c, e\}, & N &= 6 \\ \tau_6 &= \{u, a, b, v, d, e\}, & N &= 6. \end{aligned}$$

Therefore, the probability path for τ_1 is defined as: $\mathcal{F}(\tau_1) = \{f_{11}, f_{12}, f_{13}, f_{14}, f_{15}\}$, where $f_{11} = 1, f_{12} = 0.1, f_{13} = 0.03, f_{14} = 0.003$ and $f_{15} = 0.0009$.

In the same way, the set of all possible paths from node u to node v is defined as $\Psi(u, v) = \{\psi_1, \psi_2\}$ with $L = 2$ and the different paths between the two nodes being the following:

$$\begin{aligned} \psi_1 &= \{u, a, v\}, & N &= 3 \\ \psi_2 &= \{u, a, b, v\}, & N &= 4. \end{aligned}$$

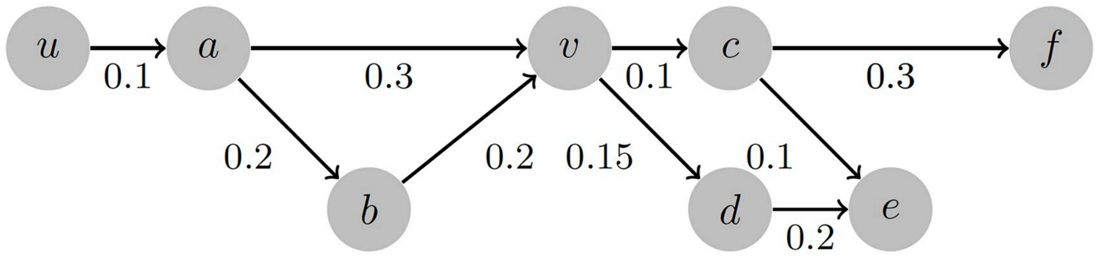

Fig 1. Example graph.

https://doi.org/10.1371/journal.pone.0206318.g001

Table 1. List of symbols used in the paper.

Notation	Description
$p_{u,v}$	Influence weight on directed edge (u, v)
$\sigma(S)$	Influence of a set of nodes S to the graph
$\mathcal{A}(u, v)$	Influence of node u to node v
$\Omega(u, v)$	Forward cumulative influence of node u to node v
$\mathcal{T}(u) = \{\tau_1, \tau_2, \ldots, \tau_M\}$	Set of all possible paths starting from node u
$\tau_i = \{n_{i1}, n_{i2}, \ldots, n_{iN}\}$	Path consisting of N nodes starting from node u
$\mathcal{F}(\tau_i) = \{f_{i1}, f_{i2}, \ldots, f_{iN}\}$	Cumulative probability path for path τ_i
$p^{\tau}_{\ell,\ell+1}$	Influence weight between successive nodes in τ
$\Psi(u, v) = \{\psi_1, \psi_2, \ldots, \psi_L\}$	Set of all possible paths between nodes u and v
$\psi_i = \{n_{i1}, n_{i2}, \ldots, n_{iN}\}$	Path between nodes u and v
$\Phi(\psi_i) = \{\phi_{i1}, \phi_{i2}, \ldots, \phi_{iN}\}$	Cumulative probability path for path ψ_i

https://doi.org/10.1371/journal.pone.0206318.t001

Then, the probability path for ψ_1 is defined as: $\Phi(\psi_1) = \{\phi_{11}, \phi_{12}, \phi_{13}\}$ with $\phi_{11} = 1$, $\phi_{12} = 1 * 0.1 = 0.1$ and $\phi_{13} = 1 * 0.1 * 0.3 = 0.03$. Similarly, $\Phi(\psi_2) = \{\phi_{21}, \phi_{22}, \phi_{23}, \phi_{24}\}$ with $\phi_{21} = 1$, $\phi_{22} = 1 * 0.1 = 0.1$, $\phi_{23} = 1 * 0.1 * 0.2 = 0.02$ and $\phi_{24} = 1 * 0.1 * 0.2 * 0.2 = 0.004$.

All the notations used throughout the paper are summarized in Table 1.

The Influence Maximization (IM) problem

In this Section we present the problem of *Influence Maximization (IM)* as well as a description of the Linear Threshold (LT) and Independent Cascade (IC) diffusion models to which our algorithm is based.

According to Kempe et al. [29] and the spreading models they consider, the nodes are categorized in two states: the *active* and the *inactive* state. Initially a set of active nodes S is considered that trigger a spreading process that continues until no further activations are possible. The problem of influence maximization (IM) is to discover the called *seed set* S of size $k \ll |V|$ (k represents our budget) that maximizes the expected number of active nodes in the end of the process, defined as $\sigma(S)$.

With the IM problem being NP-hard [29], the majority of the literature provides approximate rather than the exact solutions. The most common approximate algorithms would be: i) *heuristic* and ii) *greedy* algorithms. An example of a heuristic algorithm would be to rank all the nodes according to a centrality measure and select the k top-ranked nodes.

The earliest greedy algorithm was provided by Kempe et al. [29], that approximates the optimal set within a factor of $(1 - 1/e)$. Initially, greedy algorithm assumes that the seed set is empty, $S = \emptyset$. Then at each round, it adds to the seed set the node with the maximum marginal gain. This procedure is repeated until the budget k is reached ($|S| = k$). Since the exact calculation of marginal gain is analytically intractable, it is approximated by executing multiple Monte Carlo (MC) simulation of the spreading process. Typically the number of simulations is set equal to 10.000. This constitutes a heavy burden to the greedy algorithm, making it not scalable to very large graphs.

Diffusion models

To simulate the process of information propagation in the network different diffusion models can be used. Next, we describe two of the most widely applied models, namely the LT and IC models. As we will present later on, the proposed algorithm is designed to deal with both of

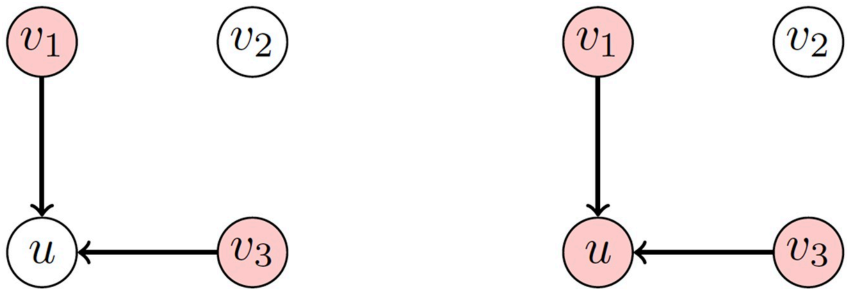

Fig 2. Illustration of the Linear Threshold model. The white colored nodes are in an inactive state whereas the pink colored nodes are in an active state. Node u will be activated if $p_{v_1,u}+p_{v_3,u} \geq \theta_u$.

https://doi.org/10.1371/journal.pone.0206318.g002

those models—contrary to most of the state-of-the-art algorithms which have been designed for one of the two models.

Linear Threshold (LT) model. In this model, a node u is influenced by each neighbor v according to a weight $p_{u,v}$. The value of this weight is such that the sum of all the weights towards all neighbors of v is less or equal to 1. Each node u chooses a threshold θ_u uniformly at random from the interval [0, 1] which represents the weighted fraction of u's neighbors that must become active in order for u to become active (an example is provided in Fig 2). Given a random choice of thresholds and an initial set of active nodes (with all other nodes being inactive), the diffusion process unfolds deterministically in discrete steps. A node u can be activated when the total cumulative weight of its active neighbors is at least θ_u.

Independent Cascade (IC) model. In the IC model, when a node u first becomes active in timestep t, it is given a single chance to activate each neighbor v—which is currently inactive—and succeeds with a probability $p_{u,v}$. If u succeeds, then v will become active in the next timestep. If u does not succeed, it cannot further attempt to activate v in future timesteps. The process continues as long as node activations are possible.

MATrix Influence (MATI) algorithm

In this section we introduce the proposed MATI algorithm, under both the LT and IC models. Based on our previous discussion for calculating influence in social networks (Section Preliminaries), we describe in detail how we compute the influence and we provide the algorithms proposed. Similar to the case of the greedy algorithm [29], at each round of MATI, the node with the largest marginal influence estimate is chosen as the next candidate. The novelty of our algorithm lies on the fact that, by having pre-calculated all possible paths between nodes along with the respective influences of the nodes -acting individually- in the network, we are able to efficiently compute the marginal gain of adding a candidate node.

Influence computation under the LT model

Kempe et al. [29] have shown the equivalence of the Linear Threshold model to the *live-edge model*. According to this model, a node $u \in V$ chooses just one of its incoming edges with probability $p_{u,v}$. If an edge is selected, it is considered *live*, otherwise *blocked*. We can deduce from the above that the nodes expected to be activated by a seed set S is the expected number of nodes that can be reached from S over all possible worlds ("All possible worlds" refers to all the different scenarios where different paths are considered live in the specific graph.). As it has been shown by Goyal et al. [28], the expected spread of seed set S can be calculated as follows:

$$\sigma(S) = \sum_{v\in V}\sum_{X} \Pr[X] I(S, v, X) = \sum_{v\in V}\mathcal{A}(S, v), \tag{2}$$

where X is a possible *live-edge* graph, $\Pr[X]$ is the sampling probability of graph X, $I(S, v, X)$ is an indicator function which equals to 1 if there exists a live path in X from S to v and 0 otherwise, and $\mathcal{A}(S, v)$ is the probability the single node v to be activated (influenced) by S. In the special case of a single node u, its expected spread to a node v ($u \neq v$) is defined as:

$$\begin{aligned}\mathcal{A}(u, v) &= \sum_{\psi_i\in\Psi(u,v)} \Pr[\psi_i] \\ &= \sum_{\psi_i\in\Psi(u,v)}\prod_{\ell=1}^{N-1} p^{\psi_i}_{\ell,\ell+1} \\ &= \sum_{\psi_i\in\Psi(u,v)} \phi_{iN} = \sum_{i=1}^{L}\phi_{iN},\end{aligned} \tag{3}$$

where $\Pr[\psi_i]$ is the probability of path ψ_i being *live* and $\Psi(u, v)$ is the set of all possible paths between nodes u and v. It becomes apparent that the influence of a node u to itself is equal to 1 (i.e., $\mathcal{A}(u, u) = 1$). According to the above, the influence of node u to node v in our example graph (see Fig 1 in Section Preliminaries) is equal to:

$$\mathcal{A}(u, v) = \sum_{i=1}^{2}\phi_{iN} = [\phi_{13}]_{\psi_1} + [\phi_{24}]_{\psi_2} = 0.034.$$

We can now define the expected total influence spread of a single node u to the network:

$$\sigma(u) = \sum_{v\in V}\mathcal{A}(u, v) \approx \sum_{v\in I(u)}\mathcal{A}(u, v), \tag{4}$$

where $I(u)$ represents the nodes in the graph that can be influenced by node u depending on a threshold θ which is set in order to limit the calculations of probability and cumulative probability paths. Roughly speaking, a node v belongs to set $I(u)$, *iff* $A(u, v) \geq \theta$. Therefore, the lower the parameter value θ is, the higher the accuracy that can be achieved.

The *forward cumulative influence* $\Omega(u, v)$ is another quantity of interest, that corresponds to the influence of node u to v and of node u to the nodes that can be found right after node v in the paths $\mathcal{T}(u)$ of node u. Algorithms 2 and 3 show how Ω is calculated.

Practically each $\Omega(u, v)$ element is calculated as the sum of i) the probabilities that all unique paths between nodes u and v are live and ii) the probabilities that all unique paths to nodes visited after node v while performing a Depth-first search starting from node u are live. In our example graph (see Fig 1 in Section Preliminaries) the aforementioned paths are the following: {u,a,v}, {u,a,b,v}, {u,a,v,c}, {u,a,v,c,f}, {u,a,v,c,e}, {u,a,v,d}, {u,a,v,d,e}, {u,a,b,v,c}, {u,a,b,v,c,f}, {u, a,b,v,c,e}, {u,a,b,v,d}, {u,a,b,v,d,e}. According to the above, the cumulative influence of node u to node v in our example graph is equal to: $\Omega(u, v)$ = 0.03 + 0.004 + 0.003 + 0.0009 + 0.0003 + 0.0045 + 0.0009 + 0.0004 + 0.00012 + 0.00004 + 0.0006 + 0.00012 = 0.04488.

Revisiting the case of a set of nodes, Goyal et al. [28] showed that the spread of a set S of nodes is the sum of the spread of each individual node $u \in S$ on the subgraphs induced by the set $V - S + u$:

$$\sigma(S) = \sum_{u\in S}\sigma^{V-S+u}(u), \tag{5}$$

where $\sigma^{V-S+u}(u)$ denotes the total influence of u in the subgraph induced by $V - S + u$. Similar to [28], we write $V - S$ to denote the difference of sets V and S, $V \setminus S$, and $V - S + u$ to denote $((V \setminus S) \cup \{u\})$.

By taking advantage of the $\mathcal{A}(u, v)$ (Eq 5) and $\Omega(u, v)$ definitions, we get the following key result that helps towards the calculation of the influence gain after the addition of a node x to a set of nodes S. This result constitutes the basis of the proposed MATI algorithm under the LT diffusion model.

Theorem 1. *Under the LT model, to calculate the influence after adding a node x to a set of nodes S, one has to subtract from the sum of the individual spread of S and x, the forward cumulative influence* Ω *of all the nodes that belong to set S, which contain node x in paths connecting the latter to nodes in set S. That is,*

$$\sigma(S \cup \{x\}) = \sigma(S) + \sigma(x) - \sum_{y \in S} \Omega(x, y) - \sum_{y \in S} \Omega(y, x). \tag{6}$$

Proof.

$$\begin{aligned}
\sigma(S \cup \{x\}) &\overset{(1)}{=} \sum_{u \in S+x} \sigma^{V-S-x+u}(u) \\
&\overset{(2)}{=} \sigma^{V-S}(x) + \sum_{u \in S} \sigma^{V-S-x+u}(u) \\
&\overset{(3)}{=} \sigma^{V-S}(x) + \sigma^{V-x}(S) \\
&\overset{(4)}{=} \sigma(x) + \sigma(S) - \sum_{y \in S} \Omega(x, y) - \sum_{y \in S} \Omega(y, x).
\end{aligned}$$

Equality (1) is a direct application of Eq 5 (see [28] for its proof). Equalities (2) and (3) can easily be verified by making simple calculations. Finally, equality (4) comes from set theory (see Fig 3 for an illustration). Roughly speaking, equality (4) expresses that the influence gain after adding node x to a set S, is equal to the summation of the influence gain of x and S independently, subtracting from this value the nodes that can be influenced by x or S, through

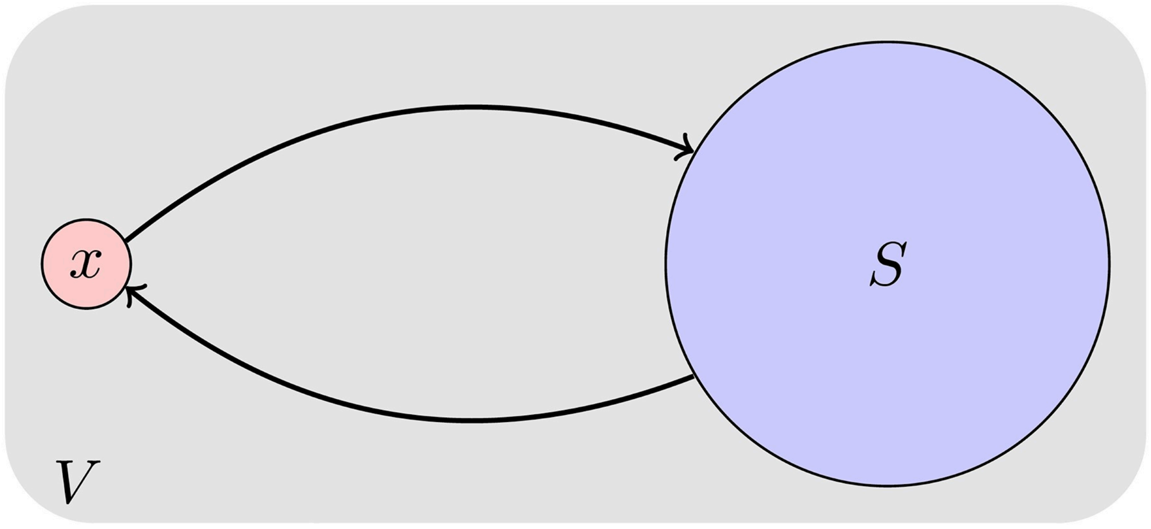

Fig 3. Illustration of Theorem 1.

https://doi.org/10.1371/journal.pone.0206318.g003

paths that pass via nodes on set S or node x, respectively. For instance, if both x and S influence nodes that do not cross each other, we get that $\sigma(S \cup \{x\}) = \sigma(x) + \sigma(S)$.

Algorithms 1 to 4 show the complete structure of MATI algorithm under the LT model. Routine CalcStatsLT (Algorithm 2) computes $\mathcal{A}$ and Ω, and routine CalcInf (Algorithm 4) returns the influence of all nodes $v \in V$, as was described in this section. We use a CELF queue which is a queue storing the nodes' marginal gains in decreasing order, as in [27]. At each iteration, we add the top node of the CELF queue at the seed set, until the budget k is reached (see Algorithm 1). The influence gain of every node to be selected is calculated by subtracting the respective influence for which the candidates selected so far are responsible for, through common paths as shown in Theorem 1.

Algorithm 1 MatiLT

```
Input: G = (V, E), k                    ▷ k: budget (number of seed nodes)
Initialize: S = ∅
$\mathcal{A}, \Omega$ = CalcStatsLT(G)
Q = CalcInf($\mathcal{A}$, V)
for i = 1 to k do
  s, σ(s) = Q.top()
  S = S ∪ s
  U = V\S
  for each u ∈ U do
    σ(u) = Q(u)
    for each v ∈ S do
      σ(u) -= Ω(v, u)
      σ(u) -= Ω(u, v)
    end for
    Q.add((u, σ(u)))
  end for
end for
return S
```

Algorithm 2 CalcStatsLT

```
Input: G = (V, E)
Initialize: $\mathcal{A}, \Omega = 0$
for each u ∈ V do
  A(u, u) = 1
  A(u,:), Ω(u,:),_ = DFStatistics(G, u, 1, A(u,:), Ω(u,:))
end for
return $\mathcal{A}, \Omega$
```

Algorithm 3 DFStatistics

```
Input: G = (V, E), u, $p_u$, $A_r$, $\Omega_r$
Initialize: $\Omega_{temp} = p_u$
for each w ∈ Neighbors(u) do
  $p_w = p_{u,w} * p_u$
  $A_r(w)$ += $p_w$
  $A_r, \Omega_r, \Omega'_{temp} = DFStatistics(G, w, p_w, A_r, \Omega_r)$
  $\Omega_{temp} += \Omega'_{temp}$
end for
$\Omega_r(u)$ += $\Omega_{temp}$
return $A_r$, $\Omega_r$, $\Omega_{temp}$
```

Algorithm 4 CalcInf

```
Input: $\mathcal{A}, V$
Initialize: Q = ∅; σ(u) = 0, $\forall_u$ ∈ V
for each u ∈ V do
  for each v ∈ I(u) do                    ▷ I(u): nodes influenced by u
    $\sigma(u) += \mathcal{A}(u, v)$
  end for
```

```
7:    $Q.add((u, \sigma(u)))$
8: end for
9: return $Q$
```

Influence computation under the IC model

In the IC diffusion model, the activation probability of a node to another one in a path can be calculated by multiplying the influence weights $p^{\tau}_{\ell,\ell+1}$ leading to it in path ψ_i. That is, in a path $\psi_i = \{n_{i1} = u, n_{i2}, \ldots, n_{iN} = v\}$, the influence of node u to node v can be calculated as follows:

$$\mathcal{A}_{\psi_i}(u,v) = \prod_{\ell=1}^{N-1} p^{\psi_i}_{\ell,\ell+1} = \phi_{iN}.$$

The total activation probability of node v from node u, while taking into consideration the L different (unique) paths $\Psi(u, v) = \{\psi_1, \psi_2, \ldots, \psi_L\}$ that lead from u to v, can be computed as:

$$\mathcal{A}(u,v) = 1 - \prod_{\psi_i \in \Psi(u,v)} (1 - \sigma_{\psi_i}(u,v)) = 1 - \prod_{i=1}^{L}(1 - \phi_{iN}),$$

where $\prod_{i=1}^{L}(1 - \phi_{iN})$ is equal to the probability that node u does not influence v (i.e., none of the paths from u to v is active).

In the case of the IC model, $\Omega(u, v)$ cannot be calculated only according to the influence weights $p_{k,k+1}$ in a specific path. In fact, the calculation of the influence in the case of the addition of a node u will change the so far calculated influence of a set of seed nodes S. Therefore, we use the following heuristics to compute the additional influence of node u, i.e., $\sigma(S \cup \{u\})$.

$\sigma(S + u)$ computation.

1. For every path originating from node u (i.e., $\mathcal{T}(u)$) or a node belonging to seed set S, we keep the subpaths before falling into a node belonging to $S \cup \{u\}$.
2. The $\sigma(S + u)$ is equal to the sum of the influence probabilies that correspond to each of these subpaths.

Algorithm 5 MatiIC

```
1: Input: $G = (V, E)$, $k$                    ⊳ $k$: budget (number of seed nodes)
2: Initialize: $S = \emptyset$, $\sigma(S) = 0$
3: $\mathcal{A} = \text{CalcStatsIC}(G)$
4: $Q = \text{CalcInf}(\mathcal{A}, V)$
5: for $i = 1$ to $k$ do
6:    $s, \sigma(s) = Q.top()$
7:    $S = S \cup s$
8:    $U = V \setminus S$
9:    $\sigma(S) = \sigma(S) + \sigma(s)$
10:   for each $u \in U$ do
11:      $\sigma(S \cup \{u\}) = |S \cup \{u\}|$
12:      $\sigma(S \cup \{u\})$ += $\text{AdditiveInf}(\mathcal{T}(u), \mathcal{F}(u), S)$
13:      $\sigma(S \cup \{u\})$ += $\text{AdditiveInf}(\mathcal{T}(S), \mathcal{F}(S), S \cup \{u\})$
14:      $Q.add((u, \sigma(S \cup \{u\}) - \sigma(S)))$          ⊳ Order is maintained
15:   end for
16: end for
17: return $S$
```

Algorithm 6 CalcStatsIC

```
1: Input: $G = (V, E)$
2: Initiaze: $\mathcal{A} = 0$
3: for each $u \in V$ do
```

```
4:   Generate $\mathcal{T}(u)$ and $\mathcal{F}(\tau_i)$, $\forall\ \tau_i$ using DFS
5:   for each $v \in V$ do
6:     $pr = 1$
7:     Generate $\Psi(u, v)$ and $\Phi(\psi_i)$, $\forall \psi_i$ (based on $\mathcal{T}(u)$)
8:     for each $\psi_i \in \Psi(u, v)$ do
9:       $pr = pr * (1 - \phi_{ij})$
10:     end for
11:     $\mathcal{A}(u, v) = 1 - pr$
12:   end for
13: end for
14: return $\mathcal{A}$
```

Algorithm 7 AdditiveInf

```
1: Input: $\mathcal{T}, \mathcal{F}, S$                ▷ $\mathcal{T}$: set of paths, $S$: set of nodes
2: Initialize: $i = 0$; $inf = 0$
3: for each $\tau \in \mathcal{T}$ do
4:   $i = i + 1$
5:   for each $u \in \tau$ do
6:     $j \leftarrow index(u)$
7:     if $j == 1$ then
8:       continue
9:     else if $u \notin S$ then
10:       $inf \mathrel{+}= f_{ij}$
11:     else
12:       break
13:     end if
14:   end for
15: end for
16: return $inf$
```

Algorithm 5 shows the structure of the MATI algorithm under the IC model. Initially, routines CalcStatsIC (Algorithm 6) and CalcInf (Algorithm 4) are called to compute $\mathcal{A}$ and the contents of CELF queue Q, respectively. While calculating the marginal influence for every candidate node u, routine AdditiveInf (Algorithm 7) computes the additional influence as previously described (see $\sigma(S \cup \{u\})$ computation).

Experiments

We have conducted experiments in real-world datasets in order to evaluate the performance of the MATI algorithm and compare it to state-of-the-art influence maximization algorithms on the quality of results and efficiency. The algorithm has been implemented in Python and all experiments are run on a Linux machine with a 3.00GHz CPU Intel Xeon CPU and 64GB memory.

Datasets

We have used four publicly available graph datasets [41] whose statistics are summarized in Table 2. To generate influence weights on all edges, we adopt the classical uniform method by [29, 42]. More precisely, we set the weight of every incoming edge of a node v to be equal to $\frac{1}{d_v}$, where d_v is the in-degree of node v. It has to be noted that the datasets were transformed from

Table 2. Properties of the real-world graphs used.

Dataset	NetHEPT	WikiVote	Epinions	Email-EuAll
# Nodes	15K	7K	75K	225K
# Edges	62K	103K	405K	341K

https://doi.org/10.1371/journal.pone.0206318.t002 an undirected format to a directed one by simply assuming that if an edge between two nodes exists the one can influence the other. Thus, the number of the edges used in the experiments is twice the one that is reported.

- NetHEPT: This is a collaboration network taken from the "High Energy Physics (Theory)" section of http://arxiv.org, with nodes representing authors and edges capturing co-authorship relationships. Here, a user publishing a paper is considered as an action.
- WikiVote: This network has been created after extracting all administrator elections and vote history data from Wikipedia, the free encyclopedia written by peers all around the world. It contains data from 2, 794 elections with 103, 663 total votes and 7, 066 users dated from the inception of Wikipedia until January 2008.
- Epinions: This is a who-trust-whom online social network of the general consumer review site *Epinions* (*epinions.com*). Each member decides whether to trust or not the other members. From the trust relationships that are created together with the review ratings provided by the users, it is determined which reviews to be shown to each user.
- Email-EuAll: This network has been generated using email data from a large European research institution for a period from October 2003 to May 2005 (18 months). Given a set of email messages, each node corresponds to an email address. We create a directed edge between nodes i and j, if i sent at least one message to j.

Baseline algorithms

In order to evaluate the performance of the MATI algorithm, we compare its performance to those of eight baseline algorithms which are described below:

- **Random**: This heuristic chooses k random nodes. The behavior depicted in the figures is the average behavior after 100 different random picks.
- **Degree**: A heuristic based on the concept of "degree centrality", considering high-degree nodes as influential commonly used in sociology literature [43]. This heuristic chooses k nodes in decreasing *out-degree* order.
- **Harmonic**: This heuristic is based on the concept of *harmonic centrality* [44] of a node v, which can be calculated as follows:

$$C(v) = \sum_{u \neq v} \frac{1}{d(u, v)}$$

  where $d(u, v)$ indicates the shortest path between the nodes u and v. The k nodes are chosen in decreasing harmonic centrality order.
- **Greedy**: The original greedy algorithm with Monte-Carlo Simulations as described in Section The Influence Maximization (IM) problem. Following the literature [29], we run 10, 000 Monte Carlo (MC) simulations to estimate the spread of any seed set.
- **LDAG**: This algorithm constructs a local direct acyclic graph (DAG) around every node v and restricts the calculation of the respective influence inside this local area [24]. The nodes added to the aforementioned DAG of node v depend on a threshold θ. The individual influence of the nodes added to node v has to be larger than this threshold. Here we set θ to $\frac{1}{320}$ as used by the authors. The algorithm is tailored for the Linear Threshold diffusion model only.

- **SimPath**: The algorithm computes the influence spread of the nodes by exploring simple paths in their neighborhood [28]. A prunning threshold η is used to balance running time and quality of the influence spread of the seed nodes. Here, the pruning threshold is set to 10^{-3} and the look-ahead value l is set to 4, as proposed by the authors.
- **IMM**: The **I**nfluence **M**aximization via **M**artingales algorithm adopts a two-phase paradigm which includes *parameter estimation* and *node selection*. In the parameter estimation phase of IMM, the martingales statistical approach is employed to derive the number θ of reverse reachable (RR) sets to be chosen before the seed nodes are selected. A $(1 - 1/e - \epsilon)$-approximation is achieved with ϵ being a tunable parameter. In the experiments we use $\epsilon = 0.1$. The algorithm is defined for both the Linear Threshold and Independent Cascade models.

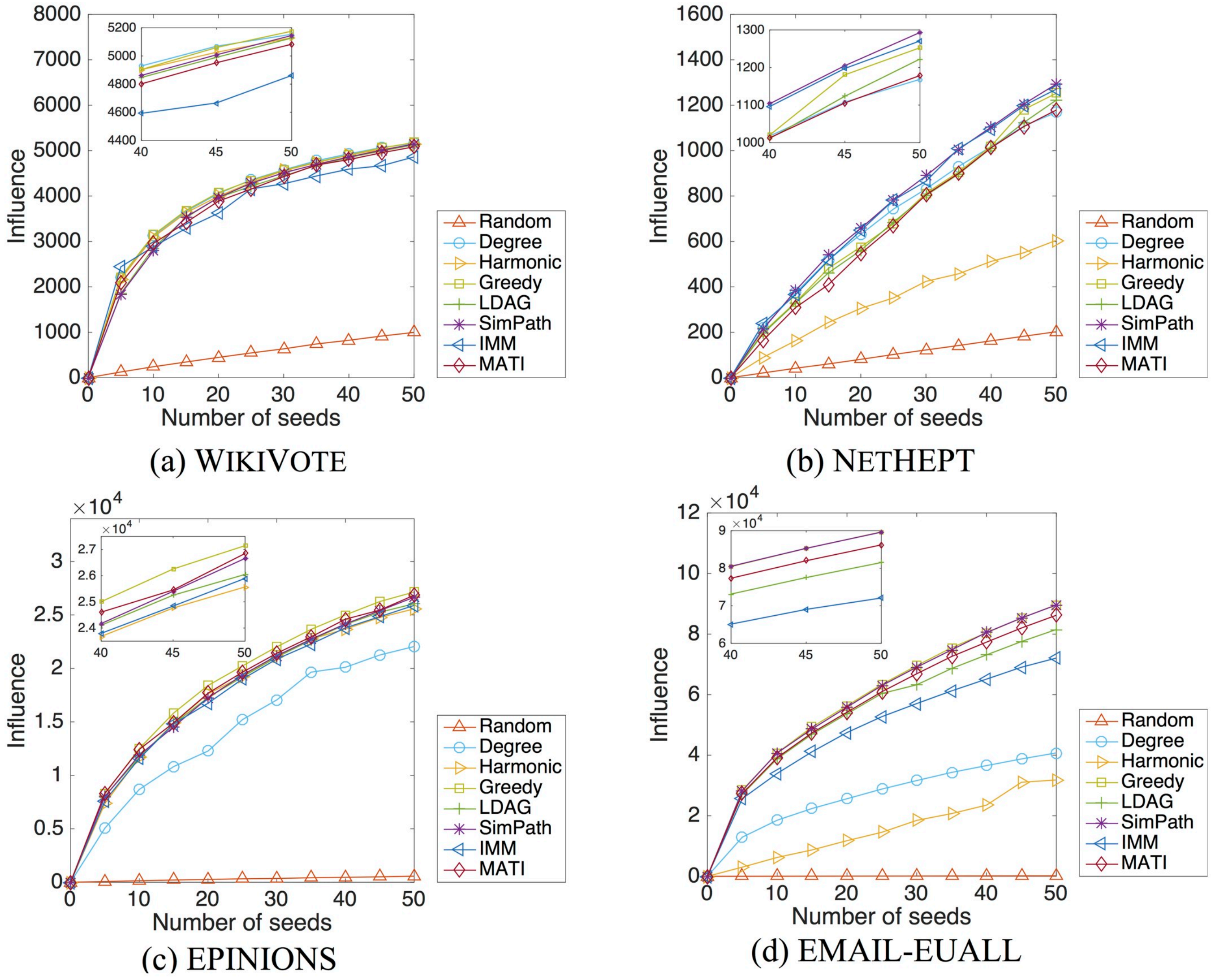

Fig 4. Influence spread in number of nodes for the different algorithms, under the LT model. We show results for the following networks: (a)WikiVote; (b) NetHEPT; (c)Epinions; (d)Email-EuAll. Each plot depicts the influence in number of nodes achieved by the different methods: Random, Degree, Harmonic, Greedy, LDAG, SimPath, IMM and MATI. Each point shows the number of nodes that the respective number of seed nodes—given by the different methods—achieve to influence.

https://doi.org/10.1371/journal.pone.0206318.g004

- **SKIM**: The **SK**etch-based **I**nfluence **M**aximization (SKIM) algorithm uses per-node summary structures called *combined reachability sketches* that represent the node's influence coverage across a number l of graph instances. The combined reachability sketch is the bottom-k min-hash sketch of the combined reachability sketch of the node. Parameter k works as a tradeoff between computation and accuracy. In the experiments we use $l = 64$ and $k = 64$. The algorithm is tailored for the Independent Cascade model only.

Unless noted otherwise, the threshold θ for the proposed MATI algorithm is set to 0.0001. The value was chosen experimentally, based on performance observation.

Experimental results

We compare the performance of the aforementioned algorithms, with respect to the quality of seed sets and efficiency aspects.

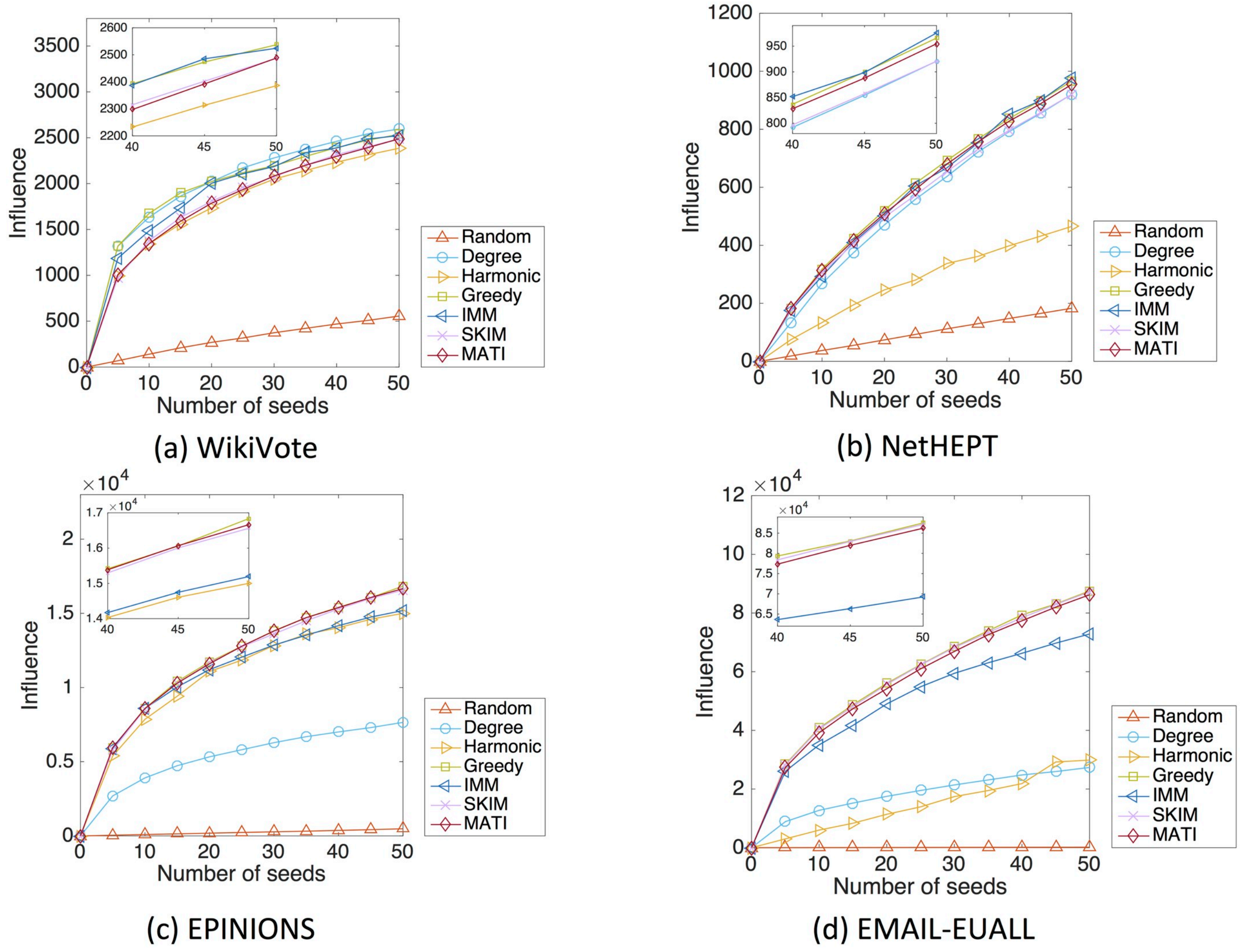

Fig 5. Influence spread in number of nodes for the different algorithms, under the IC model. We show results for the following networks: (a)WikiVote; (b) NetHEPT; (c)Epinions; (d)Email-EuAll. Each plot depicts the influence in number of nodes achieved by the different methods: Random, Degree, Harmonic, Greedy, IMM, SKIM and MATI. Each point shows the number of nodes that the respective number of seed nodes—given by the different methods—achieve to influence.

https://doi.org/10.1371/journal.pone.0206318.g005

Table 3. Comparison of running times in seconds and influence spread in number of nodes for different values of the parameter θ under (a) the LT and (b) the IC model for the NetHEPT network.

θ	Run. Time (s)	Influence	θ	Run. Time (s)	Influence
0.1	1.2	984.7	0.1	1.2	867.77
0.01	6.5	1162.3	0.01	6.8	959.42
0.001	88.6	1209.6	0.001	106.5	938.21
0.0001	708.2	1190.8	0.0001	820.6	950.32
(a)			(b)		

https://doi.org/10.1371/journal.pone.0206318.t003

Quality of seed sets. The quality of the seed sets obtained by different algorithms is evaluated based on the expected spread of influence measured in number of nodes. Figs 4 and 5 show the spread of influence versus the size of seed set, under the LT and IC models respectively.

Under the LT model, the seed sets obtained via MATI are quite competitive in quality compared to those of the Greedy, LDAG, SimPath and SKIM algorithms. For all four datasets, the influence loss for up to 50 seeds is less than 2% with respect to the Greedy algorithm. Under the IC model, our algorithm is still effective, despite the heuristics involved in the influence estimation.

We observe that for both LT and IC models the Random heuristic fails to identify good candidates. On the other hand, the Degree and Harmonic heuristics reveal seed nodes that are actually good candidates for small networks such as WikiVote and NetHEPT but fail to reveal effective seed nodes for larger networks such as Epinions and Email-EuAll. This is due to the fact that the aforementioned heuristics do not take into account that some of the chosen nodes are clustered together and the overlapping influence some nodes occur reduces their final total influence.

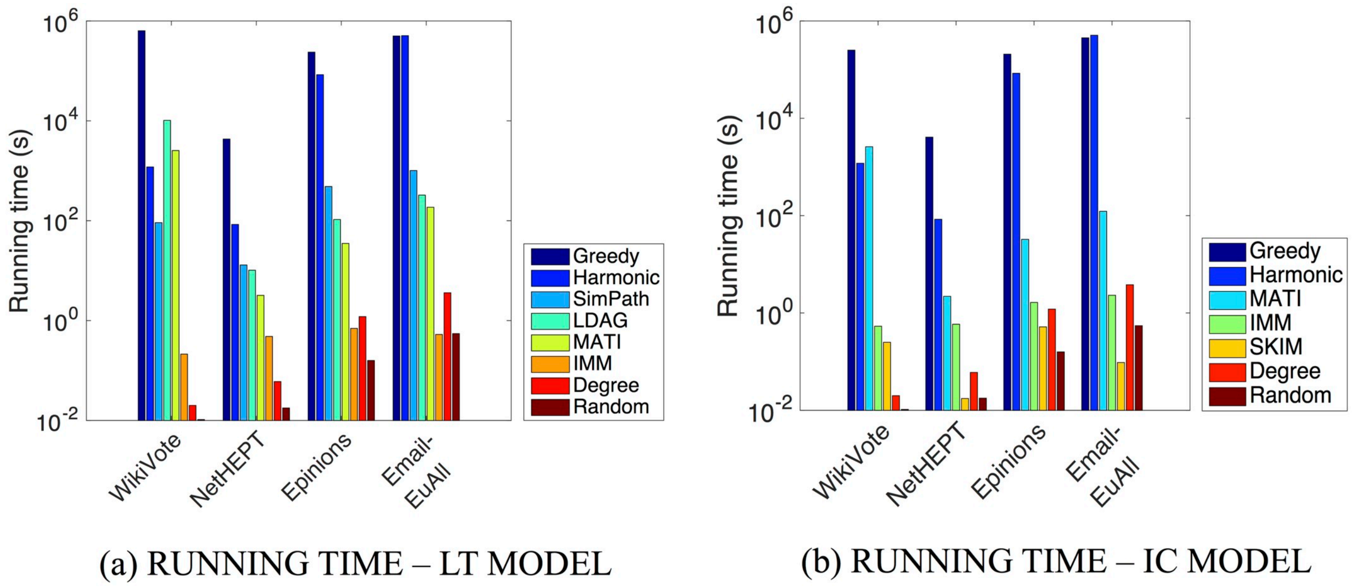

Fig 6. Comparison of running times in seconds of the different algorithms under the (a) LT and (b) IC models. We show results for the following networks: WikiVote; NetHEPT; Epinions; Email-EuAll. Each plot depicts the running time in seconds that each different algorithm requires to produce a group of $k = 50$ seed nodes. Results for the following methods are shown: Random, Harmonic, Degree, Greedy, LDAG, SimPath, IMM, SKIM and MATI.

https://doi.org/10.1371/journal.pone.0206318.g006

We have also performed experiments for different values of the parameter θ of our algorithm for the NetHEPT dataset, in order to observe the running time of our algorithm with respect to the influence that is achieved. The results are depicted in Table 3. As θ decreases, the running time is always increasing. This is justified by the fact that a smaller θ allows computation of influence in a greater neighborhood around each node. In most of the cases, the influence achieved also increases. This can be explained by the fact that the formation of paths of greater length provide a more accurate computation of a node's influence.

Efficiency of MATI. We have also examined the running time of the proposed algorithm. Fig 6 reports the execution time required by various algorithms for the LT and IC models respectively. The figures have a logarithmic scale on the y-axis. In all cases, MATI is faster than the Greedy and LDAG algorithms. In all datasets except WikiVote, MATI also performs better than SimPath. It takes Greedy more than one week to select 50 seed nodes for datasets such as Epinions. The IMM and SKIM algorithms seem to be slightly faster than our algorithm and this is due to the initial computation of the large data structures described in Section MATrix Influence (MATI) algorithm. The latter computations happen once and despite their

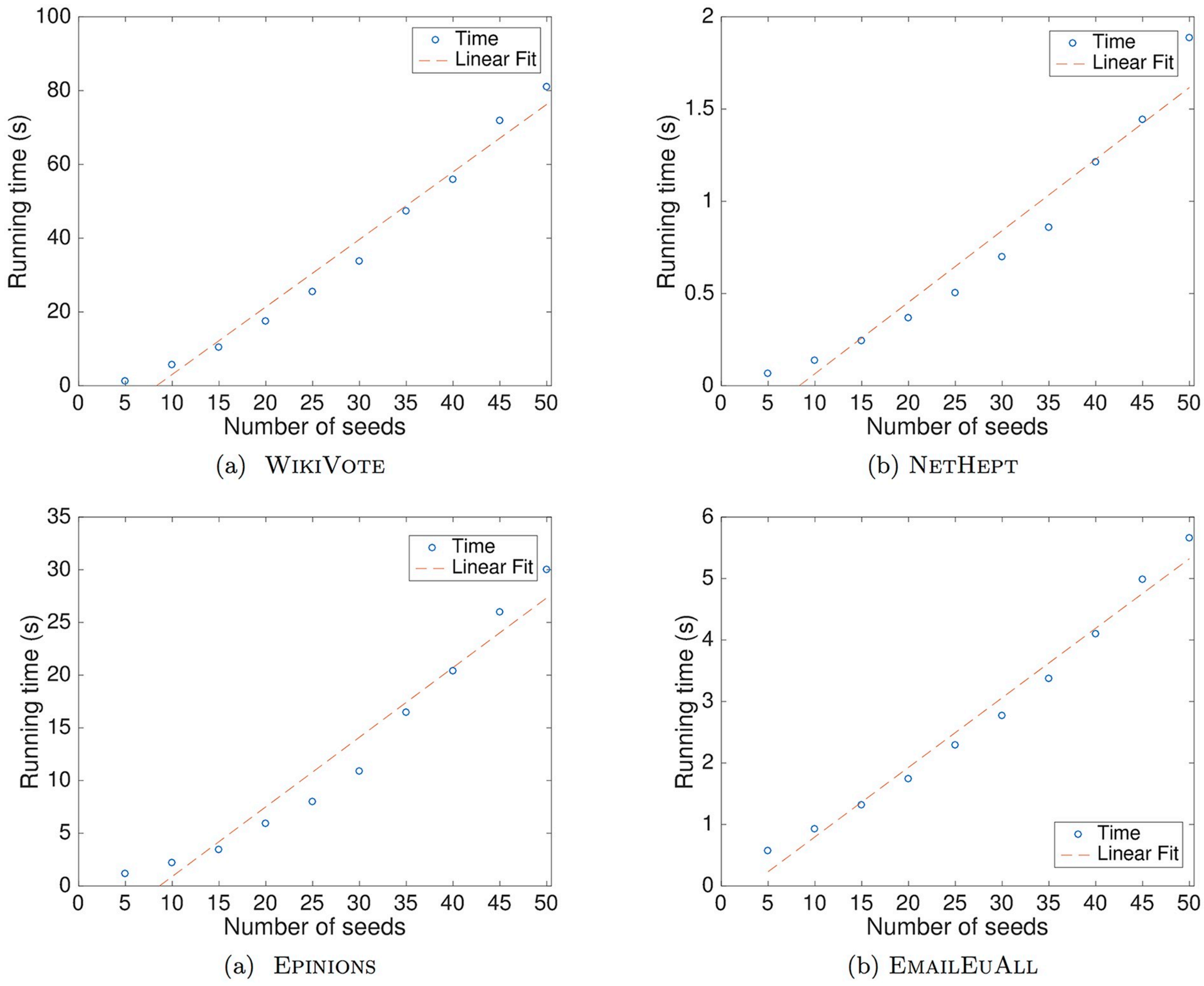

Fig 7. Computation times in seconds of the MATI algorithm for different numbers of seed nodes under the LT model. We show results for the following networks: (a)WikiVote; (b)NetHEPT; (c)Epinions; (d)Email-EuAll. Each plot depicts the running time in seconds that MATI requires to produce a ranging number of seed nodes. We observe that the computation time is close to linear with respect to the number of seed nodes that the algorithm identifies as influential.

https://doi.org/10.1371/journal.pone.0206318.g007

"expensiveness" they can facilitate the re-computation of the node's influence in case there is a node deletion in the network. Nevertheless MATI remains competitive in terms of running time and it will be also competitive in the case of a possible change in the network as mentioned whereas this is not guaranteed for the other baselines. Additionally, MATI's running time can always be reduced by increasing parameter θ which will not result in a significant decrease in the influence spread (see Table 3). It is also worth noticing that even IMM is faster, the influence spread given by the seed nodes proposed is lower than our algorithm's in some datasets. Specifically we can observe the difference between the influence spread of IMM and MATI in the case of the WikiVote and Email-EuAll datasets for the LT model and that of Epinions and Email-EuAll datasets for the IC model (see Figs 4 and 5 respectively). SKIM is also competitive in terms of running time but is only designed for the Independent Cascade model while MATI is designed for both aforementioned diffusion models. Finally we observe that although the Degree and Random heuristics are time efficient, they fail to output a seed set of high quality.

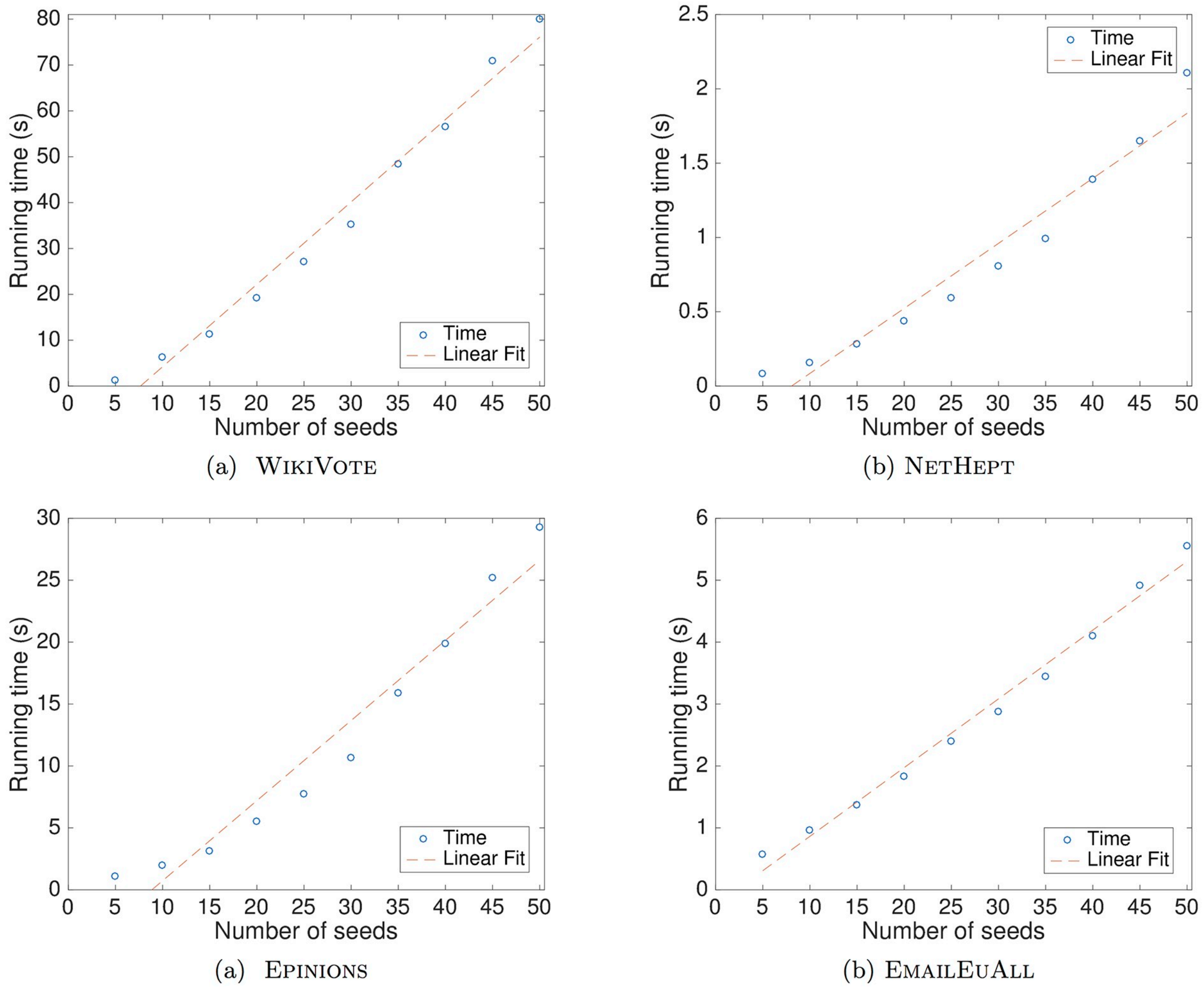

Fig 8. Computation times in seconds of the MATI algorithm for different numbers of seed nodes under the IC model. We show results for the following networks: (a)WikiVote; (b)NetHEPT; (c)Epinions; (d)Email-EuAll. Each plot depicts the running time in seconds that MATI requires to produce a ranging number of seed nodes. We observe that the computation time is close to linear with respect to the number of seed nodes that the algorithm identifies as influential.

https://doi.org/10.1371/journal.pone.0206318.g008

In Figs 7 and 8, the running times that the MATI algorithm requires to identify a ranging number of seed nodes are depicted. By using the least squares regression method [45], we observe that the computation time is linear with respect to the number of influential nodes that are identified.

Conclusion

In this paper we introduced an efficient algorithm for influence maximization in social networks, under both the LT and IC diffusion models. Even though the greedy algorithm has been proven to produce the best results so far in terms of quality of seeds, it does not scale to large networks.

MATI takes into consideration the possible paths that are created in each node's neighborhood and pre-calculates the nodes' influences. Our algorithm depends on the creation of large data structures which increases memory consumption, but at the same time facilitates the re-computation of the nodes' influence in the case of a node or edge deletion in the network. After performing extensive experiments in four datasets, we have shown that MATI is competitive regarding both the quality of seeds and the running time when compared to state-of-the-art algorithms. We plan to further evaluate our algorithm by doing more experiments with larger datasets and comparing it with more baseline methods. Additionally, we will be experimenting with heuristics that can further speed up the running time and efficient data structures that may reduce the memory consumption of our algorithm. Finally, we will be studying how our method can be extended for the case of a dynamic graph where nodes and edges are added or deleted from the network.

Author Contributions

Conceptualization: Maria-Evgenia G. Rossi, Bowen Shi, Nikolaos Tziortziotis.

Data curation: Maria-Evgenia G. Rossi, Christos Giatsidis.

Software: Maria-Evgenia G. Rossi, Bowen Shi, Christos Giatsidis.

Supervision: Nikolaos Tziortziotis, Fragkiskos D. Malliaros, Michalis Vazirgiannis.

Validation: Nikolaos Tziortziotis.

Writing – original draft: Maria-Evgenia G. Rossi.

Writing – review & editing: Nikolaos Tziortziotis, Fragkiskos D. Malliaros, Michalis Vazirgiannis.

References

1. Hoppenstaedt F. Mathematical theories of populations: demographics, genetics and epidemics. Society for industrial and applied mathematics, 1975.
2. Kermack WO, McKendrick AG. Contributions to the mathematical theory of epidemics—II. The problem of endemicity. Bulletin of mathematical biology 53.1-2 (1991): 57–87.
3. Van der Plank JE. Plant diseases: epidemics and control. Elsevier, 2013.
4. Leskovec J, Adamic LA, Huberman BA. The dynamics of viral marketing. ACM Transactions on the Web (TWEB) 1.1 (2007): 5. https://doi.org/10.1145/1232722.1232727
5. Zeng D, Chen H, Lusch R, Li SH. Social media analytics and intelligence. IEEE Intelligent Systems 25.6 (2010): 13–16. https://doi.org/10.1109/MIS.2010.151
6. Moreno Y, Nekovee M, Pacheco AF. Dynamics of rumor spreading in complex networks. Physical Review E 69.6 (2004): 066130. https://doi.org/10.1103/PhysRevE.69.066130

7. Balog K, Azzopardi L, De Rijke M. Formal models for expert finding in enterprise corpora. Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2006.

8. Hsu WH, King AL, Paradesi MS, Pydimarri T, Weninger T. Collaborative and Structural Recommendation of Friends using Weblog-based Social Network Analysis. AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs. Vol. 6. 2006.

9. Liben Nowell D, Kleinberg J. The link prediction problem for social networks. Journal of the Association for Information Science and Technology 58.7 (2007): 1019–1031.

10. Taskar B, Wong MF, Abbeel P, Koller D. Link prediction in relational data. Advances in neural information processing systems. 2004.

11. Cohen R, Havlin S, Ben-Avraham D. Efficient immunization strategies for computer networks and populations. Physical review letters 91.24 (2003): 247901. https://doi.org/10.1103/PhysRevLett.91.247901

12. Pastor-Satorras R, Vespignani A. Immunization of complex networks. Physical Review E 65.3 (2002): 036104. https://doi.org/10.1103/PhysRevE.65.036104

13. Ding Y, Yan E, Frazho A, Caverlee J. PageRank for ranking authors in co citation networks. Journal of the Association for Information Science and Technology 60.11 (2009): 2229–2243.

14. Radicchi F, Fortunato S, Markines B, Vespignani A. Diffusion of scientific credits and the ranking of scientists. Physical Review E 80.5 (2009): 056103. https://doi.org/10.1103/PhysRevE.80.056103

15. Zhou YB, Lü L, Li M. Quantifying the influence of scientists and their publications: distinguishing between prestige and popularity. New Journal of Physics 14.3 (2012): 033033. https://doi.org/10.1088/1367-2630/14/3/033033

16. Lü L, Medo M, Yeung CH, Zhang YC, Zhang ZK, Zhou T. Recommender systems. Physics Reports 519.1 (2012): 1–49. https://doi.org/10.1016/j.physrep.2012.02.006

17. Trusov M, Bucklin RE, Pauwels K. Effects of word-of-mouth versus traditional marketing: findings from an internet social networking site. Journal of marketing 73.5 (2009): 90–102. https://doi.org/10.1509/jmkg.73.5.90

18. Domingos P, Richardson M. Mining the network value of customers. Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2001.

19. Lü L, Chen D, Ren XL, Zhang QM, Zhang YC, Zhou T. Vital nodes identification in complex networks. Physics Reports 650 (2016): 1–63. https://doi.org/10.1016/j.physrep.2016.06.007

20. Kitsak M, Gallos LK, Havlin S, Liljeros F, Muchnik L, Stanley HE, Makse HA. Identification of influential spreaders in complex networks. Nature physics 6.11 (2010): 888–893. https://doi.org/10.1038/nphys1746

21. Lü L, Zhang YC, Yeung CH, Zhou T. Leaders in social networks, the delicious case. PloS one 6.6 (2011): e21202. https://doi.org/10.1371/journal.pone.0021202

22. Malliaros FD, Rossi MEG, Vazirgiannis M. Locating influential nodes in complex networks. Scientific reports 6 (2016). https://doi.org/10.1038/srep19307

23. Rossi MEG, Malliaros FD, Vazirgiannis M. Spread it good, spread it fast: Identification of influential nodes in social networks. Proceedings of the 24th International Conference on World Wide Web (pp. 101-102). ACM.

24. Chen W, Yuan Y, Zhang L. Scalable influence maximization in social networks under the linear threshold model. In Data Mining (ICDM), 2010 IEEE 10th International Conference on (pp. 88-97). IEEE.

25. Chen W, Wang C, Wang Y. Scalable influence maximization for prevalent viral marketing in large-scale social networks. Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2010.

26. Cohen E, Delling D, Pajor T, Werneck RF. Sketch-based influence maximization and computation: Scaling up with guarantees. In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (pp. 629-638). ACM.

27. Goyal A, Lu W, Lakshmanan LV. Celf++: optimizing the greedy algorithm for influence maximization in social networks. Proceedings of the 20th international conference companion on World wide web. ACM, 2011.

28. Goyal A, Lu W, Lakshmanan LV. Simpath: An efficient algorithm for influence maximization under the linear threshold model. In Data Mining (ICDM), 2011 IEEE 11th International Conference on (pp. 211-220). IEEE.

29. Kempe D, Kleinberg J, Tardos É. Maximizing the spread of influence through a social network. Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2003.

30. Leskovec J, Krause A, Guestrin C, Faloutsos C, VanBriesen J, Glance N. Cost-effective outbreak detection in networks. Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2007.

31. Chen W, Wang Y, Yang S. Efficient influence maximization in social networks. Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 199-208). ACM.

32. Jiang Q, Song G, Cong G, Wang Y, Si W, Xie K. Simulated Annealing Based Influence Maximization in Social Networks. AAAI (Vol. 11, pp. 127-132).

33. Jung K, Heo W, Chen W. Irie: Scalable and robust influence maximization in social networks. Data Mining (ICDM), 2012 IEEE 12th International Conference on (pp. 918-923). IEEE.

34. Ohsaka N, Akiba T, Yoshida Y, Kawarabayashi KI. Fast and Accurate Influence Maximization on Large Networks with Pruned Monte-Carlo Simulations. AAAI (pp. 138-144).

35. Tang Y, Xiao X, Shi Y. Influence maximization: Near-optimal time complexity meets practical efficiency. In Proceedings of the 2014 ACM SIGMOD international conference on Management of data (pp. 75-86). ACM.

36. Borgs C, Brautbar M, Chayes J, Lucier B. Maximizing social influence in nearly optimal time. Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (pp. 946-957). Society for Industrial and Applied Mathematics.

37. Tang Y, Shi Y, Xiao X. Influence maximization in near-linear time: A martingale approach. Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM, 2015.

38. Williams D. Probability with martingales. Cambridge university press, 1991.

39. Cohen E. Size-Estimation Framework with Applications to Transitive Closure and Reachability. Journal of Computer and System Sciences 55.3 (1997): 441–453. https://doi.org/10.1006/jcss.1997.1534

40. Goyal A, Bonchi F, Lakshmanan LV. A data-based approach to social influence maximization. Proceedings of the VLDB Endowment 5.1 (2011): 73–84. https://doi.org/10.14778/2047485.2047492

41. Leskovec J, Krevl A. SNAP Datasets: Stanford Large Network Dataset Collection.

42. Goyal A, Bonchi F, Lakshmanan LV. Learning Influence Probabilities in Social Networks. Proceedings of the third ACM international conference on Web search and data mining (pp. 241-250). ACM.

43. Scott John Social network analysis. Sage,2017.

44. Boldi P, Vigna S. Axioms for centrality. Internet Mathematics 10.3-4 (2014): 222–262. https://doi.org/10.1080/15427951.2013.865686

45. Bishop C M. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag (2006).