Influence Maximization with Spontaneous User Adoption

Anonymous Author(s)

ABSTRACT

We incorporate the realistic scenario of spontaneous user adoption into influence propagation (also refer to as self-activation) and propose the self-activation independent cascade (SAIC) model: nodes may be self activated besides being selected as seeds, and influence propagates from both selected seeds and self activated nodes. Self activation occurs in many real world situations; for example, people naturally share product recommendations with their friends, even without marketing intervention. Under the SAIC model, we study three influence maximization problems: (a) boosted influence maximization (BIM) aims to maximize the total influence spread from both self-activated nodes and k selected seeds; (b) preemptive influence maximization (PIM) aims to find k nodes that, if self-activated, can reach the most number of nodes before other self-activated nodes; and (c) boosted preemptive influence maximization (BPIM) aims to select k seed that are guaranteed to be activated and can reach the most number of nodes before other self-activated nodes. We propose scalable algorithms for all three problems and prove that they achieve $1 - 1/e - \varepsilon$ approximation for BIM and BPIM and $1 - \varepsilon$ for PIM, for any $\varepsilon > 0$. Through extensive tests on real-world graphs, we demonstrate that our algorithms outperform the baseline algorithms significantly for the PIM problem in solution quality, and also outperform the baselines for BIM and BPIM when self-activation behaviors are nonuniform across nodes.

CCS CONCEPTS

• **Information systems** → *Social advertising*; *Social networks*; • **Theory of computation** → *Probabilistic computation*; *Submodular optimization and polymatroids*;

KEYWORDS

influence maximization, greedy algorithm, self activation, reverse influence sampling, preemptive influence maximization

1 INTRODUCTION

Influence maximization is the task of finding a small set of seed nodes to generate the largest possible influence spread in a social network [16]. It models the important viral marketing applications in social networks, and many aspects of influence maximization have been extensively studied in the research literature. In most studies, influence propagation starts from a set of seed nodes, which are selected before the propagation starts. Propagation starts from all seed nodes together at the same time and proceeds in either discrete or continuous time to reach other nodes in the network, and the objective to maximize is typically the *influence spread*, defined as the expected number of nodes activated through the stochastic diffusion process.

In practice, however, when a marketing campaign starts, users' reactions to the campaign are not synchronized at the same time. Some users react to the campaign immediately, while others may react after a significant delay. Moreover, propagation may not only start from seed users that the campaign originally selected. It is possible that other users may spontaneously react to the campaign and also propagate the information and influence in the campaign. We call this phenomenon *self activation*, which is in contrast to the seed activation by the external force. While seed activation by the external force typically requires a marketing budget to be successful, self activations are spontaneous and do not require a budget. Self activation may also lead to other interesting objectives one may want to optimize, as we will discuss shortly. To give some concrete examples of self-activation without marketing influence, people may naturally share consumer product recommendations with their friends and businesses may attract clients through organic referrals. A product or business marketing team interested in influence maximization should consider this natural activity. Therefore, self activation is a realistic phenomenon in viral marketing, but it has not been well addressed in the influence maximization literature. In this paper, we incorporate self activation into the influence propagation model and provide a systematic study on the impact of self activation to the influence maximization task.

We first incorporate self activation with the classical independent cascade (IC) model to propose the self-activation independent cascade (SAIC) model of influence propagation. In the SAIC model, social network is modeled as a directed graph, and each node u has a *self-activation probability* $q(u)$ indicating the probability of u being self activated by the campaign. If activated, node u also has a random *self-activation delay* $\delta(u)$ sampled from distribution $\Delta(u)$, such that u is self-activated at time $\delta(u)$ after the campaign starts at time 0. A seed node v selected explicitly by the campaign will be deterministically activated at time $\delta(v)$, equivalently as saying that its self-activation probability $q(v)$ is boosted to 1. Propagation from seed nodes and self-activated nodes follows the classical IC model: if a node u is activated, then it has one chance to activate each of its out-going neighbor v with success probability $p(u, v)$. We further extend the classical IC model by allowing real-time delay on the edges: if u would successfully activate v, then this activation would occur after the random *propagation delay* of $d(u, v)$ sampled from a distribution $D(u, v)$, from the time u is activated. Thus, overall the propagation starts from the seed nodes and self-activated nodes, and a node is activated either because it is a seed, or because it is self-activated, or because it is activated by a neighbor, and the activation time is the earliest time when one of the above activation happens. Once a node is activated, it stays as activated. The reason we allow real time delays is to make it more realistic when we study the new objective functions discussed below.

With the incorporation of self activation into the SAIC model, we are able to study several different influence maximization objectives. The first objective is closer to the classical influence maximization, where we aim at select k seed nodes to maximize the total influence spread after boosting the self-activation probabilities of seed nodes to 1. We refer to this objective as *boosted influence spread*, denoted $\sigma^B(S)$ for seed set S, and the corresponding influence maximization problem as *boosted influence maximization (BIM)*. Note that the

total boosted influence spread $\sigma^B(S)$ counts both nodes activated by seeds and nodes activated by self-activated nodes.

Besides $\sigma^B(S)$, self activation further allows us to study some interesting new objectives. Conceptually, in the SAIC model, we can view a node as an *organic influencer* if the node is frequently and easily self-activated and its influence often reaches many other nodes first before other self-activated nodes. To model this, we precisely define the *preemptive influence spread* $\rho(A)$ of node set A as the expected number of nodes that some node $u \in A$ reaches first, if u is self-activated, before other self-activated nodes. Then the *preemptive influence maximization (PIM)* problem is to identify the set of k nodes that has the largest preemptive influence spread, which models the task of identifying top organic influencers in a network. Furthermore, we define *boosted preemptive influence spread* $\rho^B(S)$ of a seed set S as the preemptive influence spread of set S after we boost the self-activation probability of every node in S to 1. Then the *boosted preemptive influence maximization (BPIM)* problem is to find k seed nodes with the maximum boosted preemptive influence spread $\rho^B(S)$. BPIM corresponds to the viral marketing campaign that focuses on the reach of the campaign from the selected seed nodes rather than self-activated nodes, because for example the seed nodes carry high-quality and more effective campaign messages.

For the above objectives, we first study their properties and show that σ^B and ρ^B are monotone and submodular, while ρ is additive. Then, based on these properties and the reverse influence sample (RIS) approach [3], we design scalable approximation algorithms for the three problems BIM, BPIM, and PIM. Even though our algorithms are patterned from the existing algorithm, the new objectives studied here requires nontrivial adaptation of the algorithm. Especially for BPIM and PIM, we need to redesign the reverse simulation procedure to generate what we call *preemptive reverse reachable (P-RR) sets*, which are more sophisticated than the standard reverse reachable (RR) sets [3, 21, 22]. We prove that our algorithms solve BIM and BPIM with approximation ratio $1 - 1/e - \varepsilon$ for any $\varepsilon > 0$, and solve PIM with approximation ratio $1 - \varepsilon$, and all algorithms can run in time near linear to the graph size.

Finally, we conduct extensive experiments on real-world datasets and compare our algorithms with related baselines solving classical influence maximization or influence-based network centrality. We demonstrate that for the PIM problem, our IMM-PIM algorithm significantly outperforms the baselines on the achieved preemptive influence spread in all test cases, showing that utilizing the knowledge of self activation is important in finding organic influential nodes in a network. For the BIM and BPIM problems, our algorithms have minor improvements in influence spread in some cases where the self-activation behaviors of the nodes are non-uniform, which algorithms such as IMM that are oblivious to self-activation may be used for these problems, but it may still be beneficial to use our algorithms designed for the self-activation scenarios. All our algorithms can scale to large networks with hundred thousands of nodes and edges.

To summarize, we have the following contributions: (a) we incorporate the realistic self-activation scenario into influence propagation and study three influence maximization problems PIM, BPIM, and BIM due to this incorporation; (b) we design scalable algorithms for all three problems with theoretical approximation guarantee; and (c) we demonstrate through experiments that our algorithms provide significantly better results for PIM, and also outperform other algorithms in non-uniform self-activation scenarios.

Due to the space constraint, some experiments, some pseudocode and all proofs are moved to the extended version [1].

1.1 Related Work

Domingos and Richardson are the first to study influence maximization [11, 20], but Kempe et al. [16] are the first to formulate the problem as a discrete optimization problem, describe the independent cascade (IC), linear threshold and other models, and propose to use submodularity and greedy algorithm to solve influence maximization. Since then influence maximization has been extensively studied in various fronts, including its scalability [3, 9, 10, 14, 19, 21, 22, 24], robust influence maximization [7, 12], competitive and complementary influence maximization [4, 6, 13, 17, 23], etc.

Borgs et al. [3] propose the novel reverse influence sampling (RIS) approach that guarantees both the approximation ratio and near-linear running time, and it has been improved in a series of studies [3, 19, 21, 22]. In this paper, we adapt the IMM algorithm in [21] mainly because of its clarity, and other algorithms such as D-SSA of [19] can be plugged in too.

Our PIM problem has connection with the Shapley centrality proposed in [8]. In particular, in the special case when all nodes have self-activation probability 1, uniform self-activation delay distribution, and propagation delays can be ignored, the preemptive influence spread of a node coincides with its Shapley centrality. Thus, the general preemptive influence spread studied in this paper is more realistic than the Shapley centrality, and we compare against the Shapley centrality algorithm in our experiments and show that our algorithm achieves much better PIM result.

2 MODEL AND PROBLEM DEFINITION

2.1 Self-Activation Propagation Model

A social network is modeled as a directed graph $G = (V, E)$, where V is a finite set of vertices or nodes, and $E \subseteq V \times V$ is the set of directed edges connecting pairs of nodes. Let $n = |V|$ and $m = |E|$. In this paper, we study the influence propagation model where every node has a chance to be self-activated, even without being selected as a seed. In this model, at time 0 we assume a marketing campaign is started, and then every node may be self-activated by this campaign, and this activation may occur after a random delay from the beginning of the campaign. Technically node self activation is governed by the following set of parameters.

Definition 2.1. (Self-Activation Probability and Delay). In a social network, every node $u \in V$ can be self-activated as a seed by the campaign with *self-activation probability* $q(u) \in [0, 1]$. If u is self-activated, then it is activated after a random delay $\delta(u) \in [0, +\infty)$ drawn from a *self-activation delay* distribution $\Delta(u)$.

We combine self activation and independent cascade model [16] and further add real-time propagation delays on edges to obtain the self-activation independent cascade (SAIC) model. In the SAIC model, every node $u \in V$ is associated with self-activation probability and delay as defined in Definition 2.1. Meanwhile, every

edge $(u, v) \in E$ is associated with two parameters: 1) a *propagation probability* $p(u, v) \in (0, 1]$ ($p(u, v) = 0$ if and only if $(u, v) \notin E$); 2) a *propagation delay* distribution $D(u, v)$ with range $[0, +\infty)$. The SAIC model proceeds following the rules below:

(1) For each node $u \in V$, it is self-activated with probability $p(u)$, and if so it is activated at time $\delta(u)$ drawn from $\Delta(u)$ (denoted as $\delta(u) \sim \Delta(u)$), unless it has been previously activated by other nodes before $\delta(u)$.
(2) For any node $u \in V$ activated at time t (self-activated or neighbor-activated), it tries once to activate each of its outgoing neighbors v with propagation probability $p(u, v)$. If the activation is successful, the propagation delay $d(u, v)$ is sampled from $D(u, v)$ and v would be activated at time $t + d(u, v)$, unless v has been activated before this time.
(3) A node v is activated (both self-activated or neighbor-activated) at earliest time t when any activation of v would happen, and v stays active afterwards.

The above description of the SAIC model only considers the propagation starting from the self-activated nodes. We could further include externally selected *seed nodes* in the model as follows. Let S be the set of (externally selected) seed nodes. Then besides the three rules above, we have one more rule:

(4) For each seed $u \in S$, its self-activation probability is boosted to 1, that is, it is for sure activated, and it is activated at time $\delta(u) \sim \Delta(u)$, unless it has been activated by its neighbors before that time.

Note that when the self-activation probabilities of all nodes are 0, the self-activation delays for all nodes take deterministic value 0, and propagation delays of all nodes take deterministic value 1, SAIC model falls back to the classical independent cascade (IC) model. We add self activation to model the realistic situation that some users may spontaneously help propagating the marketing campaign. We introduce delay parameters because in reality people take actions asynchronously, and when we consider preemptive influence spread (to be defined shortly), delay factor does matter on who could claim credits on the activation of each node.

The SAIC model can be equivalently described as propagations in a *possible world model*. A possible world W contains all randomness in a SAIC propagation. In particular, W is a tuple $(A_W, \delta_W, L_W, d_W)$, where A_W is the random set of all self-activated nodes governed by the self-activation probability q, δ_W is the vector of self-activation delays sampled from Δ, L_W is the set of live edges governed by the propagation probability p (i.e. each edge $(u, v) \in E$ is live with probability $p(u, v)$), and d_W is the vector of propagation delays sampled from D. We use $\mathcal{W}(q, \Delta, p, D)$ to denote the probability space of all possible worlds, determined by parameters q, Δ, p, D. In a fixed possible world W, a live path P is a path consisting of only live edges in L_W. The propagation delay $d_W(P)$ of a live path is the sum of propagation delays on all live edges. For a $u \in A_W$ and a live path P from u to v in W, let $T_W(P) = \delta_W(u) + d_W(P)$ be the *total delay* of live path P. We also use $T_W(u, v)$ to denote the minimum total delay among all live paths from u to v. Propagation starts from nodes in A_W and follows the direction of all live edges and incurs delay on the edges, and a node v is activated at a time t if the minimum total delay of all live paths from any node in A to v is t. If we have externally selected seed set S, then the propagation starts from $S \cup A$ instead of A, and all other aspects remain the same. It is easy to see that the above described possible world model is just a different way of stating the SAIC model, when we determine all randomness before the propagation starts.

2.2 Self-Activation Influence Maximization

In the SAIC model, we consider several optimization tasks, each corresponding to a different objective function. We start with an objective function that is close to the influence spread objective function in the classical IC model. For a possible world W and a seed set $S \subseteq V$, let $\Phi^B_W(S)$ denote the set of nodes activated in the possible world W when S is the seed set (nodes in S have their self-activation probabilities boosted to 1). We call $\sigma^B(S) = \mathbb{E}_{W \sim \mathcal{W}(q,\Delta,p,D)}[|\Phi^B_W(S)|]$ the *boosted influence spread* of seed set S in the SAIC model. Note that $\sigma^B(\emptyset)$ may be greater than 0 since some nodes may be self-activated, and thus we use the word "boosted" to refer to the final influence spread due to the boosting of self-activation probabilities of seed nodes to 1. It is easy to see that the delay distributions Δ and D do not affect the final set of activated nodes, and thus $\sigma^B(S)$ doesn't depend on Δ and D. Moreover, when $q \equiv 0$, $\sigma^B(S)$ becomes the influence spread in the classical IC model. The first optimization task is to maximize $\sigma^B(S)$:

Definition 2.2 (Boosted Influence Maximization). *Boosted influence maximization (BIM)* is the optimization task with the directed graph $G = (V, E)$, the self-activation probabilities q, the propagation probabilities p, and a budget k as the input, and the goal is to find an optimal seed set S^* having at most k nodes, such that the *boosted influence spread* of S^* is maximized, i.e., $S^* = \text{argmax}_{|S| \le k} \sigma^B(S)$.

The boosted influence spread and boosted influence maximization is close to the classical influence spread and influence maximization concepts. We introduce them as a stepping stone for the new concept of preemptive influence spread and preemptive influence maximization.

In the SAIC model, a natural metric measuring the influence ability of a set of nodes S is the number of nodes that are actually activated due to the propagation from nodes in S, not by other sources. We define this metric formally as follows. First, we assume that all delay distributions in Δ and D are continuous functions and thus there is no probability mass at any given value. Thus, it is safe to assume that in any possible world W, the total delay of any path would be different, since the worlds with paths having same delays have probability measure 0. Given a possible world $W = (A_W, \delta_W, L_W, d_W)$, let $\mathcal{P}_W(u, v)$ denotes a set of all live paths in W starting from node u and ending at v. For a set of nodes A, we use $\Gamma_W(A)$ to denote the set of nodes v that have minimum total delays from some node in A to v, i.e. $\Gamma_W(A) = \{v \mid \exists u \in A_W \cap A, \exists P \in \mathcal{P}_W(u, v), T_W(P) < +\infty, \forall u' \in A_W \setminus A, \forall P' \in \mathcal{P}_W(u', v), T_W(P) < T_W(P')\}$. Set $\Gamma_W(A)$ contains all activated nodes in W whose activation sources are some nodes in A. In other words, A could claim full credits for activating $\Gamma_W(A)$ in W. We define $\rho(A) = \mathbb{E}_{W \sim \mathcal{W}(q,\Delta,p,D)}[|\Gamma_W(A)|]$ the *preemptive influence spread* of node set A. Intuitively, preemptive influence spread $\rho(A)$ measures the contribution of node set A in propagating

and activating nodes in the SAIC model, when there is no externally selected seed nodes.

When we use preemptive influence spread $\rho(A)$ as our objective function, we have the preemptive influence maximization problem.

Definition 2.3 (Preemptive Influence Maximization). Preemptive influence maximization (PIM) is the optimization task with the directed graph $G = (V, E)$, the self-activation probabilities q, the self-activation delay distribution Δ, the propagation probabilities p, the propagation delay distribution D, and a budget k as the input, and the goal is to find an optimal set A^* having at most k nodes, such that the *preemptive influence spread* of A^* is maximized, i.e., $A^* = \text{argmax}_{|A|\le k}\, \rho(A)$.

Preemptive influence maximization defined above corresponds to the application where a company may want to identify top organic influencers in an online social network, to study the characteristics that make them influential. These users could be targeted to propagate company-specific information without changing their activation behavior.

We remark that, when the self-activation probabilities of all nodes are 1, self-activation delays of all nodes follow the same distribution, and propagation delays of all nodes take deterministic value 0 (i.e. propagations are instantaneous), preemptive influence spread of each individual node coincides with the Shapley centrality defined in [8].

In preemptive influence maximization, we do not have the action of selecting seeds and changing the behavior of seeds. We can further incorporate seed selection with preemptive influence maximization as follows. In a possible world $W = (A_W, \delta_W, L_W, d_W)$, for a seed set S, similar to $\Gamma_W(A)$ we define $\Gamma^B_W(S)$ as the set of nodes that are activated due to S, after nodes in S are selected as seeds and their self-activation probabilities are boosted to 1, that is, $\Gamma^B_W(S) = \{v \mid \exists u \in S, \exists P \in \mathcal{P}_W(u, v), T_W(P) < +\infty, \forall u' \in A_W \setminus S, \forall P' \in \mathcal{P}_W(u', v), T_W(P) < T_W(P')\}$. We define the *boosted preemptive influence spread* $\rho^B(S)$ of a seed set S as $\rho^B(S) = \mathbb{E}_{W\sim\mathcal{W}(q,\Delta,p,D)}[|\Gamma^B_W(S)|]$. We can now use $\rho^B(S)$ as our third objective function to define the third optimization task:

Definition 2.4 (Boosted Preemptive Influence Maximization). Boosted preemptive influence maximization (BPIM) is the optimization task with the directed graph $G = (V, E)$, the self-activation probabilities q, the self-activation delay distribution Δ, the propagation probabilities p, the propagation delay distribution D, and a budget k as the input, and the goal is to find an optimal seed set S^* having at most k nodes, such that the *boosted preemptive influence spread* of S^* is maximized, i.e., $S^* = \text{argmax}_{|S|\le k}\, \rho^B(S)$.

BPIM defined above models the applications where the marketing campaign wants to engage in explicit incentive for a set of seed nodes (e.g. giving out free sample products) so that the seed nodes will be boosted to adopt the campaign and start propagating it. The difference between BPIM and BIM (Definition 2.2) is that BPIM only optimizes for the number of nodes first activated by the seed nodes, while BIM optimizes for the number of all activated nodes. The difference between BPIM and PIM (Definition 2.3) is that BPIM actively boosts the self-activation probabilities of seed nodes while PIM does not change node behaviors, and this is also the reason in PIM we avoid calling the set A selected as a seed set. Although all three problems look similar on surface, they are different and require separate algorithmic solutions. Moreover, PIM is very different from BIM and BPIM in that their algorithmic solutions would have different approximation guarantees. This is because the preemptive influence spread has different properties from the other two objective functions, as we discuss in the next section.

Finally, we remark that our proposed SAIC model is rich enough to consider realistic self-activation scenarios and all the above optimization tasks, while we do not introduce further parameters to complicate situation. For example, for a seed node $u \in S$, we could further consider shortening its self-activation delay or boosting its self-activation probability partially instead of 1. The added flexibility would not significantly change our algorithm design and analysis but only complicate our presentation. On the other hand, assuming seed selection would not shorten the self-activation delay is also reasonable, since when an online marketing campaign starts, a user in the network need to come online to be aware of this marketing campaign, and thus the initial delay from the time the campaign starts to the user coming online is not likely affected by the user being selected as a seed. Finally, the added parameters, such as self-activation delay distributions, self-activation probabilities are likely to be empirically obtained from real marketing campaigns, while the extraction of propagation probabilities and propagation delays have been well studied in the literature (e.g. [15, 18]).

2.3 Properties of Influence Spread Functions

We now show the key properties of the three influence spread functions $\rho(\cdot)$, $\rho^B(\cdot)$, and $\sigma^B(\cdot)$, which are crucial for later algorithm design. For a set function $f : 2^V \to \mathbb{R}$, we say that f is a) *additive* if for any subset $S \subseteq V$, $f(S) = \sum_{v\in S} f(\{v\})$; b) *monotone* if for any two subsets $S \subseteq T \subseteq V$, $f(S) \le f(T)$; and c) *submodular* if for any two subsets $S \subseteq T \subseteq V$ and an element $v \in V \setminus T$, $f(T \cup \{v\}) - f(T) \le f(S \cup \{v\}) - f(S)$. The following lemma summarizes the key properties of the three influence spread functions.

Lemma 2.5 (Influence Spread Properties). *(1) The preemptive influence spread function ρ is additive; (2) the boosted preemptive influence spread function ρ^B is monotone and submodular; and (3) the boosted influence spread function σ^B is monotone and submodular.*

3 SCALABLE IMPLEMENTATIONS

In this section, we develop scalable algorithms for all three problems PIM, BPIM, and BIM, based on the reverse influence sampling (RIS) approach [3, 21, 22].

The key concept in RIS is the reverse-reachable set. A (random) *Reverse-Reachable (RR) set $R(v)$ rooted at node $v \in V$* is the set of nodes reachable from v by reverse simulating a propagation from v. More precisely, in the SAIC model, $R(v)$ is the set of nodes that can reach v in a random possible world $W = (A_W, \delta_W, L_W, d_W)$ following only live edges in L_W. We use root$(R(v))$ to denote its root v. When we do not specify the root, an *RR set R* is one rooted at a node picked uniformly at random from V. We will use the notations $R_W(v)$ and R_W when we want to clarify that the RR set is under the possible world W.

An RR set R has the following intrinsic connection with the influence spread $\sigma(S)$ of seed set S in the classical IC model [3, 22]:

$$\sigma(S) = n \cdot \mathbb{E}[\mathbb{I}\{S \cap R \neq \emptyset\}], \tag{1}$$

where $\mathbb{I}$ is the indicator function. RIS approach utilizes this fact to generate enough RR sets to estimate the influence spread and turn influence maximization into a coverage problem of finding k nodes that covers (a.k.a. appears in) the most number of RR sets. We now need to adapt the RIS approach for our problems considered in the paper. Our adaptations are patterned over the IMM algorithm [21], although it would be as easy to adapt other state-of-the-art RIS algorithms too.

3.1 Algorithm for BIM

We first present the adaption of IMM to the BIM problem, since BIM is close to the original influence maximization problem. The boosted influence spread σ^B has the following connection with a random RR set R:

Lemma 3.1. *For any seed set S,*

$$\sigma^B(S) = n \cdot \mathbb{E}_{W \sim \mathcal{W}(q,\Delta,p,D)}[\mathbb{I}\{(S \cup A_W) \cap R_W \neq \emptyset\}]. \quad (2)$$

Eq. (2) enables the RIS approach as for the classical influence maximization. We adapt the IMM algorithm of [21] to get the IMM-BIM algorithm, as given in Algorithm 1. The two main parameters λ' and $\lambda^*(\ell)$ used in the algorithm are given below:

$$\lambda' \leftarrow [(2 + \frac{2}{3}\varepsilon') \cdot (\ln \binom{n}{k} + \ell \cdot \ln n + \ln \log_2 n) \cdot n]/\varepsilon'^2 \quad (3)$$

$$\lambda^*(\ell) \leftarrow 2n \cdot ((1 - 1/e) \cdot \alpha + \beta)^2 \cdot \varepsilon^{-2} \quad (4)$$

$$\alpha \leftarrow \sqrt{\ell \ln n + \ln 2}; \beta \leftarrow \sqrt{(1 - 1/e) \cdot (\ln \binom{n}{k} + \alpha^2)}$$

The algorithm contains two phases. In Phase 1, we generate θ RR sets $\mathcal{R}$, where θ is computed to guarantee the approximation with high probability. In Phase 2, we use the greedy algorithm to find k seed nodes that cover as many RR sets in $\mathcal{R}$ as possible: in each iteration, we find one seed node that covers the most number of remaining RR sets not covered by previously selected seed nodes. The NodeSelection procedure of Phase 2 implements the above greedy algorithm, and is exactly the same as in [21], and thus we omit it here.

Phase 1 follows the IMM structure: it uses the for-loop to estimate a lower bound of OPT, the optimal solution to the BIM problem, by repeatedly halving the estimate x_i and checking if the estimate is valid. The validity check is by running the greedy NodeSelection procedure (line 14) to find a seed set S_i and getting its influence spread estimate, since the greedy algorithm should give a constant approximation of OPT. IMM-BIM differs from IMM because it needs to incorporate self-activation probabilities $q(u)$'s. In particular, when we generate an RR set R, for each node $u \in R$, we sample a random coin with bias $q(u)$ to see if u is self-activated. If so, it means this RR set R has already be covered by the self-activated u, and there is no need to select an extra seed node to cover R. In this case, we do not need to store R, but only need to count its number in variable *covered*, which records the number of RR sets covered by self-activated nodes. Only an RR set R that contains no self-activated nodes needs to be stored in $\mathcal{R}$ for later greedy seed selection (line 11). The variable *covered* is used to estimate boosted influence spread $\sigma^B(S)$. In particular, by Eq. (2), $\sigma^B(S)$ can

Algorithm 1: IMM-BIM: adapted IMM for the BIM problem

Input: Graph $G = (V, E)$, propagation probabilities $\{p(u,v)\}_{(u,v)\in E}$, self-activation probabilities $\{q(u)\}_{u\in V}$, budget k, accuracy parameters (ε, ℓ)
Output: seed set S

// Phase 1: Estimate θ, the number of RR sets needed, and generate these RR sets

1 $\mathcal{R} \leftarrow \emptyset$; $LB \leftarrow 1$; $\varepsilon' \leftarrow \sqrt{2}\varepsilon$; *covered* $\leftarrow 0$;
2 using binary search to find a γ such that $\lceil \lambda^*(\ell) \rceil / n^{\ell+\gamma} \leq 1/n^{\ell}$ *// Workaround 2 in [5], $\lambda^*(\ell)$ is defined in Eq. (4)*
3 $\ell \leftarrow \ell + \gamma + \ln 2 / \ln n$;
4 **for** $i = 1$ *to* $\log_2(n-1)$ **do**
5 $\quad x_i \leftarrow n/2^i$;
6 $\quad \theta_i \leftarrow \lambda'/x_i$; *// λ' is defined in Eq. (3)*
7 $\quad$ **while** $|\mathcal{R}|$ + *covered* $< \theta_i$ **do**
8 $\quad\quad$ Select a node v from V uniformly at random;
9 $\quad\quad$ Generate RR set R from v;
10 $\quad\quad$ **if** $\exists u \in R$, *u is self-activated with probability $q(u)$* **then**
11 $\quad\quad\quad$ *covered* $\leftarrow$ *covered* $+ 1$;
12 $\quad\quad$ **else**
13 $\quad\quad\quad$ insert R into $\mathcal{R}$;
14 $\quad S_i \leftarrow$ NodeSelection$(\mathcal{R}, k)$;
15 $\quad$ **if** $n \cdot F^S_{\mathcal{R}}(S_i) \geq (1 + \varepsilon') \cdot x_i$ **then**
$\quad\quad$ *// $F^S_{\mathcal{R}}(S)$ is defined in Eq. (5)*
16 $\quad\quad LB \leftarrow n \cdot F^S_{\mathcal{R}}(S_i)/(1 + \varepsilon')$;
17 $\quad\quad$ **break**;
18 $\theta \leftarrow \lambda^*(\ell)/LB$; *// $\lambda^*(\ell)$ is defined in Eq. (4)*
19 **while** $|\mathcal{R}|$ + *covered* $\leq \theta$ **do**
20 $\quad$ Select a node v from V uniformly at random;
21 $\quad$ Generate RR set R from v;
22 $\quad$ **if** $\exists u \in R$, *u is self-activated with probability $q(u)$* **then**
23 $\quad\quad$ *covered* $\leftarrow$ *covered* $+ 1$;
24 $\quad$ **else**
25 $\quad\quad$ insert R into $\mathcal{R}$;

// Phase 2: select seed nodes from the generated RR sets

26 $S \leftarrow$ NodeSelection$(\mathcal{R}, k)$;
27 **return** S.

be estimated as n times the fraction of RR sets that are covered either by S or by self-activated nodes. This fraction is defined as $F^S_{\mathcal{R}}(S)$:

$$F^S_{\mathcal{R}}(S) = \frac{\textit{covered} + \sum_{R\in\mathcal{R}} \mathbb{I}\{S \cap R \neq \emptyset\}}{\textit{covered} + |\mathcal{R}|}. \quad (5)$$

By an analysis similar to that of the IMM algorithm [21], we have the following theorem. Let $\tilde{v}$ be a random node selected from V with probability proportional to its indegree, and let $\sigma(\tilde{v})$ denote the influence spread of $\tilde{v}$ in the corresponding IC model.

Theorem 3.2. *Let S^* be the optimal solution of the BIM problem. For every $\varepsilon > 0$ and $\ell > 0$, with probability at least $1 - \frac{1}{n^\ell}$, the output S^o of* IMM-BIM *satisfies*

$$\sigma^B(S^o) \geq \left(1 - \frac{1}{e} - \varepsilon\right)\sigma^B(S^*),$$

In this case, the expected running time for IMM-BIM *is* $O((k+\ell)(n+m)\log n/\varepsilon^2 \cdot (\mathbb{E}[\sigma(\tilde{v})]/\sigma^B(S^*))) = O((k+\ell)(n+m)\log n/\varepsilon^2)$.

Similar to IMM, the above theorem shows that IMM-BIM achieves $1 - 1/e - \varepsilon$ approximation with near-linear running time. The theorem explicitly shows the ratio $\mathbb{E}[\sigma(\tilde{v})]/\sigma^B(S^*)$, which is less than 1, in order to compare later with other algorithms.

3.2 Algorithm for BPIM

We next discuss our implementation of BPIM, which we call IMM-BPIM. Since the objective function $\rho^B(S)$ is monotone submodular (Lemma 2.5), IMM-BPIM follows the general structure of greedy seed selection. However, it differs from IMM and IMM-BIM significantly in its RR set definition and generation process. Intuitively, the preemptive influence spread of a seed node u only counts the activated nodes that u reaches first before any other seed nodes or self-activated nodes. In terms of RR sets, this means that a node u can be included in the preemptive RR set only if the total delay from u to the root v is smaller than the minimum total delay from any self-activated nodes to v. We formally define the *preemptive reverse-reachable (P-RR) set as follows*. Given a possible world $W = (A_W, \delta_W, L_W, d_W)$ in the SAIC model, a P-RR set $R^P_W(v)$ rooted at v is the set of nodes u such that (1) u could reach v through live edges in L_W, and (2) the total delay of u to v, $T_W(u, v)$, is less than or equal to the minimum total delay from any self-activated node in A_W to v. When we do not specify the root v, P-RR set R^P_W is a P-RR set with root selected uniformly at random among all nodes in V. The subscript W could be omitted when the context is clear. With this definition, we can obtain the following connection between a P-RR set and the preemptive influence spread $\rho^B(S)$.

Lemma 3.3. *For any seed set S,*

$$\rho^B(S) = n \cdot \mathbb{E}_{W\sim\mathcal{W}(q,\Delta,p,D)}[\mathbb{I}\{S \cap R^P_W \neq \emptyset\}]. \tag{6}$$

With Eq. (6), we can see that as long as we can properly generate P-RR sets, we can use the IMM algorithm in the same way to find the seed sets. Therefore, our main focus now is to efficiently generate a random P-RR set. This algorithm is implemented as P-RR as given in Algorithm 2. The main idea is that for each node $v \in V$, we add its shadow node $\hat{v}$ and an edge from $\hat{v}$ to v, and consider the delay on the edge $(\hat{v}, v)$ as a sample of the self-activation delay $\delta(v) \sim \Delta(v)$. Then the delay from any node (real or shadow) to the root v^r is the minimum delay along any path from the node to v^r. Let u^s be a node that is self-activated and its corresponding shadow node $\hat{v}^s$ has the minimum delay to the root v^r among all shadow nodes. Then the P-RR set R^P is the set of nodes whose shadow nodes have delay less than or equal to the delay of $\hat{v}^s$.

To find u^s and R^P, we apply the idea from the Dijkstra's shortest path algorithm: from the candidate nodes we touched so far (set Q), we take node w that has the shortest delay as the next one to explore (line 6). If w is a shadow node $\hat{v}$, we insert v into R^P (line 9), then test if v is self-activated or not, and if so, we find $u^s = v$,

Algorithm 2: P-RR: Preemptive RR Set Generation

Input: root v^r, Graph $G = (V, E)$, self-activation probability p, random distribution of self-activation delay Δ, propagation probability p, random distribution of propagation delay D

Output: P-RR set R^P, node u^s that is the first activating v^r

1 $Q \leftarrow \{v^r\}$; $R^P \leftarrow \emptyset$; $u^s \leftarrow -1$;
2 **for** *each* $v \in V$ **do**
3 $delay[v] \leftarrow +\infty$;// *initial delays for reaching the root*
4 $delay[v^r] \leftarrow 0$;
5 **while** $Q \neq \emptyset$ **do**
6 $w \leftarrow \min_{w' \in Q} delay[w']$;
7 delete w from Q;
8 **if** w *is a shadow node* $\hat{v}$ **then**
9 insert v into R^P;
10 **if** v *is self-activated with probability* $q(v)$ **then**
11 $u^s \leftarrow v$;
12 **break**;
13 **else**
  // *let w be the real node v*
14 sample $\delta(v) \sim \Delta(v)$;
15 $delay[\hat{v}] \leftarrow delay[v] + \delta(v)$;
16 insert $\hat{v}$ into Q;
17 **for** *each real in-neighbor u of v in G* **do**
18 **if** (u, v) *is sampled as live with probability* $p(u, v)$ **then**
19 sample $d(u, v) \sim D(u, v)$;
20 $tmp \leftarrow delay[v] + d(u, v)$;
21 **if** $delay[u] = +\infty$ **then**
22 insert u into Q;
23 **if** $tmp < delay[u]$ **then**
24 $delay[u] \leftarrow tmp$;
25 **return** R^P, u^s.

and R^P contains all the shadow nodes we have explored so far and the algorithm stops (lines 10–12). If w is a real node v, we first sample the delay $\delta(v) \sim \Delta(v)$ as the delay from v's shadow $\hat{v}$ to v, compute the delay of $\hat{v}$ as $delay[v] + \delta(v)$, and insert $\hat{v}$ into the candidate node set Q (lines 14–16). We then do reverse simulation along all of v's incoming edges (u, v), and sample the edge delay $d(u, v) \sim D(u, v)$, and do proper updates of $delay[u]$ (lines 17–24). The algorithm guarantees that the node sequence we explore has increasing delays, which in turn guarantees the correctness of R^P and u^s found by the algorithm. We remark that u^s would be useful in solving PIM, as to be explained in the next subsection.

With the P-RR set generation algorithm P-RR, we just plug it into the IMM algorithm and obtain IMM-BPIM. The full pseudocode is omitted. We have the theorem below for the IMM-BPIM algorithm.

Theorem 3.4. *Let S^* be the optimal solution of the BPIM. For every $\varepsilon > 0$ and $\ell > 0$, with probability at least $1 - \frac{1}{n^\ell}$, the output*

S^o *of the* IMM-BPIM *algorithm satisfies*

$$\rho^B(S^o) \geq \left(1 - \frac{1}{e} - \varepsilon\right) \rho^B(S^*).$$

In this case, the expected running time of the IMM-BPIM *algorithm is* $O((k+\ell)(n+m)\log^2 n/\varepsilon^2 \cdot (\mathbb{E}[\sigma(\tilde{v})]/\rho^B(S^*)))$.

Note that for the ratio $\mathbb{E}[\sigma(\tilde{v})]/\rho^B(S^*)$, one would expect that typically the optimal solution of BPIM would be larger than any single node influence spread, and thus the ratio is less than 1 and we have a near-linear time algorithm. In this case, the expected running time for IMM-BPIM still has one extra $\log n$ term comparing to that of IMM-BIM or IMM. This is because our reverse simulation algorithm P-RR needs to run a Dijkstra-like algorithm, and in particular we need to implement the set Q in Algorithm 2 as a priority queue to support insertion, deletion, updates, and finding the minimum value.

3.3 Algorithm for PIM

Finally, we consider the preemptive influence maximization (PIM) algorithm. PIM differs from BIM and BPIM in that we do not select seeds and boost their self-activation probabilities to 1. We only select k nodes who *spontaneously* have the largest preemptive influence spread, due to self activations. By Lemma 2.5, we know that the preemptive influence spread ρ is additive, which implies that we just need to estimate individual node's preemptive influence spread and select the top k of them. We still use the RIS approach for estimating individual node's preemptive influence spread, based on the following result. Let $u^s_W(v)$ be the node in the possible world W that is self-activated and can reach v with the minimum total delay $T(u,v)$, and u^s_W denotes such a random $u^s_W(v)$ where v is selected uniformly at random.

Lemma 3.5. *For any node* u,

$$\rho(\{u\}) = n \cdot \mathbb{E}_{W\sim\mathcal{W}(q,\Delta,p,D)}[\mathbb{I}\{u = u^s_W\}]. \tag{7}$$

With Lemma 3.5, we can randomly select a root v, and reverse simulate from v to find the node $u^s_W(v)$, and for each such node $u = u^s_W(v)$, we compute the fraction of times it appears in the reverse simulation, and multiply it with n to get u's preemptive influence spread $\rho(\{u\})$. This reverse simulation procedure has been done as part of P-RR algorithm, and its output u^s is the u^s_W we refer here.

With the above new way of reverse simulation, we can plug it into the IMM framework to obtain our algorithm IMM-PIM. The main difference is that we do not need greedy NodeSelection procedure to give a $1-1/e$ approximation of the optimal seed set covering the RR set sequence $\mathcal{R}$. Instead, each node u maintains a counter est_u to record the number of times the reverse simulation hits $u^s = u$, and we just select the top k nodes with the largest counters as our output set. This would give a $1-\varepsilon$ approximation instead of the $1-1/e-\varepsilon$ approximation as previous algorithms. For the same reason, the parameter $\lambda^*(\ell)$ should be redefined, replacing the factor $1-1/e$ in the parameter with 1. With these changes, IMM-PIM has the following theoretical guarantee.

Theorem 3.6. *Let* S^* *be the optimal solution of the PIM. For every* $\varepsilon > 0$ *and* $\ell > 0$, *with probability at least* $1 - \frac{1}{n^\ell}$, *the output* S^o *of the* IMM-BPIM *algorithm satisfies*

$$\rho(S^o) \geq (1-\varepsilon)\,\rho(S^*).$$

In this case, the expected running time of the IMM-PIM *algorithm is* $O((k+\ell)(n+m)\log^2 n/\varepsilon^2 \cdot (\mathbb{E}[\sigma(\tilde{v})]/\rho(S^*)))$.

4 EMPIRICAL EVALUATION

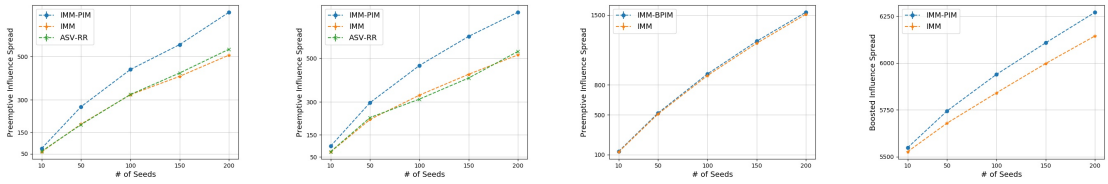
The main purpose of our empirical evaluation is to validate if and when using our self-activation aware influence maximization algorithms are beneficial over using the classical self-activation oblivious algorithms, and to quantify is the difference in performance. We conduct experiments on two real-world social networks to test the performance of our algorithms and compare them with the classical influence maximization and the Shapley centrality algorithms.

4.1 Experiment Setup

Data Description. We use the following two datasets, all of which have been used in a number of influence maximization studies. (1) **Flixster**. The Flixster dataset is a network of American social movie discovery service (www.flixster.com). To transform the dataset into a weighted graph, each user is represented by a node, and a directed edge from node u to v is formed if v rates one movie shortly after u does so on the same movie. The dataset is analyzed in [2], and the influence probability are learned by the topic-aware model. We use the learning result of [2] in our experiment, which is a graph containing 29 357 nodes and 212 614 directed edges. There are 10 probabilities on each edge, and each probability represents the influence from the source user to the sink on a specific topic. In our experiment, we test the first topic. (2) **NetHEPT**. The NetHEPT dataset [9] is extensively used in many influence maximization studies. It is an academic collaboration network from the "High Energy Physics Theory" section of arXiv from 1991 to 2003, where nodes represent the authors and each edge represents one paper co-authored by two nodes. There are 15 233 nodes and 58 891 undirected edges (including duplicated edges) in the NetHEPT dataset. We clean the dataset by removing those duplicated edges and obtain a directed graph $G = (V, E)$, $|V| = 15\,233$, $|E| = 62\,774$ (directed edges). The propagation probability on edges are set by weighted cascade model [16]: the probability of an edge (u, v) is set as the inverse of the in-degree of v. (3) **DBLP**. The DBLP dataset [24] is an academic collaboration network extracted from online archive DBLP (dblp.uni-trier.de). There are 654K nodes and 1990K directed edges in the DBLP. The propagation probabilities on the edges are also set by the weighted cascade model.

Algorithms. For the two problems (PIM, BPIM), we test our corresponding algorithms with the baseline IMM, which is oblivious to the self-activation behaviors and treat all nodes as no self activation and seed nodes as activated at time 0. For the PIM problem, we further compare against the efficient Shapley computation algorithm ASV-RR proposed in [8], which essentially treats all nodes as self-activations in a uniform random order. We use the same parameters settings for these algorithms: $\ell = 1$, $\varepsilon = 0.1$. We test seed set sizes of 10, 50, 100, 150 and 200.

Self-activation parameters and test cases. In practice, self-activation delays can be estimated from the users' access patterns to online social networks, and self-activation probabilities can be



(a) PIM-NetHEPT (b) PIM-Flixster (c) BPIM-NetHEPT (d) BPIM-NetHEPT

Figure 1: Influence Spread Results ($\varepsilon = 0.1$ with case 3).

Table 1: Running time results (in seconds).

Data	IMM-BIM	IMM-PIM	IMM-BPIM	IMM	ASV-RR
NetHEPT	0.7534	195.75	48.123	1.9712	74.175

Data	IMM-BIM	IMM-PIM	IMM-BPIM	IMM	ASV-RR
Flixster	1.3516	955.45	218.51	5.1072	235.34

estimated from the fraction of times users' participating in information cascades not due to the influence from the neighbors or external selections as seed users. Unfortunately for the datasets we use, these information are not available. Instead, we use synthetic settings, focusing on whether the knowledge of the self-activation behaviors would benefit our algorithm design. For self-activation probability $q(u)$ of node u, we first randomly select a value β_u from $[0, c]$ as a node u's base value, then we further test five cases: (0) uniform: $q(u) = \alpha_u$; (1) positively correlated: $q(u)$ is positively correlated with u's out-degree $d^+(u)$, in particular $q(u) = \min\{\beta_u \cdot d^+(u), 1\}$; (2) negatively correlated: $q(u) = \beta_u / d^+(u)$; (3) random mixing of cases 0 and 1: randomly pick half of the nodes with $q(u) = \alpha_u$ and the other half with $q(u) = \min\{\beta_u \cdot d^+(u), 1\}$; (4) random mixing of cases 0 and 2: randomly pick half of the nodes with $q(u) = \alpha_u$ and the other half with $q(u) = \beta_u / d^+(u)$. These five cases aim at scenarios where all users are equally likely to react to a campaign (case 0), high-degree nodes (usually more influential) are more likely or less likely to react to the campaigns, and mixture of uniform behavior and a correlation (or reverse correlation) behavior. We set $c = 2$ for BPIM and PIM tests, because otherwise the preemptive influence spread for PIM is too small. For self-activation delays, we use exponential distribution with rate 1 for all nodes. We already vary the self-activation behaviors through the self-activation probabilities, and thus we simply keep the self-activation delay distributions uniform. We also use the same exponential distribution for propagation delay distributions. Two proposed algorithms and two baselines are written in c++ and compiled by Visual studio. All experiments are conducted on a 15" MacBook Pro with a 2.5 GHz Intel Core i7 and 16 GB of 1600 MHz DDR3 memory.

4.2 Results

Influence spread result. As we can see in Figure 1, IMM-PIM significantly outperforms the baselines on the achieved preemptive influence spread, and IMM-BPIM also outperforms others when self-activation behaviors of the nodes are non-uniform. On average IMM-PIM improves about 32.7% than the other baselines, IMM-BPIM improves about 2.3% than IMM algorithm, and IMM-BIM improves about 2.1% than IMM algorithm.. Detailed results are in extended version [1].

Running time results. Table 1 reports the running time of all algorithms on both datasets, by using the default setting with the seed set size $k = 200$ in test case 3. We can clearly see the order of running time is BIM < IMM < IMM-BPIM < ASV-RR < IMM-PIM (we ignore the prefix IMM in our algorithm to fit the table width). This is inline with our theoretical analysis, which shows that the running time is inversely proportional to the optimal value of each problem. For example, the optimal solution of BIM is larger than that of the classical influence maximization because BIM has self-activated nodes contributing extra influence spread, and PIM has the smallest optimal value because the self-activation probabilities are small in general and the optimal set has to compete with other self-activated nodes on preemptive influence spread. IMM-BPIM and IMM-PIM are further slower due to the Dijkstra-like reverse simulation, which takes more time than the simple breadth-first-search simulation. But even for the slowest IMM-PIM algorithm, on the Flixster dataset with more than hundred thousands nodes and edges, it could complete in less than 16 minutes on our laptop test machine. Besides the dataset, running time is also related to the ε. We also test the ε values from 0.1 to 0.5 on NetHEPT for the proposed algorithms, and the details are in the extented version due to the space constraint [1].

In summary, our tests clearly demonstrate that for the PIM problem targeted for identifying top organic influencers in a graph with self-activation, our IMM-PIM algorithm significantly outperform other baselines in terms of result quality, which suggests that knowing the self-activation behavior is important for this task. For BPIM, our algorithm have small improvements for certain test cases with non-uniform self-activation behaviors. This may suggest that baseline such as IMM may be usable for these tasks, but one could still benefit from our algorithms in certain cases.

5 CONCLUSION AND FUTURE WORK

We introduce self activation and preemptive influence maximization task in this study. A future direction of our study would be incorporating self activation and preemptive influence spread considerations into other influence maximization tasks, such as competitive and complementary influence maximization, adaptive and online influence maximization, etc.

REFERENCES

[1] Anonymous. [n. d.]. Self-Activation Influence Maximization. ([n. d.]). https://github.com/Github-Conference/WSDM2020-951

[2] Nicola Barbieri, Francesco Bonchi, and Giuseppe Manco. 2012. Topic-aware social influence propagation models. In *ICDM'12*. IEEE.

[3] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. 2014. Maximizing social influence in nearly optimal time. In *ACM-SIAM (SODA '14)*.

[4] Ceren Budak, Divyakant Agrawal, and Amr El Abbadi. 2011. Limiting the spread of misinformation in social networks. In *WWW11*.

[5] Wei Chen. 2019. An Issue in the Martingale Analysis of the Influence Maximization Algorithm IMM. In *CSoNet*.

[6] Wei Chen, Alex Collins, Rachel Cummings, Te Ke, Zhenming Liu, David Rincon, Xiaorui Sun, Yajun Wang, Wei Wei, and Yifei Yuan. 2011. Influence maximization in social networks when negative opinions may emerge and propagate. In *SDM*.

[7] Wei Chen, Tian Lin, Zihan Tan, Mingfei Zhao, and Xuren Zhou. 2016. Robust Influence Maximization. In *KDD*.

[8] Wei Chen and Shang-Hua Teng. 2017. Interplay between social influence and network centrality: A comparative study on shapley centrality and single-node-influence centrality. In *WWW*. 967–976.

[9] Wei Chen, Yajun Wang, and Siyu Yang. 2009. Efficient influence maximization in social networks. In *KDD*.

[10] Wei Chen, Yifei Yuan, and Li Zhang. 2010. Scalable Influence Maximization in Social Networks under the Linear Threshold Model. In *ICDM*.

[11] Pedro Domingos and Matthew Richardson. 2001. Mining the network value of customers. In *KDD*.

[12] X. He and D. Kempe. 2016. Robust Influence Maximization. In *KDD*.

[13] Xinran He, Guojie Song, Wei Chen, and Qingye Jiang. 2012. Influence Blocking Maximization in Social Networks under the Competitive Linear Threshold Model.

[14] Kyomin Jung, Wooram Heo, and Wei Chen. 2012. IRIE: Scalable and Robust Influence Maximization in Social Networks. In *ICDM*.

[15] Saito K., Kimura M., Ohara K., and Motoda H. 2010. Selecting information diffusion models over social networks for behavioral analysis. In *ECML-PKDD*.

[16] David Kempe, Jon M. Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *KDD*.

[17] Wei Lu, Wei Chen, and Laks VS Lakshmanan. 2015. From competition to complementarity: comparative influence diffusion and maximization. *PVLDB* (2015).

[18] Gomez-Rodriguez M., Song L., Du N., Zha H., and Schölkopf B. 2016. Influence estimation and maximization in continuous-time diffusion networks. *ACM Transactions on Information Systems* 34, 2 (2016).

[19] Hung T. Nguyen, My T. Thai, and Thang N. Dinh. 2016. Stop-and-Stare: Optimal Sampling Algorithms for Viral Marketing in Billion-scale Networks. In *SIGMOD*.

[20] Matthew Richardson and Pedro Domingos. 2002. Mining knowledge-sharing sites for viral marketing. In *KDD*.

[21] Youze Tang, Yanchen Shi, and Xiaokui Xiao. 2015. Influence maximization in near-linear time: a martingale approach. In *SIGMOD*.

[22] Youze Tang, Xiaokui Xiao, and Yanchen Shi. 2014. Influence maximization: near-optimal time complexity meets practical efficiency. In *SIGMOD*.

[23] Chayant Tantipathananandh, Tanya Berger-Wolf, and David Kempe. 2007. A framework for community identification in dynamic social networks. In *KDD'07*.

[24] Chi Wang, Wei Chen, and Yajun Wang. 2012. Scalable influence maximization for independent cascade model in large-scale social networks. *DMKD* (2012).

A PSEUDOCODE FOR IMM-PIM

The parameter $\lambda^*(\ell)$ is replaced by $\tilde{\lambda}^*(\ell)$, defined as follows: we redefine it as $\tilde{\lambda}^*(\ell)$ below:

$$\tilde{\lambda}^*(\ell) \leftarrow 2n \cdot (\alpha + \tilde{\beta})^2 \cdot \varepsilon^{-2} \tag{8}$$

$$\alpha \leftarrow \sqrt{\ell \ln n + \ln 2}; \tilde{\beta} \leftarrow \sqrt{\ln \binom{n}{k} + \alpha^2}.$$

The pseudocode of IMM-PIM is given in Algorithm 3. The main difference comparing with IMM or IMM-BPIM is that: (a) replacing the greedy NodeSelection procedure by simply counting the number of occurrences of each node u as the first node reaching v in the reverse simulation (through variable est_u) and selecting the top k nodes with the highest number of occurrences; and (b) replacing $\lambda^*(\ell)$ with $\tilde{\lambda}^*(\ell)$.

B PROOFS OF THEOREMS AND LEMMAS

Proof of Lemma 2.5. We prove all three statements on a fixed possible world $W = (A_W, \delta_W, L_W, d_W)$, since taking expectation on W preserves additivity, monotonicity, and submodularity. For (1), it is straightforward if we observe that for each activated node v in W, there is a unique source node $u \in A_W$ that is the first activating v, by our assumption that no two live paths in W have the same total delay. This implies that $\Gamma_W(\{u\}) \cap \Gamma_W(\{u'\}) = \emptyset$ for any two different nodes $u, u' \in V$, and thus $|\Gamma_W(A)| = \sum_{u \in A} \Gamma_W(\{u\})$, for any $A \subseteq V$. For (2), monotonicity is trivial. For submodularity, it is sufficient to prove that for any two subsets $S \subseteq T \subseteq V$ and $u \in V \setminus T$, $\Gamma^B_W(T \cup \{u\}) \setminus \Gamma^B_W(T) \subseteq \Gamma^B_W(S \cup \{u\}) \setminus \Gamma^B_W(S)$. For a node $v \in \Gamma^B_W(T \cup \{u\}) \setminus \Gamma^B_W(T)$, there must exist a live path from u to v in W, such that the total delay of P, $T_W(P)$, is the minimum among all live paths from $T \cup \{u\} \cup A_W$ to v, which also implies that $T_W(P)$ is strictly less than the total delay of any live path from $T \cup A_W$ to v. Since $S \subseteq T$, this directly implies that $v \in \Gamma^B_W(S \cup \{u\})$ but $v \notin \Gamma^B_W(S)$. For (3), again the monotonicity is trivial, and submodularity proof is similar to (2). It is sufficient to prove that for any two subsets $S \subseteq T \subseteq V$ and $u \in V \setminus T$, $\Phi^B_W(T \cup \{u\}) \setminus \Phi^B_W(T) \subseteq \Phi^B_W(S \cup \{u\}) \setminus \Phi^B_W(S)$. For a node $v \in \Phi^B_W(T \cup \{u\}) \setminus \Phi^B_W(T)$, there is no live path from any node in $T \cup A_W$ to v but there exists a live path from u to v in W. Since $S \subseteq T$, this directly implies that $v \in \Phi^B_W(S \cup \{u\})$ but $v \notin \Phi^B_W(S)$. □

Proof of Lemma 3.1.

$$\begin{aligned}
&\mathbb{E}_{W \sim \mathcal{W}(q,\Delta,p,D)}[\mathbb{I}\{(S \cup A_W) \cap R_W \neq \emptyset\}] \\
&= \sum_{v \in V} \Pr\{v = \text{root}(R_W)\} \cdot \mathbb{E}_W[\mathbb{I}\{(S \cup A_W) \cap R_W \neq \emptyset \mid \\
&\qquad v = \text{root}(R_W)\}] \\
&= \frac{1}{n} \sum_{v \in V} \mathbb{E}_W[\mathbb{I}\{(S \cup A_W) \cap R_W(v) \neq \emptyset\}] \\
&= \frac{1}{n} \sum_{v \in V} \mathbb{E}_W[\mathbb{I}\{v \in \Phi^B_W(S)\}] \qquad (9) \\
&= \frac{1}{n} \cdot \mathbb{E}_W[|\Phi^B_W(S)|] \\
&= \frac{1}{n} \cdot \sigma^B(S),
\end{aligned}$$

Algorithm 3: IMM-PIM: Preemptive IMM Algorithm

Input: Graph $G = (V, E)$, self-activation probabilities q, self-activation delay distributions Δ, propagation probabilities p, propagation delay distributions D, budget k, accuracy parameters (ε, ℓ)

Output: set S

// Phase 1: Estimate θ, the number of P-RR sets needed, and generate these P-RR sets

$\mathcal{R} \leftarrow \emptyset$; $LB \leftarrow 1$; $\varepsilon' \leftarrow \sqrt{2}\varepsilon$;

using binary search to find a γ such that $\lceil \tilde{\lambda}^*(\ell) \rceil / n^{\ell+\gamma} \leq 1/n^{\ell}$ *// Workaround 2 in [5], $\tilde{\lambda}^*(\ell)$ is defined in Eq. (8)*

$\ell \leftarrow \ell + \gamma + \ln 2 / \ln n$;

$est_u \leftarrow 0$ for every $u \in V$;

for $i = 1$ *to* $\lfloor \log_2 n \rfloor - 1$ **do**

  $x_i \leftarrow n/2^i$;

  $\theta_i \leftarrow \lambda'/x_i$; *// λ' is defined in Eq. (3)*

  while $|\mathcal{R}| \leq \theta_i$ **do**

    Select a node v from V uniformly at random;

    $(-, u) \leftarrow$ P-RR(v, G, q, Δ, p, D); *// generate a random P-RR set pair (R^P, u), need the returned node u that is both self-activated and the earliest in reaching root v, but ignore the set R^P*

    if $u \neq -1$ **then**

      $est_u \leftarrow est_u + 1$;

  $topk \leftarrow$ sum of the top k largest values in $\{est_u\}_{u \in V}$;

  if $n \cdot topk \geq (1 + \epsilon') \cdot x_i$ **then**

    $LB \leftarrow n \cdot topk / (\theta_i \cdot (1 + \epsilon'))$;

    break;

$\theta \leftarrow \tilde{\lambda}^*/LB$; *// $\tilde{\lambda}^*$ is defined in Eq. (8)*

while $|\mathcal{R}| \leq \theta$ **do**

  Select a node v from V uniformly at random;

  $(-, u) \leftarrow$ P-RR(v, G, q, Δ, p, D);

  if $u \neq -1$ **then**

    $est_u \leftarrow est_u + 1$;

// Phase 2: obtain the top k nodes

$S \leftarrow$ set of top k nodes with the largest values in $\{est_u\}_{u \in V}$;

return S.

where Eq. (9) is by the definitions of RR set $R_W(u)$ and final activated set $\Phi^B_W(S)$. □

Proof of Theorem 3.2 (Sketch). The proof follows the same structure as the proof of IMM in [21] together with the workaround 2 summarized in [5]. Let $\mathcal{R}$ be the sequence of RR sets where every $R \in \mathcal{R}$ has no self-activated nodes, and thus $\mathcal{R}$ is the sequence generated by the IMM-BIM algorithm (Algorithm 1). Let $\mathcal{R}^S$ be the sequence of RR sets where every $R \in \mathcal{R}^S$ contains some node that is self-activated. By Lemma 3.1, we can have an unbiased estimate of boosted influence spread as $\hat{\sigma}^B(S) = n \cdot (\sum_{R \in \mathcal{R}} \mathbb{I}\{S \cap R \neq \emptyset\} + |\mathcal{R}^S|)/(|\mathcal{R}| + |\mathcal{R}^S|)$. Note that $|\mathcal{R}^S|$ is exactly maintained by variable *covered* in Algorithm 1, therefore, the above formula

matches the definition of $F^S_{\mathcal{R}}(S)$ in Eq.(5). Also notice that only for RR sets not containing self-activated nodes, we need to find seeds to cover them, and thus the procedure NodeSelection only takes $\mathcal{R}$ as the input. The rest of the analysis would follow the same way as the IMM analysis, since the boosted influence spread is monotone submodular (Lemma 2.5), same as the classical influence spread. □

Proof of Lemma 3.3.

$$\begin{aligned}
&\mathbb{E}_{W\sim\mathcal{W}(q,\Delta,p,D)}[\mathbb{I}\{S\cap R^P_W\neq\emptyset\}]\\
&=\sum_{v\in V}\Pr\{v=\text{root}(R^P_W)\}\cdot\mathbb{E}_W[\mathbb{I}\{S\cap R^P\neq\emptyset\mid v=\text{root}(R^P_W)\}]\\
&=\frac{1}{n}\sum_{v\in V}\mathbb{E}_W[\mathbb{I}\{S\cap R^P_W(v)\neq\emptyset\}]\\
&=\frac{1}{n}\sum_{v\in V}\mathbb{E}_W[\mathbb{I}\{v\in\Gamma^B_W(S)\}] \qquad (10)\\
&=\frac{1}{n}\cdot\mathbb{E}_W[|\Gamma^B_W(S)|]\\
&=\frac{1}{n}\cdot\rho^B(S),
\end{aligned}$$

where Eq. (10) is due to the definitions of the P-RR set and $\Gamma^B_W(S)$, the set of nodes that are first reached by some node in S. □

Proof of Theorem 3.4 (Sketch). Lemma 3.3 has the same form as Eq. (1), and the boosted preemptive influence spread is monotone submodular (Lemma 2.5), same as the classical influence spread. Therefore, our IMM-BPIM algorithm runs the same as IMM and achieves the same approximation guarantee result. The only difference is in the generation of P-RR sets, which affects time complexity. For a P-RR set R^P, Let R be the corresponding RR set that contains all nodes visited during the reverse simulation process. Similar to the analysis in [21], let the width of R, $\omega(R)$, be the number of incoming edges pointing to nodes in R. Let $EPT=\mathbb{E}[\omega(R)]$. In IMM, EPT would be the expected running time of generating one RR set. But in our algorithm, in particular in P-RR algorithm (Algorithm 2), we need to use a priority queue for set Q to support element insertion, deletion, update, and finding min operations, same as the implementation of the Dijkstra shortest path algorithm. Therefore, the expected running time of generating one P-RR set is $O(\mathbb{E}[\omega(R)+|R|\log|R|])=O(\mathbb{E}[\omega(R)\log n])=O(EPT\log n)$, leading to an extra $\log n$ factor. From [21], we further know that $EPT=m\mathbb{E}[\sigma(\tilde{v})]/n$, where $\tilde{v}$ is a node randomly selected with probability proportional to its indegree, and $\sigma(\tilde{v})$ is the classical influence spread of $\tilde{v}$ in the corresponding IC model. Let θ be the number of P-RR sets generated, and let $\text{OPT}=\rho^B(S^*)$ be the optimal solution for the BPIM problem. Based on the IMM analysis in [21], we know that the total expected running time is given by

$$\begin{aligned}
&O(\mathbb{E}[\theta]\cdot\mathbb{E}[\omega(R)+|R|\log|R|])\\
&=O\left(\frac{(k+\ell)n\log n}{\text{OPT}\cdot\varepsilon^2}\cdot EPT\log n\right)\\
&=O\left(\frac{(k+\ell)(m+n)\log^2 n}{\varepsilon^2}\cdot\frac{\mathbb{E}[\sigma(\tilde{v})]}{\text{OPT}}\right).
\end{aligned}$$

Finally, by our assumption that the optimal solution is at least as large as $\mathbb{E}[\sigma(\tilde{v})]$, the time complexity is as stated in the theorem. □

Proof of Lemma 3.5.

$$\begin{aligned}
&\mathbb{E}_{W\sim\mathcal{W}(q,\Delta,p,D)}[\mathbb{I}\{u=u^s_W\}]\\
&=\frac{1}{n}\sum_{v\in V}\cdot\mathbb{E}[\mathbb{I}\{u=u^s_W(v)\}]\\
&=\frac{1}{n}\sum_{u\in V}\mathbb{E}[\mathbb{I}\{v\in\Gamma_W(\{u\})\}]\\
&=\frac{1}{n}\cdot\mathbb{E}[|\Gamma_W(u)|]\\
&=\frac{1}{n}\cdot\rho(\{u\}).
\end{aligned}$$

□

Proof of Theorem 3.6 (Sketch). By Lemma 2.5, the preemptive influence spread function ρ is additive. Therefore, we can find top k nodes with the largest preemptive influence spread individually, and together they form the optimal solution for the PIM problem. To estimate individual node's preemptive influence spread, we utilize Lemma 3.5, such that the number of times a node u is identified as the source node u^s by the P-RR algorithm directly corresponds the preemptive influence spread $\rho(u)$. Therefore, we only need to replace the NodeSelection procedure with counting the appearances of each node as sources (via variable est_u) and selecting the top k of them. The parameter $\lambda^*(\ell)$ in the original IMM algorithm is also replaced with $\tilde{\lambda}^*$ by replacing $(1-1/e)$ with 1, because we find exact solution of top k nodes appearing in most reverse simulations instead of a $1-1/e$ approximation. With this, following the proof structure of the IMM algorithm, we can show that our IMM-PIM algorithm guarantees $1-\varepsilon$ approximation.

As for time complexity, the analysis follows the same way as in the proof of Theorem 3.4. That is, because of P-RR algorithm needs to use a priority queue to run a Dijkstra-like algorithm, we need an extra factor $\log n$ in the time complexity. □

C ADDITIONAL EXPERIMENTAL RESULTS

Self-activation parameters and test cases. In practice, self-activation delays can be estimated from the users' access patterns to online social networks, and self-activation probabilities can be estimated from the fraction of times users' participating in information cascades not due to the influence from the neighbors or external selections as seed users. Unfortunately for the datasets we use, these information are not available. Instead, we use synthetic settings, focusing on whether the knowledge of the self-activation behaviors would benefit our algorithm design. For self-activation probability $q(u)$ of node u, we first randomly select a value β_u from $[0,c]$ as a node u's base value, then we further test five cases: (0) uniform: $q(u)=\alpha_u$; (1) positively correlated: $q(u)$ is positively correlated with u's out-degree $d^+(u)$, in particular $q(u)=\min\{\beta_u\cdot d^+(u),1\}$; (2) negatively correlated: $q(u)=\beta_u/d^+(u)$; (3) random mixing of cases 0 and 1: randomly pick half of the nodes with $q(u)=\alpha_u$ and the other half with $q(u)=\min\{\beta_u\cdot d^+(u),1\}$; (4) random mixing of cases 0 and 2: randomly pick half of the nodes with $q(u)=\alpha_u$ and the other half with $q(u)=\beta_u/d^+(u)$. These five cases aim at scenarios where all users are equally likely to react to a campaign (case 0), high-degree nodes (usually more influential) are more likely or less likely to react to the campaigns, and mixture of uniform

Table 2: The performance of influence spread on NetHEPT.

NetHEPT	Case	Method	10	50	100	200
PIM	0	IMM	23.35 ± 0.40	75.19 ± 0.64	130.78 ± 0.78	208.24 ± 0.91
		ASV-RR	25.09 ± 0.41	73.30 ± 0.64	131.96 ± 0.79	220.53 ± 0.96
		IMM-PIM	30.32 ± 0.40 (+20.1%)	108.44 ± 0.71 (+44.2%)	184.17 ± 0.88 (+39.6%)	289.16 ± 1.08 (+31.1%)
	1	IMM	59.26 ± 0.41	188.02 ± 0.69	323.70 ± 0.86	506.95 ± 1.01
		ASV-RR	62.37 ± 0.41	185.44 ± 0.68	324.81 ± 0.86	534.16 ± 1.04
		IMM-PIM	75.18 ± 0.42 (+20.5%)	268.35 ± 0.73 (+42.7%)	440.31 ± 0.91 (+35.6%)	705.06 ± 1.10 (+32.0%)
	2	IMM	7.71 ± 0.27	25.33 ± 0.43	45.42 ± 0.54	76.45 ± 0.62
		ASV-RR	8.04 ± 0.28	25.00 ± 0.44	44.81 ± 0.55	79.03 ± 0.69
		IMM-PIM	10.66 ± 0.25 (+55.8%)	39.96 ± 0.48 (+86.5%)	69.877 ± 0.61 (+69.1%)	123.14 ± 0.70 (+56.9%)
	3	IMM	49.93 ± 0.44	135.20 ± 0.67	234.35 ± 0.84	371.37 ± 0.98
		ASV-RR	43.23 ± 0.41	139.90 ± 0.67	245.02 ± 0.85	407.87 ± 1.04
		IMM-PIM	77.81 ± 0.48 (+32.6%)	260.85 ± 0.79 (+57.8%)	414.33 ± 0.97 (+53.8%)	640.14 ± 1.15 (+55.8%)
	4	IMM	12.24 ± 0.30	53.80 ± 0.56	92.24 ± 0.69	145.59 ± 0.79
		ASV-RR	18.88 ± 0.38	49.15 ± 0.56	89.53 ± 0.70	148.14 ± 0.84
		IMM-PIM	29.82 ± 0.44 (+57.9%)	97.93 ± 0.70 (+82.0%)	159.18 ± 0.82 (+72.6%)	258.13 ± 0.96 (+74.2%)
BIM	0	IMM	3976.01 ± 2.01	4295.73 ± 1.86	4606.85 ± 1.77	5094.04 ± 1.69
		IMM-BIM	3976.77 ± 2.01 (< 1%)	4299.85 ± 1.90 (< 1%)	4612.03 ± 1.81 (< 1%)	5097.34 ± 1.70 (< 1%)
	1	IMM	5526.53 ± 1.83	5678.88 ± 1.79	5840.28 ± 1.68	6144.55 ± 1.61
		IMM-BIM	5547.92 ± 1.83 (< 1%)	5744.09 ± 1.78 (+1.1%)	5938.97 ± 1.72 (+1.7%)	6270.64 ± 1.66 (+2.1%)
	3	IMM	4829.76 ± 1.91	5047.73 ± 1.82	5272.48 ± 1.74	5656.25 ± 1.65
		IMM-BIM	4841.66 ± 1.91 (< 1%)	5087.22 ± 1.85 (< 1%)	5330.86 ± 1.78 (+1.1%)	5739.56 ± 1.68 (+1.5%)
BPIM	0	IMM	188.14 ± 0.63	677.81 ± 0.96	1128.41 ± 1.07	1813.26 ± 1.16
		IMM-BPIM	189.12 ± 0.64 (< 1%)	677.67 ± 0.94 (< 1%)	1134.77 ± 1.05 (< 1%)	1819.71 ± 1.14 (< 1%)
	1	IMM	127.04 ± 0.43	508.31 ± 0.72	893.64 ± 0.86	1509.87 ± 1.00
		IMM-BPIM	132.34 ± 0.42 (+4.2%)	517.91 ± 0.70 (+1.9%)	909.68 ± 0.84 (+1.8%)	1530.55 ± 0.96 (+1.4%)
	3	IMM	153.42 ± 0.51	578.23 ± 0.80	995.54 ± 0.95	1646.73 ± 1.08
		IMM-BPIM	155.73 ± 0.49 (+1.5%)	583.91 ± 0.79 (+1.0%)	1009.91 ± 0.94 (+1.4%)	1667.51 ± 1.04 (+1.3%)

Table 3: The performance of influence spread on Flixster.

Flixster	Case	Method	10	50	100	200
PIM	0	IMM	32.03 ± 0.39	110.89 ± 0.6	166.58 ± 0.68	252.28 ± 0.78
		ASV-RR	31.59 ± 0.39	119.27 ± 0.64	170.25 ± 0.71	281.37 ± 0.84
		IMM-PIM	44.1 ± 0.43 (+37.7%)	142.37 ± 0.65 (+19.4%)	223.52 ± 0.73 (+31.3%)	339.79 ± 0.82 (+20.8%)
	3	IMM	51.68 ± 0.33	165.15 ± 0.51	259.12 ± 0.59	399.44 ± 0.67
		ASV-RR	54.5 ± 0.33	175.98 ± 0.53	256.63 ± 0.6	442.88 ± 0.69
		IMM-PIM	105.39 ± 0.34 (+93.4%)	311.99 ± 0.54 (+77.3%)	475.21 ± 0.64 (+83.4%)	702.76 ± 0.71 (+58.7%)
BIM	0	IMM	4693.78 ± 1.13	5041.03 ± 1.02	5315.41 ± 0.96	5712.07 ± 0.92
		IMM-BIM	4694.19 ± 1.14 (< 1%)	5046.5 ± 1.04 (< 1%)	5327.35 ± 0.98 (< 1%)	5718.48 ± 0.93 (< 1%)
	3	IMM	6892.51 ± 1.01	7050.49 ± 0.96	7200.86 ± 0.92	7471.4 ± 0.9
		IMM-BIM	6898.82 ± 1.01 (< 1%)	7081.8 ± 0.98 (< 1%)	7258.11 ± 0.96 (< 1%)	7543.74 ± 0.91 (+1.0%)
BPIM	0	IMM	294.41 ± 0.49	877.56 ± 0.7	1293.56 ± 0.74	1860.55 ± 0.76
		IMM-BPIM	299.74 ± 0.48 (+1.8%)	887.56 ± 0.68 (+1.1%)	1307.58 ± 0.71 (+1.1%)	1875.77 ± 0.72 (< 1%)
	3	IMM	194.11 ± 0.34	599.17 ± 0.54	925.57 ± 0.61	1414.42 ± 0.65
		IMM-BPIM	201.02 ± 0.34 (+3.6%)	625.58 ± 0.53 (+4.4%)	972.24 ± 0.58 (+5.0%)	1473.65 ± 0.64 (+4.2%)

Table 4: Running time results on datasets (in seconds).

Data	IMM-BIM	IMM-PIM	IMM-BPIM	IMM	ASV-RR
NetHEPT	0.7534	195.75	48.123	1.9712	74.175
Flixster	1.3516	955.45	218.51	5.1072	235.34

Table 5: Running time results on ε value (in seconds).

NetHEPT (ε)	0.1	0.2	0.3	0.4	0.5
IMM-BIM	5098.01 (0.73s)	5075.54 (0.28s)	5063.49 (0.21s)	5037.64 (0.18s)	5004.03 (0.13s)
IMM-PIM	289.16 (492.49s)	290.67 (136.1s)	287.61 (67.5s)	288.22 (41.1s)	287.33 (29.2s)
IMM-BPIM	1819.19 (53.6s)	1811.79 (14.3s)	1792.67 (7.1s)	1781.42 (4.7s)	1768.05 (3.4s)

Table 6: Running time results on self-activation probability distribution (in seconds)

α_u	[0-0.2]	[0.2-0.4]	[0.4-0.6]	[0.6-0.8]	[0.8-1.0]
IMM-BIM	0.839	0.456	0.328	0.259	0.226
IMM-BPIM	53.92	63.47	73.22	117.03	126.82
IMM-PIM	492.49	378.04	347.67	314.58	296.72

behavior and a correlation (or reverse correlation) behavior. We set $c = 2$ for BPIM and PIM tests, because otherwise the preemptive influence spread for PIM is too small. For self-activation delays, we use exponential distribution with rate 1 for all nodes. We already vary the self-activation behaviors through the self-activation probabilities, and thus we simply keep the self-activation delay distributions uniform. We also use the same exponential distribution for propagation delay distributions.

C.1 Effectiveness analysis

We provide the full test case results for NetHEPT in Table 2. Each influence spread result is an average over 10000 simulations runs for NetHEPT and Flixter The number following the ± sign is the 95% confidence radius, and the percentage in parenthesis is the improvement of our algorithm over the best baseline result given above it. The results on PIM clearly show that our IMM-PIM algorithm has improvement in the preemptive influence spread achieved comparing against baselines IMM and ASV-RR — the improvements is from 20% to 98.1%. This shows that, when identifying top influencers in the presence of self activation behavior, it is important to incorporate the knowledge about self activation into the algorithm design

For BIM and BPIM, the improvement is less significant. In particular, the uniform case does not show significant improvement, while for positively correlated cases, the improvement could be 1% to 4%. This suggests that, due to the boost of seed nodes, the knowledge of natural self-activation behaviors become less important, and some times it is ok to use the self-activation oblivious IMM algorithm, but if a few percentage of improvement is still important, the extra knowledge on self-activation together with our algorithms are still beneficial. We ignore cases 2 and 4, and in general the improvement is less significant.

The results on Flixster are similar, we list the cases 0 and 3 results in Tables 3 which show significant improvements over IMM and ASV-RR for PIM and outperform IMM for BIM and BPIM.

C.2 Running time analysis

Table 4 reports the running time of all algorithms on both datasets, by using the default setting with the seed set size $k = 200$ in test case 3. We can clearly see the order of running time is IMM-BIM < IMM < IMM-BPIM < ASV-RR < IMM-PIM (we ignore the prefix IMM in our algorithm to fit the table width). This is inline with our theoretical analysis, which shows that the running time is inversely proportional to the optimal value of each problem. For example, the optimal solution of BIM is larger than that of the classical influence maximization because BIM has self-activated nodes contributing extra influence spread, and PIM has the smallest optimal value because the self-activation probabilities are small in general and the optimal set has to compete with other self-activated nodes on preemptive influence spread. IMM-BPIM and IMM-PIM are further slower due to the Dijkstra-like reverse simulation, which takes more time than the simple breadth-first-search simulation. But even for the slowest IMM-PIM algorithm, on the Flixster dataset with more than hundred thousands nodes and edges, it could complete in less than 16 minutes on our laptop test machine. Besides the dataset, running time is also related to the ε.

As shown in Table 5, we test ε values from 0.1 to 0.5 on NetHEPT for the proposed algorithms. We can see influence difference is less than 3% when ε increases from 0.1 to 0.5 for IMM-BIM and IMM-BPIM, and there is almost no influence difference for IMM-PIM, which indicates it could be accelerated by increasing ε.

Besides the dataset and ε, the distribution of the self-activation probability $q(u)$ would infect the running time of the proposed algorithm. We randomly select value from different uniform self-activation probability distributions $[0, 0.2]$, $[0.1, 0.3]$, $[0.2, 0.4]$, $[0.3, 0.5]$, $[0.4, 0.6]$ for the proposed algorithms. From Table 6, we can clearly see that for both IMM-BIM and IMM-PIM, they spend less running time with larger self-activation probability distribution. On the contrary, IMM-BPIM runs faster with a smaller distribution of self-activation probabilities. In summary, the running times of the proposed algorithms are sensitive to the self-activation probability distribution of the nodes.