Debunking the Myths of Influence Maximization: An In-Depth Benchmarking Study

Akhil Arora[1*] Sainyam Galhotra[2] Sayan Ranu[3]
[1]Text and Graph Analytics, Xerox Research Centre India, Bangalore, India
[2]College of Information and Computer Science, University of Massachusetts, Amherst, USA
[3]Dept. of Computer Science and Engineering, Indian Institute of Technology, Delhi, India
[1]akhil.arora@xerox.com [2]sainyam@cs.umass.edu [3]sayanranu@cse.iitd.ac.in

ABSTRACT

Influence maximization (IM) on social networks is one of the most active areas of research in computer science. While various IM techniques proposed over the last decade have definitely enriched the field, unfortunately, experimental reports on existing techniques fall short in validity and integrity since many comparisons are not based on a common platform or merely discussed in theory. In this paper, we perform an in-depth benchmarking study of IM techniques on social networks. Specifically, we design a benchmarking platform, which enables us to evaluate and compare the existing techniques systematically and thoroughly under identical experimental conditions. Our benchmarking results analyze and diagnose the inherent deficiencies of the existing approaches and surface the open challenges in IM even after a decade of research. More fundamentally, we unearth and debunk a series of myths and establish that there is no single state-of-the-art technique in IM. At best, a technique is the state of the art in only one aspect.

1. INTRODUCTION

Social networks have become an integral part of our day-to-day lives. We rely on Facebook and WhatsApp to communicate with friends. Twitter is regularly used to disseminate information such as traffic news, emergency services, etc. This reliance on social networks has resulted in wide-spread research in finding solutions to the *influence maximization (IM)* problem [3–9, 14–18, 20, 21, 23, 24, 26, 27]. In a social network, each user corresponds to a node and two users are connected through an edge if they *interact*. Interaction between two users may depict friendship, such as in FaceBook, *following* a user, such as in Twitter, or co-authorship of scientific articles, such as in DBLP. Generally, it is assumed that an user u can *directly* influence user v if u interacts with v, i.e., there is an edge from u to v. For example, u positing a positive review on a movie may result in v actually watching the movie. This event may in turn result in v influencing his/her own friends. The IM problem is to identify a set of *seed nodes* so that the total number of users influenced is maximized.

*The first two authors have contributed equally to this work.

SIGMOD'17, May 14 - 19, 2017, Chicago, Illinois, USA

ISBN 978-1-4503-4197-4/17/05…$15.00

DOI: http://dx.doi.org/10.1145/3035918.3035924

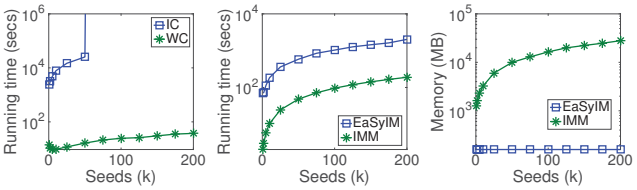


Figure 1: **(a) Running time of IMM ($\epsilon = 0.5$) under IC ($W(u,v) = 0.1$) and WC on the Orkut dataset. (b-c) Comparing IMM ($\epsilon = 0.5$) with EaSyIM ($iter = 100$) under the IC model ($W(u,v) = 0.1$) on the YouTube dataset.**

Kempe et al. [17] in their seminal work established that finding an optimal solution for IM is NP-Hard and were the first to prove that a simple GREEDY algorithm can provide the best approximation guarantee in polynomial time. They incorporated the use of three *diffusion models – Independent Cascade* (IC), *Weighted Cascade* (WC) and *Linear Threshold* (LT) for information propagation, which have been almost exclusively followed in majority of the subsequent work. All these models are essentially a function of the edge weights in the social network. The higher the edge weight between u and v, more is the influence of u on v.

Since Kempe et al.'s seminal work [17], almost every year, a new IM technique has been published that claim to be the state-of-the-art. Without doubt, this extensive research has promoted prosperity of the family of IM techniques. However, it also raises several questions that are not adequately addressed. *How to choose the most appropriate IM technique in a given specific scenario? What does it really mean to claim to be the state-of-the-art? More fundamentally, are the claims made by the recent papers true?* To ensure a streamlined growth of the field, it is critical to benchmark the existing techniques in a unified setup across common datasets and answer all of the above questions. We conduct this benchmarking study and expose a series of myths that could potentially alter the way we approach IM research.

To highlight the ambiguity that plagues the current maze of IM techniques, we provide some concrete examples to motivate the need for a benchmarking study.

IC Vs. WC. WC is a specialized instance of the IC diffusion model and not IC itself. We discuss their differences in detail in Sec. 2.1. Multiple techniques [26, 27] have claimed to scale well under IC, while in reality, they scale only for WC. To provide a concrete example, consider Fig. 1a, where IMM [26] is highly efficient and scales well for WC, however, is relatively inefficient under IC. In fact, it crashes on our machine beyond 50 seeds for IC on the Orkut dataset, while consuming more than 256 GB of RAM.

While IMM is just one representative technique, several tech-

niques equate WC with IC and make a generic claim to perform well under the IC diffusion model.

What does it mean to be the state of the art? While many techniques claim to be the state of the art, in reality, they are often the state of the art in only one aspect of the IM problem. Consider Figs. 1b-1c. On the same YouTube dataset, while EaSyIM [10] scales better with respect to memory, IMM [26] scales better with running time. Thus, neither technique can be termed as better than the other. More critically, the IM community should also serve as a decision maker. Specifically, given a problem scenario, we should be able to identify the technique that is best suited for that purpose. Our study bridges this gap.

To summarize, IM is associated with a *host of parameters*: from the choice of the *diffusion model*, to the choice of the optimization objective – (1) *quality*, (2) *computational efficiency*, (3) *memory consumption* or any possible combination of the three. Owing to this huge parameter space it is difficult to identify a technique that is optimal across all the parameters. Consequently, a benchmarking study is the need of the day. This benchmarking exercise however is non-trivial and poses an array of unique challenges.

- Given the large diversity in the strategies employed by the various IM techniques, a universal framework for IM is quite difficult to be summarized and abstracted. Nonetheless, building this framework is critical to compare, analyze and diagnose the techniques from a common viewpoint.
- In this study, we need to either gather code from the authors or re-implement them. Given the large body of work, this itself is a significant challenge. Furthermore, to integrate them into the benchmarking framework and interpret the results, it is critical to have an in-depth understanding of the code.
- When proposing a new approach, authors often testify their work across limited metrics, datasets, and parameters, on which they perform well. To evaluate the techniques as comprehensively as possible, we need to identify a suite of metrics, various classes of datasets, and a diverse set of parameters that can characterize all aspects of the IM problem.

In this paper, we have designed a systematic benchmarking platform for the IM problem. As visible in Fig. 2, the benchmark consists of four core components: (1) **Setup**, including a set of algorithms, real-world datasets, parameter configurations and a diffusion model; (2) **IM Framework**, a generalized IM module with high abstraction of the common workflow of Influence Maximization (details in Sec. 3 and 4); (3) **Evaluation**, which provide targeted diagnoses on these algorithms based on our framework, leading to directions of improvement over the existing work (Sec. 5); and (4) **Insights**, which discusses the key take-away points from the benchmarking study and generally, summarizes the state of the IM field after more than a decade of research (Sec. 6).

To summarize, the contributions of our work are as follows.

- We conduct a comprehensive benchmarking study that performs an in-depth analysis, evaluation and comparison of the extensive work on IM (Sec. 5). [1]
- We review the family of IM techniques (Sec. 4, 5) and have curated the most comprehensive publicly accessible code base. The code base can be downloaded from https://sigdata.github.io/infmax-benchmark.
- We unearth and debunk a series of myths that would shape the future of IM research (Sec. 6). Furthermore, we draw a set of interesting take-away conclusions, and identify the *skyline* techniques for IM (Sec. 7).

[1]Our benchmarking study considers all IM techniques published till May, 2016. Thus, we could not include some of the more recent techniques, such as [23].

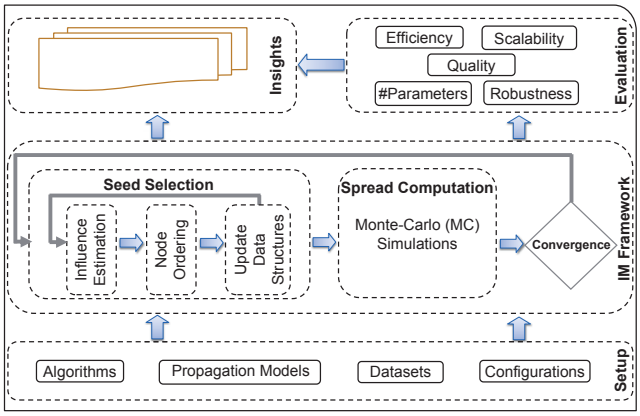


Figure 2: **The proposed benchmarking framework for IM.**

2. PRELIMINARIES

DEFINITION 1 (SOCIAL NETWORK.). *A social network with n individuals and m social ties can be denoted as an edge-weighted graph $G(V, E, W)$, where V is the set of nodes, $|V| = n$, and E is the set of directed relationships, $E \subseteq V \times V$, $|E| = m$, and W is the set of edge-weights corresponding to each edge in E.*

We use the notations $W(u, v)$ to denote the weight of edge $e = (u, v)$. **In**(v) and **Out**(v) denote the set of incoming and outgoing neighbors of vertex v respectively.

The objective in *influence maximization (IM)* is to maximize the *spread* of information (or influence) in a network through activation of an initial set of k *seed* nodes. The dynamics of how information spreads in a network is guided by an *information diffusion model*. To this end, we first define the notions of *seed* and *active* nodes. Next, we use these concepts to define the notion of *spread* of information in a network and the most popular information diffusion models that have been studied in literature.

DEFINITION 2 (SEED NODE). *A node $v \in V$ that acts as the source of information diffusion in the graph $G(V, E, W)$ is called a seed node. The set of seed nodes is denoted by S.*

DEFINITION 3 (ACTIVE NODE). *A node $v \in V$ is deemed active if either (1) It is a seed node ($v \in S$) or (2) It receives information under the dynamics of an information diffusion model $\mathcal{I}$, from a previously active node $u \in V_a$. Once activated, the node v is added to the set of active nodes V_a.*

Given a seed node $s \in S$ and a graph $G(V, E, W)$, an information diffusion model $\mathcal{I}$ defines a step-by-step process for information propagation. *Independent Cascade (IC)* and *Linear Threshold (LT)* are the two most well-studied information diffusion models. For both the IC and LT models, the first step requires a seed node $s \in S$ to be activated and added to the set of active nodes V_a.

DEFINITION 4 (INDEPENDENT CASCADE). *Under the IC model, time unfolds in discrete steps. At any time-step i, each newly activated node $u \in V_a$ gets one independent attempt to activate each of its outgoing neighbors $v \in$ **Out**(u) with a probability $p_{(u,v)} = W(u, v)$. In other words, $W(u, v)$ denotes the probability of u influencing v.*

DEFINITION 5 (LINEAR THRESHOLD). *Under the LT model, every node v contains an activation threshold θ_v, which is chosen uniformly at random from the interval $[0, 1]$. Further, LT dictates that the summation of all incoming edge weights is at most 1, i.e., $\sum_{\forall u \in \textbf{In}(v)} W(u, v) \leq 1$. v gets activated if the sum of weights*

Algorithm 1 Spread

Input: Graph $G = (V, E, W)$, seed-set S_0, diffusion model $\mathcal{I}$
1: $i \leftarrow 0$
2: **repeat**
3: $\quad i \leftarrow i + 1$
4: $\quad A \leftarrow$ compute the newly active nodes at time-step i under $\mathcal{I}$
5: $\quad S_i \leftarrow S_{i-1} \cup A$
6: **until** $S_i - S_{i-1} = \emptyset$
7: $V_a \leftarrow S_i$
8: **Return** V_a

Algorithm 2 GreedyIM

Input: Graph $G = (V, E, W)$, k, diffusion model $\mathcal{I}$
1: $S \leftarrow \emptyset$
2: $i \leftarrow 0$
3: **while** $(i < k)$ **do**
4: $\quad i \leftarrow i + 1$
5: $\quad v^* \leftarrow \arg\max_{\forall v \in V} \{\sigma(S \cup \{v\}) - \sigma(S)\}$ under $\mathcal{I}$
6: $\quad S \leftarrow S \cup \{v^*\}$
7: **end while**
8: **Return** S

$W(u, v)$ of all the incoming edges (u, v) originating from active nodes exceeds the activation threshold θ_v. Mathematically,

$$\sum_{\forall u \in \boldsymbol{In}(v) \cap V_a} W(u, v) \geq \theta_v \tag{1}$$

Alg. 1 outlines the pseudocode of how information spreads. The information spreads in discrete time steps. Let S_0 be the initial set of seed nodes at time step 0. These seeds further activate newer nodes based on the diffusion model $\mathcal{I}$ at time step 1 (line 4). Once a node becomes active, it remains active in all the subsequent steps. This diffusion process continues until no more activations are possible, i.e., the active nodes in two consecutive time steps are same (line 6). The *spread* of the information is essentially the total number of nodes (including the seed nodes) that get activated (line 8).

DEFINITION 6 (SPREAD). *Given an information diffusion model $\mathcal{I}$, the spread $\Gamma(S)$ of a set of seed nodes S is defined as the total number of nodes that are active, including both the newly activated nodes and the initially active set S, at the end of the information diffusion process. Mathematically, $\Gamma(S) = |V_a|$.*

Since the information diffusion under various models (IC/LT) is defined as a stochastic process the measure of interest is the *expected* value of spread. The *expected* value of spread $\sigma(\cdot) = \mathbb{E}[\Gamma(\cdot)]$ is computed by performing a high number (typically 10,000 [17]) of MC simulations of the spread function. The goal in IM, is therefore to solve the following problem.

PROBLEM 1 (INFLUENCE MAXIMIZATION (IM)). *Given an integer value k and a social network G, select a set S of k seeds, $S \subseteq V \mid k = |S|$, such that the expected value of spread $\sigma(S) = \mathbb{E}[\Gamma(S)]$ is maximized.*

2.1 Edge weights in Diffusion Models

It is easy to see that the edge weights define the extent to which information propagates in the network. Ideally, the edge weights should be learned from some training data and such efforts exist [12, 13, 19]. However, in this study we incorporate the use of the standard edge-weighting mechanisms corresponding to the IC and LT models. The reason for this is three fold: (1) such a rich set of training data is not readily available for the wide-variety of publicly available networks; (2) the performance (quality, efficiency and scalability) of the existing IM algorithms vary significantly with network properties, scale etc. (details in Sec. 5), thus, an effective benchmarking requires evaluation across a wide-range of networks; and (3) most of the existing works on IM assign edge weights based on some model rather than learning them.

Next, we introduce the various models that have been used to assign edge weights under the IC and LT models respectively.

2.1.1 Independent Cascade (IC)

• **Constant:** In this model, each edge $W(u, v)$ has a constant probability p. In vast majority of the IM techniques, p takes the value of either 0.01 or 0.1 [4, 9–11, 14, 17]. Additionally, some techniques use a spectrum of values for $p \in [0.01, 0.1]$ [5, 24].

• **Weighted Cascade (WC):** In WC, $p(u, v) = W(u, v) = \frac{1}{|\mathbf{In}(v)|}$. In other words, all incoming neighbors of v influence v with equal probability. As a consequence, it is easier to influence low-degree nodes than high-degree nodes. [4, 5, 7–11, 14, 16, 17, 26, 27]

• **Tri-valency Model:** In this model, the probability (or weight) on an edge is chosen randomly from a set of probabilities. For example, the weight may be chosen randomly from the set $\{0.001, 0.01, 0.1\}$. [4, 7, 16]

2.1.2 Linear Threshold (LT)

• **Uniform:** The edge-weight on each edge $e = (u, v)$ is assigned to $W(u, v) = \frac{1}{|\mathbf{In}(v)|}$. This is similar to the WC model in IC. [6, 10]

• **Random:** In this model, each edge is assigned a value uniformly at random in the range [0,1]. Finally, these edge values are normalized to generate the edge weights so that the weights of all incoming edges to a node v sum to 1. [6, 26, 27]

• **Parallel Edges:** Although social networks are generally modeled as a graph, in certain cases, they are actually *multi-graphs*. A multi-graph allows multiple edges between a pair of nodes. Consider a phone-call network, where each node corresponds to a user, and an edge (u, v) models u calling v. Since u may call v more than once, there may be *parallel* edges between these users. Now, to apply LT on such multi-graphs, parallel edges are consolidated into a single edge to form a traditional graph. In this graph, each edge (u, v) has the weight $W(u, v) = \frac{c(u,v)}{\sum_{\forall u' \in \mathbf{In}(v)} c(u', v)}$, where $c(u, v)$ is the number of parallel edges from u to v. More simply, it is a generalization of the Uniform model for multi-graphs. [15]

2.2 Properties of the IM problem

THEOREM 1. *The problem of IM, as defined in Problem 1 is NP-hard under both IC and LT.*

PROOF. Please see [17]. □

Fortunately, it has been shown that the spread function $\Gamma(\cdot)$, and its expectation $\sigma(\cdot) = \mathbb{E}[\Gamma(\cdot)]$, is *monotone* and *submodular* [17]. If the function is submodular and monotone, the greedy hill-climbing algorithm of iteratively choosing the element with maximal marginal gain approximates the optimal solution within a factor of $(1 - \frac{1}{e})$ [22]. This result can be utilized to design the greedy seed selection algorithm shown in Alg. 2.

THEOREM 2. *The expected value of spread computed using the seed set returned by Alg. 2 is within $(1 - \frac{1}{e} - \epsilon)$ of the optimal. Mathematically,*

$$\sigma(S) \geq \left(1 - \frac{1}{e} - \epsilon\right) \sigma(S^*) \tag{2}$$

where S is the seed set computed by Alg. 2 and S^* is the optimal seed set. Furthermore, $\sigma(S)$ is the best possible approximation in polynomial time.

PROOF. Please see [17]. □

While Alg. 2 does provide a polynomial time solution, it is still not scalable for even moderate sized networks. The non-scalability stems from line 5, where the node providing the highest marginal gain is provided. First, the spread due to addition of a seed node cannot be computed deterministically since it is a function of probabilities guided by the underlying diffusion model. The standard approach is therefore to simulate the information cascading procedure a high number of times and then report the expected value of spread. As recommended by Kempe et al. [17], the number of simulations is typically set to 10,000 [17]. Second, this expensive spread computation needs to be repeated for each node in the network. This completes the operation of computing v^* in one iteration. In all k subsequent seed selection iterations (line 3), this entire procedure is repeated making even the greedy pipeline non-scalable. This non-scalability forms the crux of all subsequent works on IM.

3. THE BENCHMARKING FRAMEWORK

Having motivated the need of a generic benchmark for IM in Sec. 1, in this section we present our proposed benchmarking framework with a detailed description of a generalized procedure for influence maximization.

3.1 The Generalized IM Module

To evaluate the existing IM approaches from a common standpoint, we formulate a generalized procedure for influence maximization in this framework. As illustrated in the "IM Framework" module in Fig. 2, the procedure consists of three phases, including *seed selection*, *spread computation* and *convergence check*; and characterizes the generic workflow of influence maximization via a series of the key steps involved. Alg. 3 presents the details of the procedure. M denotes the IM algorithm under consideration, while $\mathcal{P}$ is used to represent the ordered set of associated external parameters.

3.1.1 Seed Selection

In this phase, nodes of the graph G are added iteratively to the set of seed nodes S in the order of their influence. Thereby, we abstract two key steps for this phase, namely – `InfluenceEstimate` (line 4) and `UpdateDataStructures` (line 7). First of all, the influence values of each node $v \in V$ of the graph G is estimated via `InfluenceEstimate` and collected in the set Inf_V (line 4). Note that this procedure takes M as input, and the exact approach for influence estimation may vary from approach to approach, which we describe in detail in Sec. 4. Once the set Inf_V is populated, we then choose the node with the highest estimated influence v^* (line 5) and add it to the set of seed nodes S (line 6). Since the design of this procedure follows the greedy paradigm, after every addition made to the set of seed nodes S, the estimated influence of each node in the graph G is adjusted via `UpdateDataStructures` (line 7). This is done to discount the effect of nodes selected as seeds in the previous iterations on the current seed selection step.

3.1.2 Spread Computation

This phase takes as input the set of nodes identified as seeds during the seed selection phase. Since information diffusion under various models $\mathcal{I}$ is defined as a stochastic process, the purpose of this phase is to compute a (near accurate) estimate of the expected value of spread $\sigma(S)$ of the set of identified seed nodes S. To this end, the procedure calculates $\sigma(S)$ by performing r Monte Carlo (MC) simulations of the information diffusion process via `ComputeSpread` (line 9). r is a large number (typically $10,000$ [17]).

Algorithm 3 IMFramework

Input: $G = (V, E, W)$, $\mathcal{I}$, Algorithm $(M, \mathcal{P})$, Seeds k, MC Simulations r
Output: $S, \sigma(S)$

```
1: for α_i ∈ P = {α_1, ..., α_|P|} do
2:     S_{α_i} ← ∅
3:     while |S_{α_i}| < k do
4:         Inf_V ← InfluenceEstimate(G, M_α)
5:         v* ← arg max_{∀v∈V} (Inf_V)
6:         S_{α_i} ← S_{α_i} ∪ {v*}
7:         UpdateDataStructures(G, M_α)
8:     end while
9:     σ(S_{α_i}) ← ComputeSpread(G, S_{α_i}, I, r)
10:    if ¬Converged(M, σ(S_{α_1}), σ(S_{α_i})) then
11:        Return S_{α_{i-1}}, σ(S_{α_{i-1}})
12:    end if
13: end for
14: Return S_{α_P}, σ(S_{α_P})
```

3.1.3 Convergence

Majority of IM algorithms M possess an external parameter which controls its accuracy. As illustrated in Alg. 3, this parameter can possess values from a broad spectrum $\mathcal{P} = \{\alpha_1, \ldots, \alpha_{\mathcal{P}}\}$, sorted in non-increasing order of their obtained accuracy. The more stringent the choice of this parameter the better the accuracy. Consequently, the more the running time. To effectively achieve the best scalability-accuracy tradeoff for each algorithm M, we perform the two key phases of IM, namely – (1) Seed Selection and (2) Spread Computation, across the previously discussed broad spectrum of parameter values $\alpha_i \in \mathcal{P}$ until we identify the largest value of i for which convergence holds, which is tested via `Converged` (lines 10–12). The optimal parameter selection procedure is further discussed in detail in Sec. 5.

4. IM ALGORITHMS

Influence maximization (IM) being one of the most actively studied topics in computer science, has witnessed a plethora of effective approaches over the last decade. We categorize the existing approaches according to their seed selection process, as shown in Fig. 3 which covers most representatives from all popular approaches proposed.

The first category of approaches, such as GREEDY [17], CELF [20] and CELF++ [14], rely on the explicit **MC simulations** of the information diffusion process to estimate the influence of each node, thereby selecting the seed set. Although possessing theoretical guarantees on quality, these approaches are not scalable.

Conversely, the second category relies on sampling based exploration instead of estimating the influence of each node explicitly, thereby, speeding up the seed selection step.

Methods such as RIS [3], TIM$^+$ [27] and IMM [26], construct **Reverse Reachable** (RR) sets on nodes sampled from the graph G and rely on the idea that the higher the number of RR sets a node is contained in, the more influential this node is. This process indeed qualifies as the seed selection step for this category.

On the other hand, **Snapshots** based methods, such as StaticGreedy (SG) [8] and PMC [24], generate various instantiations G_i (called snapshots) of graph G apriori, by retaining edges proportional to their edge-weight probability, and estimate node influence by averaging over all generated snapshots.

Lastly, the third category of approaches incorporate the use of an alternate way to improve scalability, that of approximate scoring

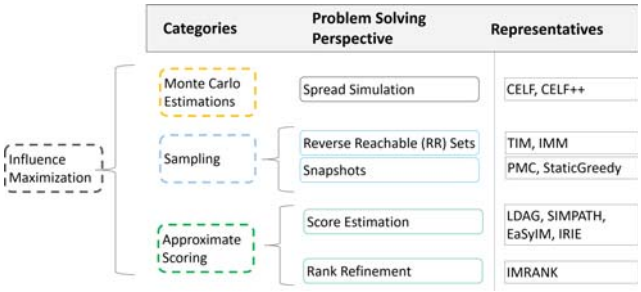


Figure 3: **Categorization of IM approaches.**

mechanisms to estimate the node influence. However, they loose quality guarantees in the process.

Score Estimation methods, such as degree discount heuristics [5], PMIA [4], LDAG [6], SIMPATH [15], IRIE [16] and EaSyIM [10], leverage the idea that influence of a node can be estimated using a function of the number of simple paths starting at that node.

Rank Refinement methods, such as IMRank [7], instead can work by taking as input an initial ordering of nodes generated using any of the score estimation methods discussed above. These methods then iteratively reorder the nodes based on their ranking-based marginal influence spread computed according to current ranking.

Next, we discuss eleven state-of-the-art IM algorithms with a description of how they work under this framework. Notice that, the *spread computation* and the *convergence* phases of the IM Framework are common across the entire spectrum of IM algorithms, hence, we just describe the *seed selection* phase for each algorithm in detail below. Based on specific problem solving perspectives, we consider the following representatives:

- CELF [20] and CELF++ [14] (Spread Simulation, Sec. 4.1)
- TIM$^+$ [27] and IMM [26] (RR Sets, Sec. 4.2)
- StaticGreedy (SG) [8] and PMC [24] (Snapshots, Sec. 4.3)
- LDAG [6], SIMPATH [15], IRIE [16] and EaSyIM [10] (Score Estimation, Sec. 4.4)
- IMRank [7] (Rank Refinement, Sec. 4.5)

Note that we do not consider degree discount heuristics [5] and PMIA [4] as IRIE [16] outperforms them significantly in terms of running time while achieving comparable spread values. We do not consider GREEDY as it is significantly outperformed by CELF [20] and CELF++ [14]. We do not consider RIS [3] as it is outperformed by TIM$^+$ and IMM. We do not consider [18], as the main contribution of this technique is a parallel IM algorithm, thus, it would be unfair to include it in a benchmarking study where all other algorithms possess sequential implementations. Moreover, owing to the highly parallelizable nature of IM and its underlying simulation processes, all the existing algorithms for IM can be parallelized. Lastly, we do not include SKIM [9] as TIM$^+$ has been shown to possess better quality while being similar in running times. Note that not all the techniques work on all the models. The model to technique mapping is present in Table 5 in Appendix.

4.1 Spread Simulation

Seed selection. In the `InfluenceEstimate` step, the spread simulation based algorithms (CELF [20] and CELF++ [14]) perform r explicit MC simulations[2] from each node in order to estimate their influence, and populate the set Inf_V. Next, the node v^* with the highest estimated influence is added to the set of seed nodes S. Eventually, in the `UpdateDataStructures` phase, all the nodes activated till now V_a are marked to be ignored while estimating the influence of nodes in subsequent iterations. The above process is performed iteratively until k seeds are obtained.

Remarks. Both CELF and CELF++ exploit the submodularity property to prune unnecessary node influence estimations while performing seed selection, and thus, provide significant speed-ups over GREEDY. However, the pruning strategies incorporated by both of them are slightly different. CELF uses the simple idea that the marginal gain of a node in the current iteration cannot be better than its marginal gain in the previous iterations, and hence, maintains it while selecting seeds in subsequent iterations. On the other hand, CELF++ extends this to also include the marginal gain with respect to the node that has the previous best marginal gain to achieve better pruning.

[2]As explained previously and as illustrated in Sec. 5, the larger the value of r, the better the influence estimation.

4.2 Reverse Reachable (RR) Sets

Seed selection. TIM$^+$ [27] and IMM [26] perform sampling to efficiently estimate the influence of nodes and perform seed selection. The number of samples θ depends upon the error in $\sigma(S) - \epsilon$, k, and various other internal parameters. Here, the `InfluenceEstimate` step is composed of the following. The first key step is the construction of RR sets corresponding to the sampled nodes. To this end, the direction of all the edges in the graph are reversed. Multiple nodes are sampled from the graph G, and for each sampled node, an instantiation G_i is constructed using the coin-flip technique. More specifically, edges are retained in proportion to their edge-weight probability and deleted otherwise. The set of nodes reachable from each sampled node constitute its RR set. Once the RR sets corresponding to each sampled node is constructed, a greedy max covering algorithm [28] is employed to iteratively select the seed nodes. Note that, the step to find the node with the maximum influence and the `UpdateDataStructures` step are subsumed within the procedure of the max covering algorithm.

Remarks. TIM$^+$ [27] provides several practical optimizations over RIS [3], by improving the parameter estimation procedure, thereby removing the correlation issue and improving the scalability significantly. Moving ahead, IMM [26] employs the concept of martingales to further optimize both the node selection and the parameter estimation phase, thereby resulting in significant improvements in terms of efficiency and scalability.

4.3 Snapshots

Seed selection. As opposed to sampling nodes, SG [8] and PMC [24] generate various instantiations G_i (called snapshots) of graph G apriori and estimate the influence of nodes by averaging over all snapshots. The number of snapshots R control the error in influence estimation; the larger the R the lesser the error. Similar to the RR sets based methods, the coin-flip technique is used to construct snapshots by retaining edges proportional to their edge-weight probability. Once R snapshots have been constructed, the `InfluenceEstimate` step computes a weighted average of nodes reachable from each node across R snapshots to estimates their influence. Next, the node v^* with the highest estimated influence is added to the set of seed nodes S. Eventually, in the `UpdateDataStructures` phase, all the nodes activated till now V_a are marked to be ignored while estimating the influence of nodes in subsequent iterations. The above process is performed iteratively until k seeds are obtained.

Remarks. SG [8] does not provide any theoretical guarantee on the number of snapshots R required to estimate $\sigma(S)$, while PMC [24] does. Moreover, PMC [24] proposes intelligent pruning strategies that help improve its scalability. Lastly, SG does not offer practical scalability over large graphs as illustrated by PMC [24] and in Sec. 5 as well.

4.4 Score Estimation

Seed selection. The score estimation heuristics [4–6, 10, 15, 16] have exploited the exponential decrease of influence probability with hop/path length to efficiently estimate the approximate influence of a node. The key idea that, the influence of a node u on another node v is a function of the number of simple paths between u and v combined with the exponential decrease of influence probabilities with path length summarizes the algorithmic design employed by these techniques. Owing to the above observation, these techniques gain efficiency and scalability, however, at the expense of quality guarantees. In summary, in the `InfluenceEstimate` step each technique either performs node scoring to estimate influence *locally*, namely – SIMPATH [15] and LDAG [6]; or *globally*, namely – IRIE [16] and EaSyIM [10]. The key difference between the local and the global estimation procedures are intuitive and inline with their names. Global estimation procedures perform the `InfluenceEstimate` step in a single attempt on the entire network, however, the local estimation procedures are per node. This also serves as the key difference between the efficiency and scalability of the two sub-classes of scoring methods. After the node with the highest score is added to the set of seed nodes S, the `UpdateDataStructures` in this group of algorithms discounts the contribution of the selected seed node in order to get an estimate of expected marginal gain of subsequent candidate nodes.

Remarks. IRIE [16] and EaSyIM [10] owing to their global estimation procedure outperform the other local estimation based heuristics like degree discount [5] and PMIA [4], which thus, are ignored in this study.

4.5 Rank Refinement

Seed selection. The seed selection phase in IMRank [7] iterates multiple times in order to get a converged ranking for a particular set of external parameters. In the first scoring round, the initial influence (`InfluenceEstimate`) is estimated using simple ranking strategies like degree discount heuristics [5] or PageRank [25]. The `UpdateDataStructures` takes the ranking from the previous scoring round as an input and estimates the marginal influence spread of all nodes with respect to the previous ranking. The marginal influence of nodes is calculated in last-to-first order of ranking. The basic intuition here is that the last node (the node possessing the least influence) cannot influence anyone and thus, can help recursively estimate the influence of its predecessors in the ordering. In subsequent steps, this input from `UpdateDataStructures` is incorporated to update the ranking of the nodes and the seed selection step stops as soon as the ranking converges.

5. BENCHMARKING EVALUATION

The four most desirable properties of an IM algorithm are quality of spread, computational efficiency, memory footprint, and robustness to datasets, diffusion models and parameters. In this section, we conduct in-depth analysis of the 11 selected IM techniques (as stated in Fig. 3) to evaluate them across each of these properties. All experiments are performed using codes written in C++ on an Intel(R) Xeon(R) E5-2698 64-core machine with 2.3 GHz CPU and 256 GB RAM running Ubuntu[3] 14.04. We present results on real (large) graphs, taken from the arXiv [1] and SNAP [2] repositories, as described in Table 1. In addition to these, a snapshot of the *Twitter* network crawled from Twitter.com in July 2009 is also used. We consider a mix of directed and undirected graphs. However, to ensure uniformity, the undirected graphs are made directed by considering, for each edge, the arcs in both directions.

[3]IRIE [16] was compiled on a Microsoft Windows 7 machine possessing the same configuration.

Dataset	n	m	Type	Avg. Degree	90-%ile Diameter
Nethept	15K	31K	Undirected	2.06	8.8
HepPh	12K	118K	Undirected	9.83	5.8
DBLP	317K	1.05M	Undirected	3.31	8
YouTube	1.13M	2.99M	Undirected	2.65	6.5
LiveJournal	4.85M	69M	Directed	14.23	6.5
Orkut	3.07M	117.1M	Undirected	38.14	4.8
Twitter	41.6M	1.5B	Directed	36.06	5.1
Friendster	65.6M	1.8B	Undirected	27.69	5.8

Table 1: **Summary of the datasets used in our experiments.**

5.1 Experimental Setup

Information-Diffusion Models: We incorporate the use of three diffusion models, namely – IC-constant denoted using IC, IC-WC denoted as WC and LT-uniform denoted as LT (Sec. 2.1). The IC model is used with a constant probability $p_{(u,v)} = 0.1$ assigned to all the edges of the network. As opposed to the IC and WC models, the LT model requires an additional parameter. Apart from the edge weights $w_{(u,v)} = \frac{1}{|\mathbf{In}(v)|}$, all nodes contain a node activation threshold $\theta_v = rand(0,1)$, $\forall v \in V$. Note that the $rand(0,1)$ function generates numbers randomly and uniformly between 0 and 1. The chosen models follow the conventional practice in the literature and are by far the most popularly used setups [6, 10, 14, 17, 20, 26, 27].

Computing expected spread: As outlined in Fig. 2, we first let each algorithm output their k most influential seeds. Once the seeds are identified, we perform 10K MC simulations to compute the expected spread. In other words, instead of merging the spread computation and seed selection in a single step, we decouple the two steps to bring all the techniques at a uniform comparison standpoint. The number 10K is selected only after extensive experiments ensuring that the spread reported is stable and has converged. The details are in Fig. 12 of appendix.

Parameters: Every technique has two kinds of parameters: internal and external. The external parameters are exposed through the API and can be tuned to optimize performance. Table 2 lists the external parameters of all techniques being benchmarked. On the other hand, the internal parameters are set based on the authors' judgements (ex: the length of the influence path in EaSyIM [10] or the size of the DAG in LDAG [6]). In our experiments, we do not tune the internal parameters and set them to their default values as recommended by the authors. However, the external parameters are extensively analyzed to understand when they perform best. We next discuss this analysis.

5.1.1 External Parameters

The goal in this set of experiments is to identify the optimum parameter values for each of the techniques being benchmarked. Note that LDAG [6], IRIE [16] and SIMPATH [15] do not have any external parameters, and thus this analysis does not cover them.

Generally, a parameter value is optimum if it provides the best possible spread for that technique, while being computationally efficient and robust across datasets and k. As in most algorithms, a

Algorithm	Parameter	IC	WC	LT
CELF [20]	#MC Simulations	10000	10000	10000
CELF++ [14]	#MC Simulations	7500	7500	10000
EaSyIM [10]	#MC Simulations	50	50	25
IMRank, $l = 1$ [7]	#Scoring Rounds	10	10	NA
IMRank, $l = 2$ [7]	#Scoring Rounds	10	10	NA
PMC [24]	#Snapshots	200	250	NA
Static Greedy [8]	#Snapshots	250	250	NA
TIM$^+$ [27]	ϵ	0.05	0.15	0.35
IMM [26]	ϵ	0.05	0.1	0.1

Table 2: **Optimal parameter values for the various algorithms.**

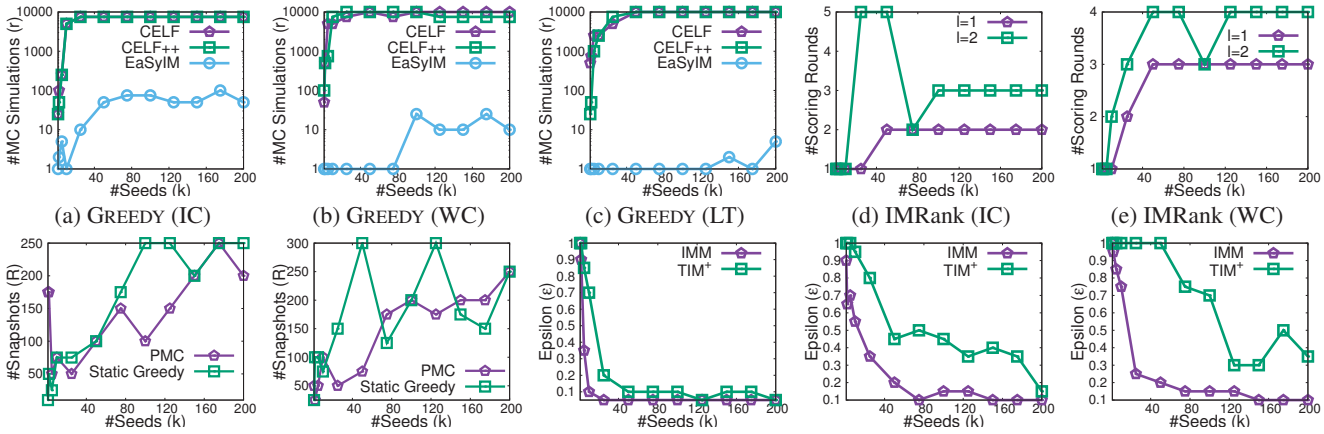


Figure 4: **Identifying the optimal parameter values for the parameters outlined in Table 2.**

higher spread often comes at the cost of higher computational cost. Hence, we formalize the following generic procedure to identify the optimal parameter values. Let X be the parameter that dictates the quality of spread. The parameter for each of the techniques is listed in Table 2. Now, let X^* be the value at which the highest spread is obtained within a reasonable time limit[4]. We identify X^* by computing the spread across the entire spectrum of X. The plots corresponding to this experiment are available in Fig. 14 in Appendix. We denote the mean spread at X^* across the 10K MC simulations as $\mu*$ and the standard deviation as sd*. X^* is also likely to be the value that corresponds to highest computational cost. For example, as the number of MC simulations is increased, both spread and the running time goes up for greedy techniques. On the other hand in TIM$^+$ and IMM, lowering ϵ increases the number of nodes sampled, and therefore both the spread and the running time goes up. Since the optimal value should also optimize the running time without compromising much on the spread, we set X to the value where the running time is minimized and the spread is within sd* from μ^* across all datasets. In other words, we choose the value that optimizes the running time while being at most one standard deviation away from the best possible spread.

The result of this analysis is presented in Fig. 4. The experiment is performed on four datasets, namely Nethept, HepPh, DBLP and YouTube. Fig. 4 shows the results for only HepPh since this is the largest dataset on which all techniques complete across all diffusion models and k. The results on the other datasets are provided in the appendix (Figs. 15 and 16). In each plot in Fig. 4, the x-axis varies the number of seeds k and the best parameter value for a particular k is shown on the y-axis. Since, our goal is to select the parameter value that holds across all ks, we select the optimal parameter value as the value where the spread is within one standard deviation from μ^* on the largest value of k[5] ($k = 200$ for our experiments). Consequently, the chosen parameter value for an algorithm is the one that is optimal across all datasets. Such a value is guaranteed to exist as long as the spread monotonically increases or decreases with the parameter being optimized. Following this analysis, the optimal values for all algorithms across each of the three models is computed and listed in Table 2. Beyond identifying the optimal values for each technique, we highlight two key observations from this analysis.

• **TIM$^+$ [27] Vs. IMM [26]:** IMM claims to be 100 times faster than TIM$^+$. While this is true if they are both executed at same ϵ, we notice that TIM$^+$'s optimal value of ϵ is higher than IMM for both WC and LT, which translates to faster running times. While IMM is still generally faster than TIM$^+$, the gap in their running times is not as drastic as two orders of magnitude. We provide more concrete data on this in Sec. 5.3.1.

• **IMRank [7]:** In IMRank, the general idea is to start with an initial ranking of top-k seed nodes, and then improve this top-k set over each scoring round. The stopping criteria present in the implementation of IMRank [7] checks for the state where the set of top-k seed nodes remain the same over two consecutive iterations or the number of iterations crosses a certain threshold. It is a little different from the theoretical convergence condition of the IMRank framework, which is the ranking of top-k nodes remains the same over two consecutive iterations. Such a stopping criteria is due to the observation that the influence spread of the top-k nodes always converges more quickly than the convergence of the ranking of top-k nodes. However, we find that the current stopping criteria of the IMRank framework is defective since it often exits early even before the convergence of the influence spread of the top-k nodes, especially when k is large[6]. Based on the suggestion by the authors of [7], in this paper we fixed this problem by changing the stopping criteria to always iterate over 10 scoring rounds. However, owing to the strange behavior of spread with scoring rounds in some cases, and specifically as portrayed for HepPh in Fig. 5, how to de-

[4]We stress on the point of "reasonable time limit" since in many cases although the quality may improve, the parameter values may not be practically tunable. An example of such a scenario is having more than 10K MC simulations or $\epsilon < 0.05$.

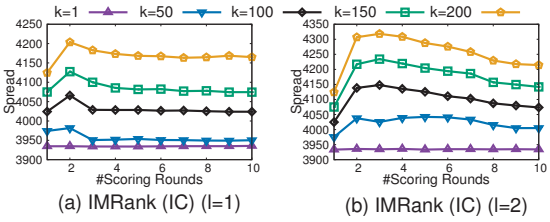


Figure 5: **Variance of spread with scoring rounds in IMRANK on the HepPh dataset.**

[5]This follows from [27], which shows that the parameter value required to obtain a desired level of quality becomes stricter with increase in k.

[6]Please refer to the insights about this observation in M7 (Sec. 6) and Appendix B.

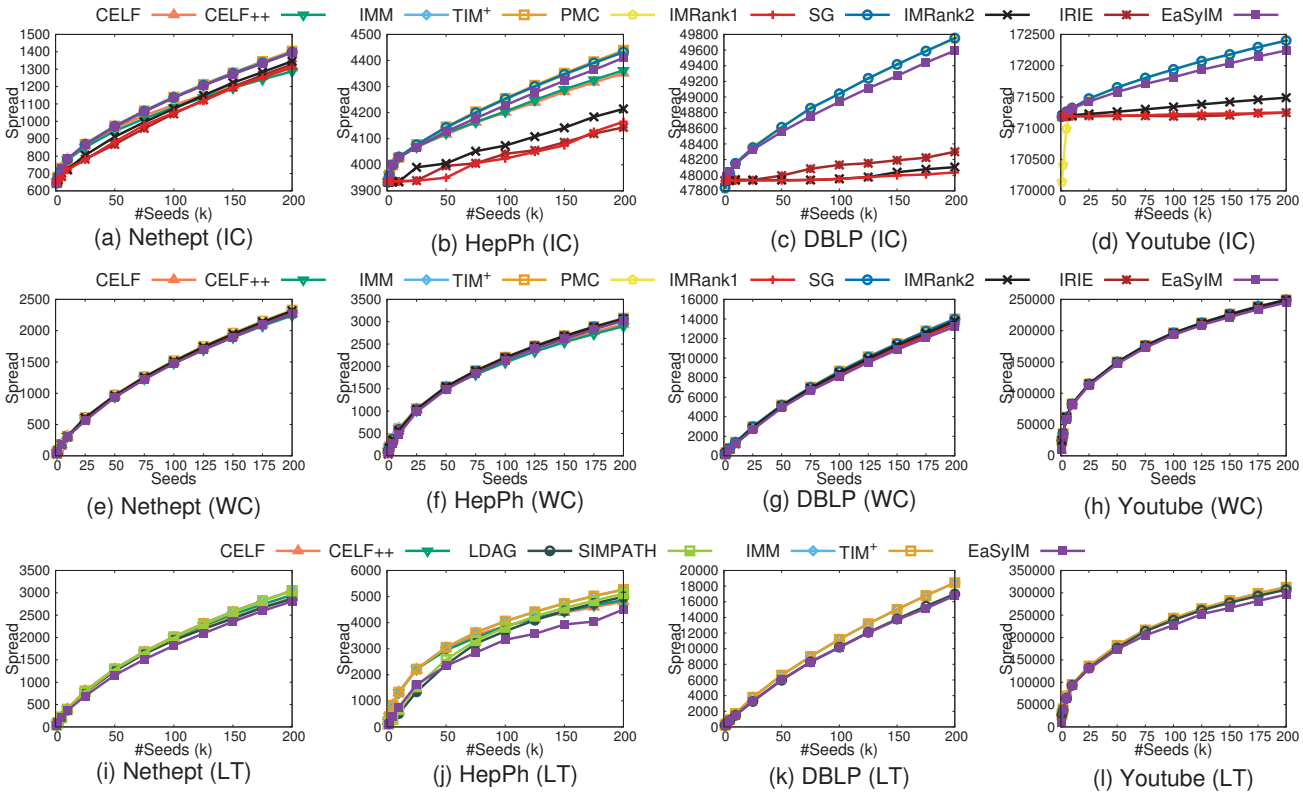


Figure 6: **Growth of spread against the number of seeds. StaticGreedy is denoted as SG.**

sign a better stopping criteria that is compatible with the IMRank framework is still an open question. Moreover, as mentioned in the parameter selection process described earlier in this section, owing to this non-monotonicity of spread with scoring rounds, it is also difficult to identify its optimal value across k and datasets.

5.2 Quality

Equipped with the optimal parameter values for each algorithm, we next benchmark their quality. Quality, in our problem, equates to the expected spread from the k influential seeds identified by an algorithm. Fig. 6 presents the growth in spread against the number of seeds across the first four datasets in Table 1. We analyze the performance in the four larger datasets in Sec. 5.5. Generally, the spread grows with k although few minor fluctuations are visible. The key observations from this experiment are as follows.

• **Scalability of TIM$^+$ and IMM:** Both TIM$^+$ and IMM do not scale beyond HepPh in terms of memory-consumption under the IC model. We further elaborate on the reasons behind this non-scalability in Sec. 5.3.1, where we benchmark the scalability of all techniques.

• **Performance of IRIE [16] and IMRank [7]:** The performance for both these techniques under the IC model is significantly weaker when compared to their performance under WC.

• **Scalability of CELF [20] and CELF++ [14]:** CELF and CELF++ do not scale beyond HepPh in terms of running-time under both IC and WC and thus they are not included in the results for DBLP and YouTube.

• **CELF and CELF++ work better at low values of k:** CELF and CELF++ provide close to highest spread till $k = 25$. However, beyond that, they fail to remain competitive. This is particularly evident in IC, which is shown in Figs. 6a and 6b. We explain the reasons behind this behavior in point $M2$ in the Myths section (Sec. 6).

• **PMC [24] and SG [8]:** PMC and SG establish themselves as the only two techniques that consistently provide high spread and scale for both IC and WC.

5.3 Scalability

While quality forms one important aspect of IM techniques, scalability is an equally important aspect that determines the utility in practical scenarios. For an algorithm to be termed scalable, it must scale well with both running time and memory consumption. We first analyze the running times.

5.3.1 Running Time

Fig. 7 demonstrates the growth of running times across the first four datasets in Table 1 against the number of seeds. As expected, most techniques show growth in running time with increase in number of seeds. The only exceptions are IMM [26] and TIM$^+$ in LT. This result however is not an aberration and the reason is discussed in detail in the respective papers. The key observations derived from the running time analysis are as follows.

• **CELF [20] Vs CELF++ [14]:** CELF++ claims to be at least 35% faster than CELF and is considered the state-of-the-art MC estimation based technique. Our analysis reveals a different behavior where both CELF and CELF++ show almost identical running times. To understand the reason behind this deviation from the claimed results, we analyze the inner workings of both algorithms and pinpoint the reason in point $M1$ of the Myths section (Sec. 6).

• **Scalability of TIM$^+$ [27] and IMM [26] under IC and WC:** Although both these techniques scale well under WC, they find it difficult under IC. In our experiments, both TIM$^+$ and IMM crash in IC due to running out of memory for all datasets except Nethept and HepPH. TIM$^+$ and IMM employ a reverse-reachability set based mechanism to compute the seeds. The size of these sets is proportional to the edge weights and therein lies the difference in the per-

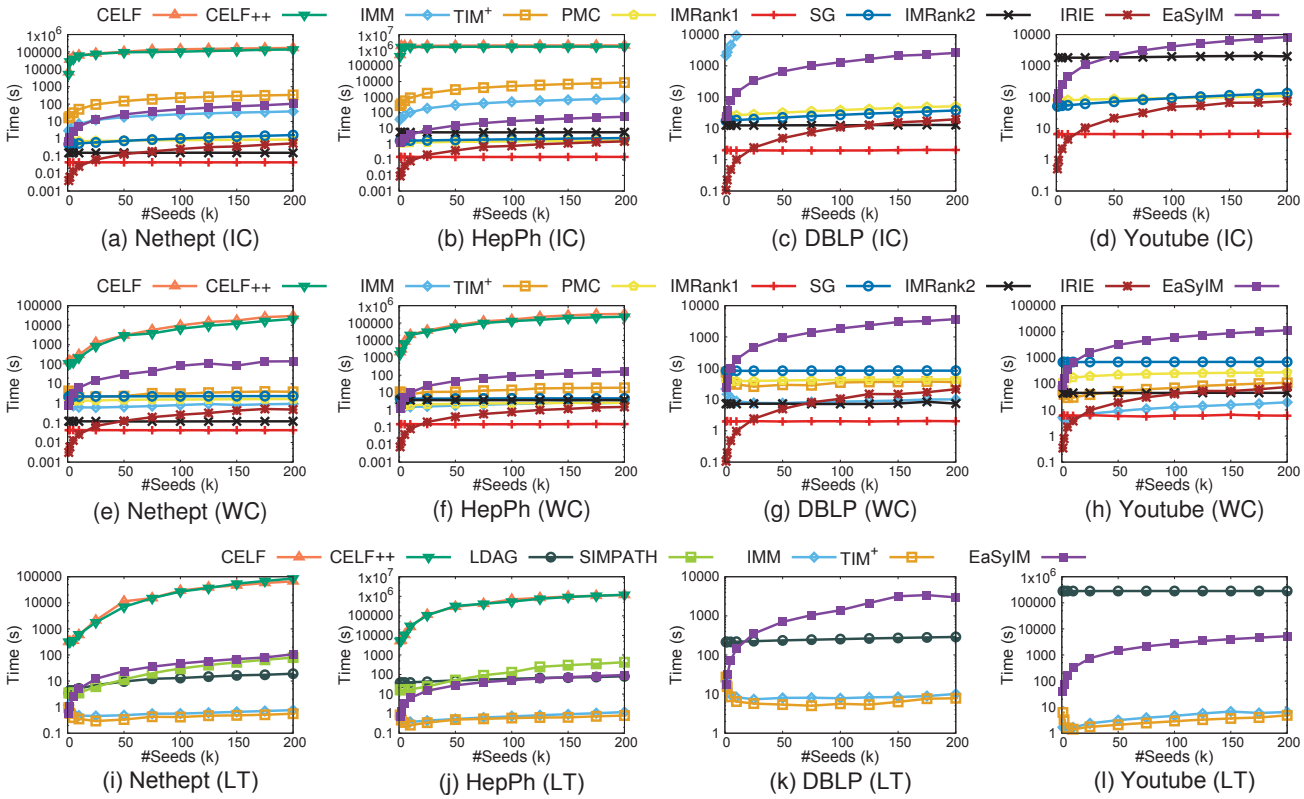


Figure 7: **Growth of running time against the number of seeds. StaticGreedy is denoted as SG.**

Dataset	IC						WC									LT					
	Spread (%)		Time (sec)		Memory (MB)		Spread (%)			Time (sec)			Memory (MB)			Spread (%)		Time (sec)		Memory (MB)	
	PMC	EaSyIM	PMC	EaSyIM	PMC	EaSyIM	PMC	IMM	EaSyIM	PMC	IMM	EaSyIM	PMC	IMM	EaSyIM	TIM+	EaSyIM	TIM+	EaSyIM	TIM+	EaSyIM
LiveJournal	**34.91%**	34.90%	**1039**	33505	31250	**1013**	3.10%	**3.12%**	2.97%	1460	**126**	107147	38081	3655	**1051**	**5.31%**	5.09%	**35**	54083	2580	**1050**
Orkut	87.63%	87.63%	**1327**	9464	18122	**2385**	8.02%	**8.03%**	DNF	4344	**395**	DNF	29727	**6674**	DNF	**29.26%**	DNF	**46**	DNF	**4891**	DNF
Twitter	**44.33%**	DNF	**44886**	DNF	**292845**	DNF	Crashed	**30.98%**	DNF	Crashed	**597**	DNF	Crashed	**27702**	DNF	**57.32%**	DNF	**87**	DNF	**27446**	DNF
Friendster	Crashed	DNF	Crashed	DNF	Crashed	DNF	Crashed	**12.36%**	DNF	Crashed	**9982**	DNF	Crashed	**107939**	DNF	**28.75%**	DNF	**2454**	DNF	**89174**	DNF

Table 3: **Performance of the scalable techniques on large datasets at** 200 **seeds. Spread is reported in terms percentage of nodes in the original network. DNF indicates that the algorithm did not terminate even after** 40 **hours. Crashed indicates that the algorithm crashed due to running out of memory. Numbers in bold indicate it to be the best performance in that category.**

formance of these two techniques under IC and WC. More specifically, under the WC model, the edge-weight propagation probability is defined as $p(u,v) = W(u,v) = 1/|\mathbf{In}(v)|$. Since typically the in-degree of a node is high in social networks, most of the edges are not retained in the sampled graph instantiations. Hence, the size of the reverse-reachability set is small. On the other hand, in the generic IC model, if the edge probabilities are higher, which is 0.1 for our experiments, then the size of the reverse-reachability set is significantly larger. Furthermore, due to more edges being retained, the computation cost of constructing the reverse-reachability set also increases significantly. Consequently, both TIM$^+$ and IMM fail to scale well under IC although they are highly scalable when the edge-weight probabilities are small such as in WC.

• **TIM$^+$ [27] is faster than IMM [26] under LT:** As we highlighted earlier, the optimal ϵ value for TIM$^+$ is higher than IMM, and consequently the number of samples required to compute the seeds is less in TIM$^+$. As a result, TIM$^+$ emerges as a more efficient technique than IMM in LT. This is a deviation from the general consensus in the IM literature, since so far, IMM and TIM$^+$ have only been benchmarked at same ϵ values. Our study reveals that such a comparison actually penalizes TIM$^+$ for the inability of IMM to operate at a higher ϵ value.

• **SIMPATH [15] Vs. LDAG [6]** SIMPATH was proposed as an improvement over LDAG and is generally thought to be more scalable than LDAG. However, in Fig. 7c, we observe contradictory results where LDAG provides faster performance. Furthermore, SIMPATH fails to finish even after 2400 hours on DBLP and YouTube. On investigating further, we discover that SIMPATH provides faster performance than LDAG only on the "parallel edges" LT model, which is used in the SIMPATH paper [15]. In our experiments, we use the "uniform" LT model (refer to Sec. 2.1.2 for the definitions of the two LT models). For the sake of completeness, we also compare the performance of SIMPATH with LDAG under the LT-"parallel edges" model and even in this model LDAG achieves better performance. We discuss the details further in point $M5$ of the Myths section (Sec. 6).

5.4 Memory Consumption

Finally, we evaluate the memory consumption of the various techniques being benchmarked. Fig. 8 presents the results. As expected, the memory consumption goes up with increase in the number of seeds. It is also evident from the plots that sampling based strategies possess a higher memory footprint. The key insights from this experiment are as follows.

• **EaSyIM [10] is most memory-efficient:** In contrast to other techniques that store structural information for each node such as reverse-reachability sets [26, 27], directed acyclic graphs [6], etc., EaSyIM only stores a number per node. Consequently, it is the most memory efficient technique for IM.

• **TIM$^+$ [27] and IMM [26]:** The scalability of TIM$^+$ and

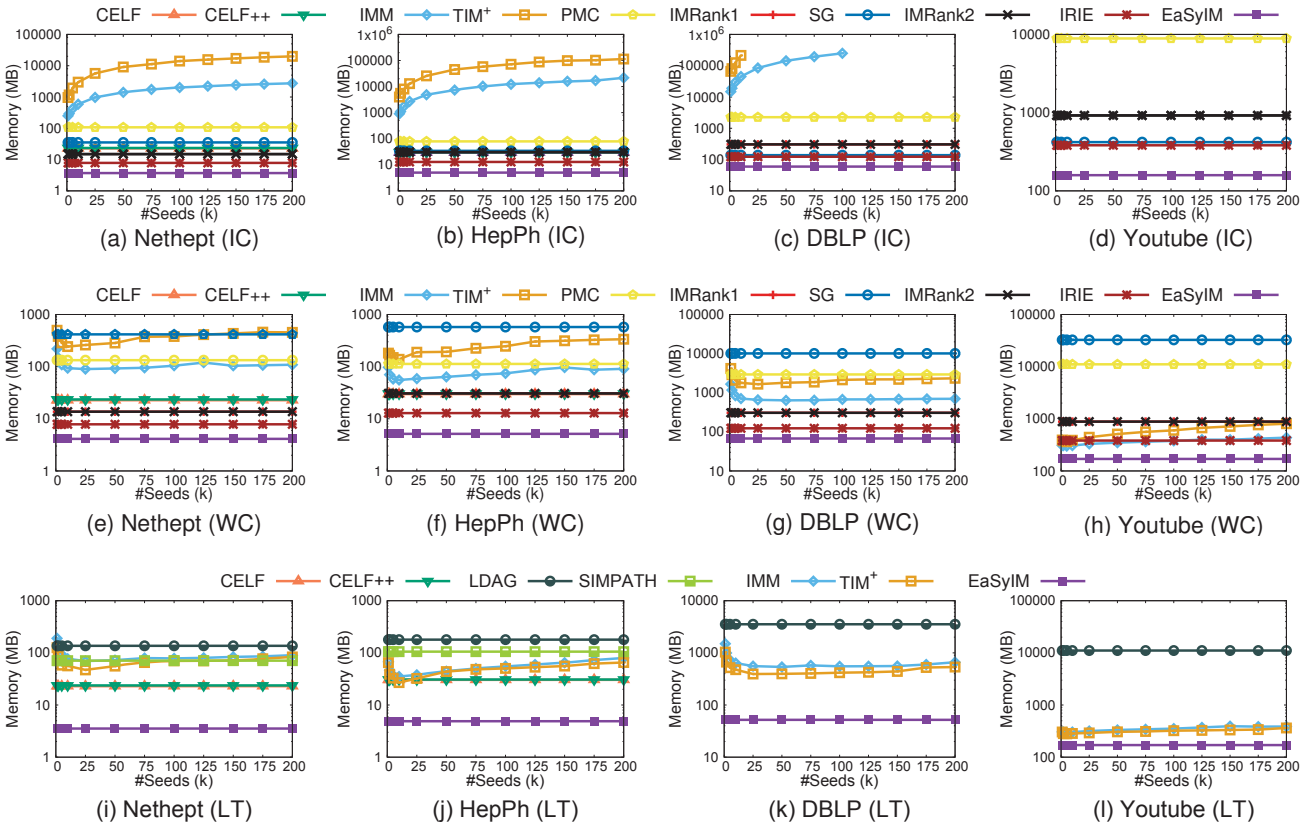


Figure 8: **Growth of main memory footprint against the number of seeds. StaticGreedy is denoted as SG.**

IMM is significantly weaker under IC when compared to its performance under WC with respect to memory consumption. The reasons are discussed earlier in Sec. 5.3.1.

5.5 Performance on large datasets

From our experiments, it is clear that the four most promising algorithms are PMC, IMM, TIM$^+$ and EaSyIM. PMC and SG are the two fastest and best performing algorithms under the IC model. However, owing to a gap in the implementation of SG [8], it crashed on the large datasets due to out of memory errors, and thus, we could not run SG on the 4 large datasets[7]. IMM and TIM$^+$ are the two fastest and best performing techniques in LT and WC. EaSyIM provides competitive performance with the lowest memory footprint. In Table 3, we summarize their performances on the four largest datasets. In IC, we only include PMC [24] and EaSyIM [10] since IMM and TIM$^+$ crash due to running out of memory on all four datasets. In WC, we omit the results of TIM$^+$ since its spread is identical to IMM with IMM being more than five times faster. EaSyIM is included to highlight its low memory usage. In LT, TIM$^+$ is marginally faster than IMM while providing almost identical spreads. Hence, IMM is omitted for LT. Consistent with previous results, EaSyIM demonstrates a low memory footprint in LiveJournal. However, it fails to finish in all of the other datasets.

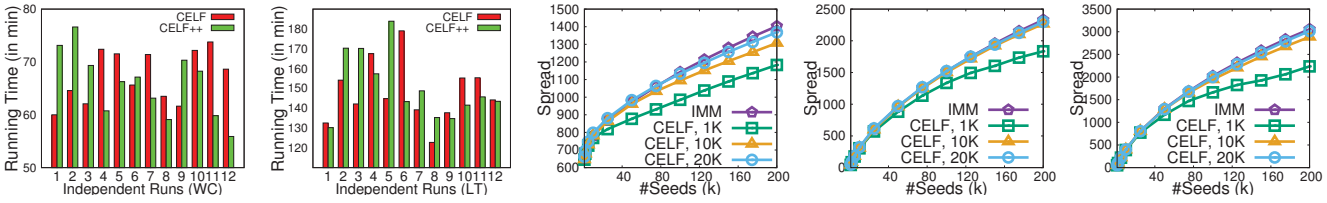
6. MYTHS

In this section, we outline several results from our benchmarking study that either contradict published results or have been assumed to be correct without thorough empirical evaluation.

[7]We are currently working with the authors to resolve the errors in the implementation, and once resolved the results of SG on large datasets will be made available through our website

M1. CELF++ is the fastest IM technique in the MC estimation paradigm: CELF++ [14] claims to be 35% faster than CELF. However, our experiments reveal that on average there is negligible difference in the running times of the two techniques (Fig. 7). Both CELF and CELF++ exploit submodularity of spread to optimize the iterative greedy selection of seed nodes. CELF utilizes the property that the marginal gain of a node can only decrease over iterations. Thus, if the marginal gain of a node n in a previous iteration is less than the marginal gain of some node in the current iteration, then n need not be updated. CELF++ adopts a more aggressive approach by pre-empting the node that might provide the highest marginal gain. Based on this pre-emption, it performs some extra work with the hope that it will save cost in the future iterations. Thus, the overall efficiency of CELF++ is dependent on how well this pre-emption works. Our experiments show that on average this pre-emption does not provide any significant benefit on the overall running time above CELF. In Figs. 9a and 9b, we run 12 independent runs of CELF and CELF++ with $k = 50$ on the Nethept dataset under both LT and WC models and as can be seen, no technique can be termed better than the other. On average, CELF consumes 147.88 minutes (sd=14.74 minutes) compared to 150.39 minutes (sd=15.91 minutes) by CELF++ on the LT model, while consuming 67.28 minutes (sd=4.70 minutes) when compared to 65.82 minutes (sd=5.94 minutes) by CELF++ on the WC model. We also include experiments comparing the average node-lookup values of CELF with CELF++, since it is a more robust evaluation metric. Please see Appendix C for the results on node-lookup and the corresponding discussion.

M2. CELF (or CELF++) is the gold standard for quality: Theoretically, the GREEDY algorithm proposed by Kempe et al. [17] and its efficient versions CELF and CELF++ provide the best spread (along with TIM$^+$ and IMM). Consequently, various IM



(a) Nethept (WC) (b) Nethept (LT) (c) Nethept (IC) (d) Nethept (WC) (e) Nethept (LT)

Figure 9: **(a-b) Running time comparison of CELF with CELF++ on the Nethept dataset over 12 independent executions. (c-e) Comparison of the spreads obtained by CELF at different numbers of MC simulations.**

techniques [4–6, 10, 15, 16, 29] have claimed state-of-the-art performance by showing comparable or better performance than CELF and CELF++. An important factor that is frequently overlooked is that the quality of CELF and CELF++ depend on the number of MC simulations performed to compute the node providing highest marginal gain in each iteration. It has been proven that the number of MC simulations must increase with increase in the number of seeds to preserve the theoretical guarantees on quality [27]. Figs. 9c-9e showcase this effect empirically. Notice that at 200 seeds, CELF achieves the spread of IMM only after 20,000 MC simulations. At lower number of seeds though, 1,000 simulations are enough. It must be mentioned though that anything beyond 10K simulations make CELF non-scalable even on small networks. For example, on Nethept, which is the smallest dataset in our study, CELF consumes 80 hours to finish at 20,000 simulations.

M3. IMM [26] is always faster than TIM$^+$ [27] The empirical results in this study indicate TIM$^+$ and IMM to have similar running times. In fact, TIM$^+$ is marginally faster than IMM in LT model. This phenomenon is a direct consequence of TIM$^+$ being able to provide good spread at a higher ϵ than IMM, and this observation has so far remained hidden in IM research.

M4. Spread estimated by TIM$^+$ [27] and IMM [26] decreases with increase in ϵ: TIM$^+$ and IMM compute their seeds based on *estimated* spreads. These estimated spreads are often inflated due to the procedure followed to compute them (read our note regarding their implementation in Appendix A). More specifically, instead of directly computing the spread through MC simulations, TIM$^+$ and IMM extrapolate the spread of the seed nodes on the sampled sub-network, over the entire network to compute the spread. Mathematically, let R be the nodes reachable from at least one of the seed nodes on the sampled network, M be the number of sampled nodes, and N be the number of nodes on the entire network. The spread is approximated as $\frac{R}{M} \times N$.

The consequence of computing the spread through extrapolation is shown in Figs. 10c-10e. As can be seen, the extrapolated spread is significantly higher from the actual expected spread computed through MC simulations. More critically, the extrapolated spread increases with increase in ϵ, which is counter-intuitive. On the other hand, the spread computed by MC simulations follows the theoretical expectations. Two important questions arise at this juncture: *Why is the extrapolated spread always higher than the actual expected spread? Furthermore, why does the extrapolated spread increase with ϵ when the opposite behavior is expected?*

ϵ controls the number of sampled nodes, and the seed selection is performed on the sub-network induced by this sampled nodes. A perfect sample is the one where $\sigma_{sub}(\{n\}) = \sigma(\{n\}) \times \frac{M}{N}$ for all sampled nodes n, where $\sigma_{sub}(\{n\})$, $\sigma(\{n\})$ denote the spreads of n on the sub-network and complete network respectively, and M and N denote the number of sampled nodes and total nodes in the entire network respectively. In reality, chances of generating such a sample is negligible. Thus, either we over-estimate the spread of a node or under-estimate. Since, TIM$^+$ and IMM select the k nodes providing the highest combined spread in the sampled sub-network, the chances of over-estimating their spread is much higher than under-estimating. Thus, we see the behavior depicted in Figs. 10c-10e. Furthermore, the smaller the sample size, the more error prone is the extrapolated result. Consequently, for larger ϵ, the extrapolated spread increases, which in reality is due to the higher amount of error incurred from a small sample size.

As visible in Figs. 10c-10e, the error in IMM is much higher than the error in TIM$^+$. In fact, IMM provides comparable spread to TIM$^+$ only at $\epsilon \leq 0.1$. This indicates that the Martingales based sampling approach is *less stable* than the sampling algorithm in TIM$^+$. An obvious question arises at this juncture: *What is the impact of a high ϵ on the quality in terms of expected spread?* The performance of IMM has been shown to be steady with ϵ in large networks, such as Twitter (Fig. 3d (IC model) and 6d (LT model) in [26]). In our experiments however, Figs. 10c- 10e indicate that this behavior is not consistent across all datasets. Thus, operating at a high ϵ may lead to inferior results in some datasets.

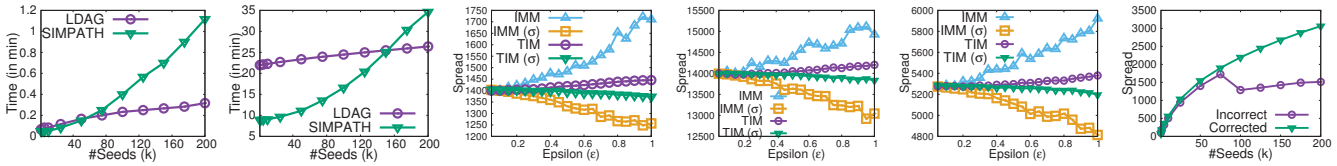
M5. SIMPATH [15] is faster than LDAG [6]: SIMPATH was proposed as an improvement over LDAG from the scalability perspective. However, as we identified in Sec. 5.3.1, this result does not hold in the "uniform" LT model, where LDAG is significantly faster. On YouTube and DBLP, SIMPATH did not finish even after running for 2400 hours (Fig. 7k-l).

In the SIMPATH paper, the algorithms are evaluated only on the LT-"parallel edges" diffusion model and not on LT-"uniform". We therefore evaluate these two techniques under the "parallel edges" LT model as well. Furthermore, for the sake of repeatability, we obtain the same DBLP dataset from the authors of SIMPATH (denoted as *DBLP (Large)*). This DBLP dataset is different and larger than the DBLP dataset listed in Table 1. Figs. 10a-10b present the results on two datasets under LT-"parallel edges". We observe that SIMPATH is faster only when the number of seeds is small. In the SIMPATH paper, these two techniques are not evaluated beyond 100 seeds and thus this observation remained hidden. Table 4 lists the time taken by the two techniques for 200 seeds. Overall, these results indicate that LDAG not only scales better than SIMPATH but is also more robust to the underlying diffusion model.

M6. WC is equivalent to IC: Several techniques have misused the term IC. Specifically, they have shown good performance only under the WC model, which is one specific instance of IC, and

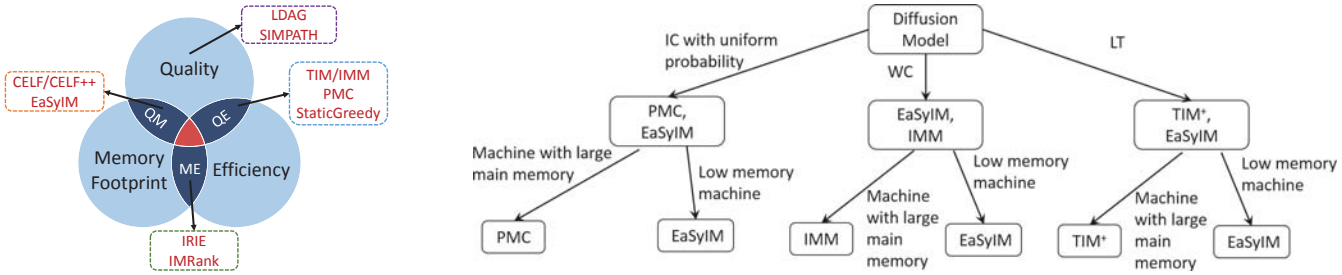
Algorithm	Nethept	Nethept-P	HepPH	DBLP	DBLP (large)-P
LDAG	0.37 min	0.32 min	1.5 min	5.5 min	26.4 min
SIMPATH	1.5 min	1.1 min	8.1 min	DNF	34.6 min

Table 4: **The times taken by LDAG and SIMPATH for 200 seeds on four datasets under the LT model. For the columns titled Nethept-P and DBLP (large)-P, we use the LT-parallel edges model. For the remaining datasets we use LT-uniform. DNF indicates that the algorithm did not finish even after 2400 hours.**



(a) Nethept (b) DBLP (Large) (c) Nethept (IC) (d) DBLP (WC) (e) HepPh (LT) (f) HepPh (WC)

Figure 10: **(a-b) Running time comparison of SIMPATH with LDAG when edge-weights are assigned using the LT-parallel-edges model. (c-e) Comparison of spreads reported by IMM and TIM$^+$ (denoted as IMM and TIM) with the spread obtained through the classical way of MC simulations (denoted as IMM(σ) and TIM$^+$(σ)) against ϵ. (f) Comparison of the spread obtained in IMRank with the original convergence criteria proposed by the authors (indicated as Incorrect) with that of corrected criteria in the HepPH dataset. Similar behavior is found in other datasets as well.**



(a) (b)

Figure 11: **(a) Summarizing the spectrum of Influence Maximization (IM) techniques based on their strengths. (b) The decision tree for choosing the most appropriate IM algorithm.**

claimed to be the state-of-the-art for IC. In reality, they all fare poorly on the generic IC model, either due to non-scalability [26, 27] or producing poor spread [7, 16].

M7. Convergence of IMRank [7]: As we have already discussed, we discovered the convergence criteria of IMRank is problematic. First, if the convergence criteria proposed in IMRank is followed, then we observe that the spread decreases with increase in the number of seeds. Fig. 10f presents the results. This behavior stems from the fact that the algorithm would often not proceed beyond scoring round 1 (read our note regarding the implementation of IMRank in Appendix B). Based on the authors' suggestion, we modified the algorithm to run till 10 scoring rounds regardless of the number of seeds or diffusion model. However, even after this modification, as we showed earlier in Fig. 5, the spread does not behave monotonically with the number of scoring rounds. Due to this randomness, the performance of IMRank is unstable, particularly when the number of seeds is larger than 50.

7. CONCLUDING INSIGHTS

To summarize, a good algorithm for IM stands on three pillars: quality of spread, running time efficiency, and main memory footprint. In addition, it is desirable for the technique to be robust across diffusion models, datasets, and parameters. We benchmark the eleven most promising techniques across all of these features. Fig. 11a summarizes the results. Notice that there is no technique that stands strong on all three pillars. In other words, there is no single state-of-the-art technique for IM.

Several techniques exist that stand on two pillars. Among these, techniques that lie in the "ME" category (memory + efficiency) do not provide a good solution since ensuring quality is of utmost importance. Consequently, in practical scenarios, the choice of the best IM technique is between those that lie in the "QM" and "QE" categories. Note that PMC is always either comparable or better than StaticGreedy, hence the latter is omitted in the following analysis. Towards that goal, Fig. 11b presents the decision tree for choosing the best IM technique given the task and resources in hand. In terms of quality, TIM$^+$, IMM and PMC provide the best spread. These three techniques also provide the fastest performance under LT, WC and IC with uniform weights respectively. Thus, if main memory budget is not a constraint, the choice is between these three techniques. When main memory is scarce, EaSyIM, CELF, CELF++ and IRIE provide alternative solutions. Among these, EaSyIM easily out-performs the other three techniques in memory footprint, while also generating reasonable quality and efficiency. Overall, the choice is between four techniques: IMM, TIM$^+$, EaSyIM, and PMC. Here, we note that the area of IM is an evolving field and a highly promising technique has been published in SIGMOD 2016 [23]. Unfortunately, we could not include the technique in our study due to how recently it is published. Nonetheless, our benchmarking study will also evolve with the inclusion of more recent techniques, as and when they get published, and results will be made available at https://sigdata.github.io/infmax-benchmark. We hope the insights obtained from this study will provide the directionality and clarity required for a more streamlined advancement in IM research.

Acknowledgements

We would like to thank the authors of all the state-of-the-art techniques for sharing their code with us. Our special thanks to Suqi Cheng (IMRank [7], StaticGreedy [8]) and Wei Lu (CELF++ [14], SIMPATH [15]) for their help in setting up the codes, and their enthusiasm to engage in constant discussions, which facilitated us in getting a clear insight of their work.

8. REFERENCES

[1] arXiv. http://www.arxiv.org.

[2] SNAP Datasets. https://snap.stanford.edu/data/.

[3] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *SODA*, pages 946–957, 2014.

[4] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *KDD*, pages 1029–1038, 2010.

[5] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *KDD*, pages 199–208, 2009.

[6] W. Chen, Y. Yuan, and L. Zhang. Scalable influence maximization in social networks under the linear threshold model. In *ICDM*, pages 88–97, 2010.

[7] S. Cheng, H. Shen, J. Huang, W. Chen, and X. Cheng. IMRank: influence maximization via finding self-consistent ranking. In *SIGIR*, pages 475–484, 2014.

[8] S. Cheng, H. Shen, J. Huang, G. Zhang, and X. Cheng. Staticgreedy: solving the scalability-accuracy dilemma in influence maximization. In *CIKM*, pages 509–518, 2013.

[9] E. Cohen, D. Delling, T. Pajor, and R. F. Werneck. Sketch-based influence maximization and computation: Scaling up with guarantees. In *CIKM*, pages 629–638, 2014.

[10] S. Galhotra, A. Arora, and S. Roy. Holistic influence maximization: Combining scalability and efficiency with opinion-aware models. In *SIGMOD*, 2016.

[11] S. Galhotra, A. Arora, S. Virinchi, and S. Roy. ASIM: A scalable algorithm for influence maximization under the independent cascade model. In *WWW (Companion Volume)*, 2015.

[12] A. Goyal, F. Bonchi, and L. V. Lakshmanan. Learning influence probabilities in social networks. In *WSDM*, pages 241–250, 2010.

[13] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan. A data-based approach to social influence maximization. *PVLDB*, 5:73–84, 2011.

[14] A. Goyal, W. Lu, and L. V. Lakshmanan. CELF++: Optimizing the greedy algorithm for influence maximization in social networks. In *WWW (Companion Volume)*, pages 47–48, 2011.

[15] A. Goyal, W. Lu, and L. V. Lakshmanan. Simpath: An efficient algorithm for influence maximization under the linear threshold model. In *ICDM*, pages 211–220, 2011.

[16] K. Jung, W. Heo, and W. Chen. IRIE: Scalable and robust influence maximization in social networks. In *ICDM*, pages 918–923, 2012.

[17] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD*, pages 137–146, 2003.

[18] J. Kim, S.-K. Kim, and H. Yu. Scalable and parallelizable processing of influence maximization for large-scale social networks? In *ICDE*, pages 266–277, 2013.

[19] K. Kutzkov, A. Bifet, F. Bonchi, and A. Gionis. STRIP: Stream learning of influence probabilities. In *KDD*, pages 275–283, 2013.

[20] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *KDD*, pages 420–429, 2007.

[21] Q. Liu, B. Xiang, E. Chen, H. Xiong, F. Tang, and J. X. Yu. Influence maximization over large-scale social networks: A bounded linear approach. In *CIKM*, pages 171–180, 2014.

[22] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294, 1978.

[23] H. T. Nguyen, M. T. Thai, and T. N. Dinh. Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. In *SIGMOD*, pages 695–710, 2016.

[24] N. Ohsaka, T. Akiba, Y. Yoshida, and K. Kawarabayashi. Fast and accurate influence maximization on large networks with pruned monte-carlo simulations. In *AAAI*, pages 138–144, 2014.

[25] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[26] Y. Tang, Y. Shi, and X. Xiao. Influence maximization in near-linear time: A martingale approach. In *SIGMOD*, pages 1539–1554, 2015.

[27] Y. Tang, X. Xiao, and Y. Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *SIGMOD*, pages 75–86, 2014.

[28] V. V. Vazirani. *Approximation Algorithms*. 2001.

[29] H. Zhang, T. N. Dinh, and M. T. Thai. Maximizing the spread of positive influence in online social networks. In *ICDCS*, pages 317–326, 2013.

APPENDIX

A Note on Implementations of TIM$^+$ and IMM

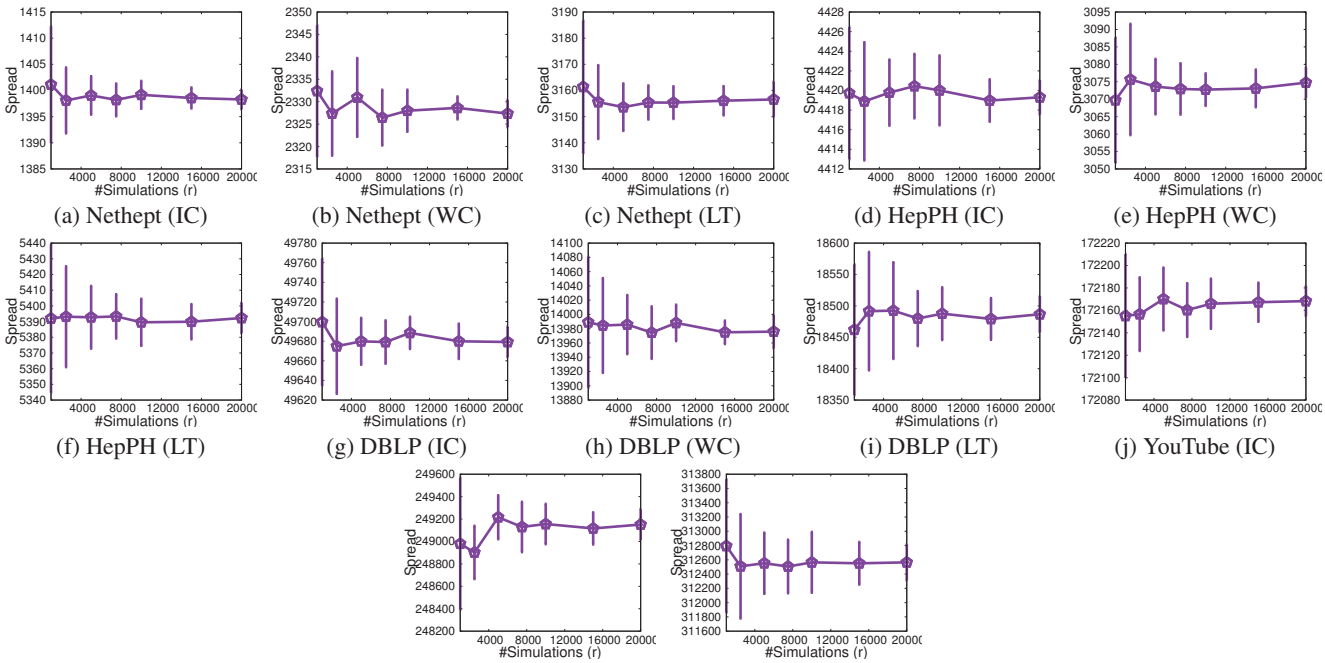
The implementations of both TIM$^+$ and IMM report a spread that is based on extrapolation and not MC simulations. Since this important information is not mentioned in the README files and the only way to identify this missing detail is to study the code of these two techniques, we want to highlight this aspect for the benefit of the IM community. Note that although the README files of both implementations claim it to be the exact same code used to report the results in their respective papers, the authors of these papers have assured us that they report the expected spreads (by running MC simulations) in their paper, and not the extrapolated spreads that their codes output.

B Note on Stopping Criteria of IMRank

The implementation of the IMRank framework provided to us by the authors possessed a bug resulting from an incorrect initialization of node ranks. This led to the early termination of scoring rounds: IMRank stopped right in the first scoring round for $k > 100$. Several engaging discussions before and after the acceptance of our work, followed by a rigorous analysis of the code by the authors, led to the discovery of this minute error. We believe that this find could have an impact on the performance of IMRank as portrayed in Fig. 10f. Since, the authors identified the bug only a day prior to the camera-ready deadline, modifications in the results of IMRank, if any, will be updated on our website: sigdata.github.io/infmax-benchmark.

C Note on Average Node-Lookup Values for CELF/CELF++

As stated in M1 (Sec: 6), the number of node-lookups provide a more robust evaluation of the performance of CELF/CELF++. More fundamentally, node-lookup value for each iteration corresponds to the number of nodes for which the spread is estimated using MC simulations, or simply the number of spread computations for each iteration [14]. To this end, measuring the average number of node-lookups per iteration provides a execution environment independent way of comparing the performance of CELF++ with CELF. In Figs. 13a and 13b, we run 12 independent runs of CELF and CELF++ with $k = 50$ on the Nethept dataset under both LT and WC models, and measure the average number of node-lookups per iteration. As can be seen, no technique can be termed better than the other. On average, CELF requires 12.98 node-lookups (sd=0.81 lookups) compared to 12.06 node-lookups (sd=1.01 lookups) by CELF++ on the LT model, while requiring 17.38 node-lookups (sd=0.87 lookups) when compared to 15.80 node-lookups (sd=0.92 lookups) by CELF++ on the WC model. Although, average node-lookups of CELF++ tend to be slightly lower when compared to that of CELF, their running times are still quite close (M1 in Sec: 6), since the pruning strategy employed by CELF++ is more rigorous and requires more time when compared to that of CELF.

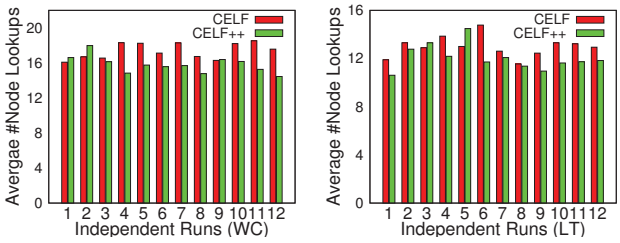


(a) Nethept (IC) (b) Nethept (WC) (c) Nethept (LT) (d) HepPH (IC) (e) HepPH (WC)

(f) HepPH (LT) (g) DBLP (IC) (h) DBLP (WC) (i) DBLP (LT) (j) YouTube (IC)

(k) YouTube (WC) (l) YouTube (LT)

Figure 12: **This figure demonstrates how the mean and the standard deviation (error bar) of the expected spread at 200 seeds varies as the number of MC simulations grows. We choose 10K MC simulations since at his point both the mean and standard deviation stabilizes. The 200 seeds are chosen using IMM [26] on each of the shown datasets. Note that the underlying algorithm, which is IMM in this experiment, does not affect the result. In other words, IMM is only used as a representative.**



(a) Nethept (WC) (b) Nethept (LT)

Figure 13: **Comparing average number of node-lookups for CELF with CELF++ on the Nethept dataset over 12 independent executions.**

Algorithm	Independent Cascade	Linear Threshold
CELF [20]	✓	✓
CELF++ [14]	✓	✓
EaSyIM [10]	✓	✓
IMRank [7]	✓	
IRIE [16]	✓	
PMC [24]	✓	
Static Greedy [8]	✓	
TIM$^+$ [27]	✓	✓
IMM [26]	✓	✓
SIMPATH [15]		✓
LDAG [6]		✓

Table 5: **The diffusion models supported by the algorithms being benchmarked.**

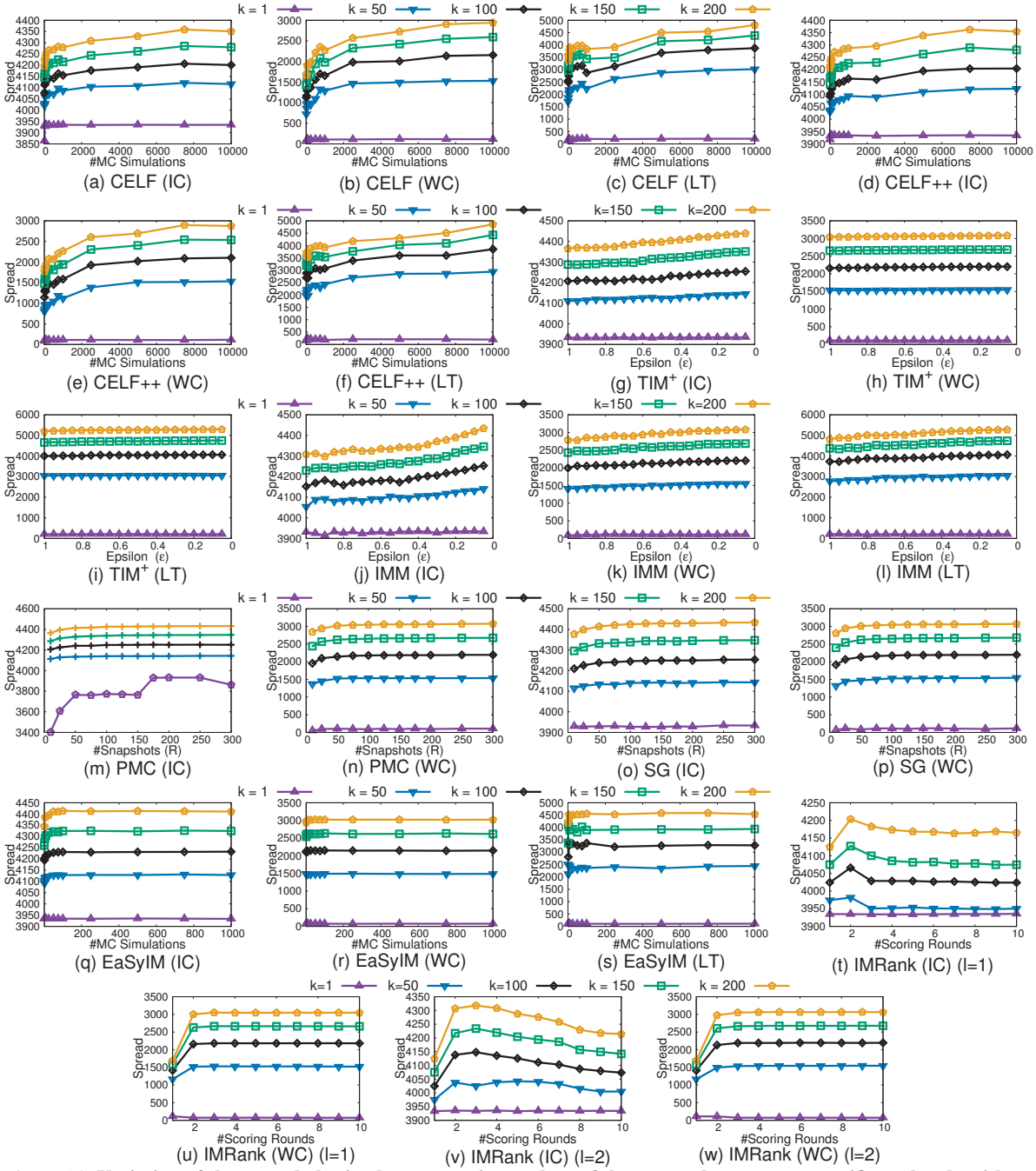
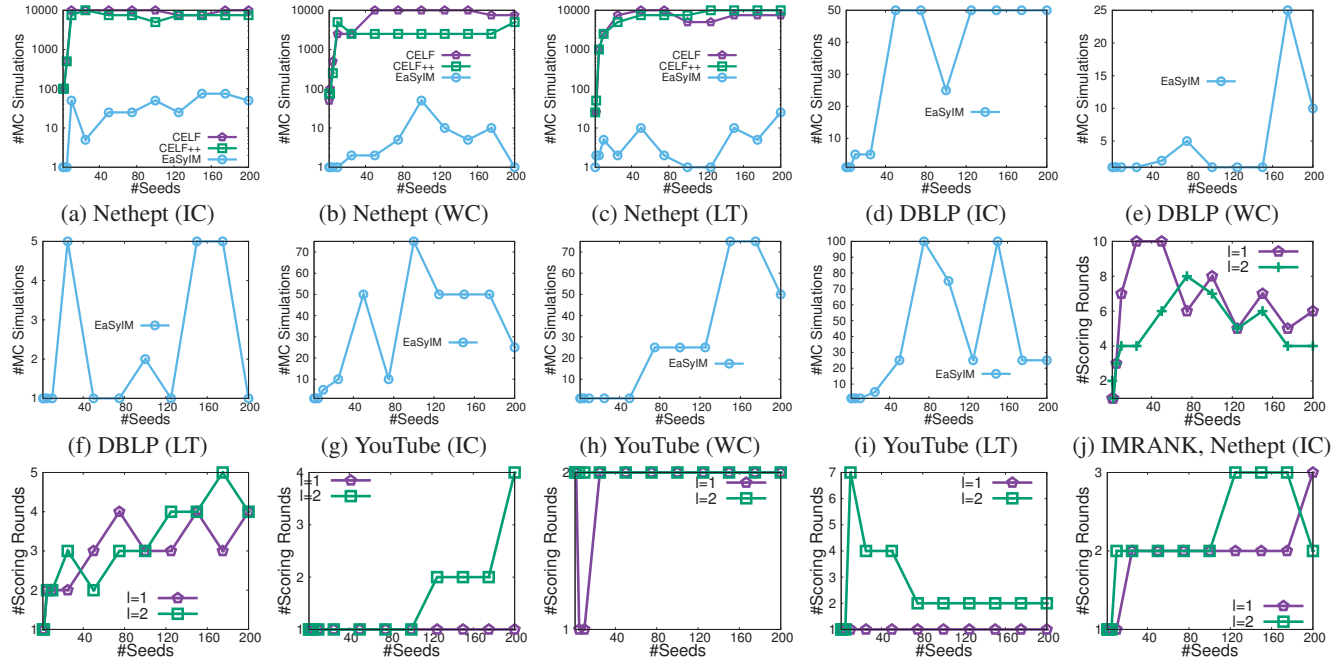


Figure 14: **Variation of the spread obtained across various values of the external parameter specific to the algorithm.**

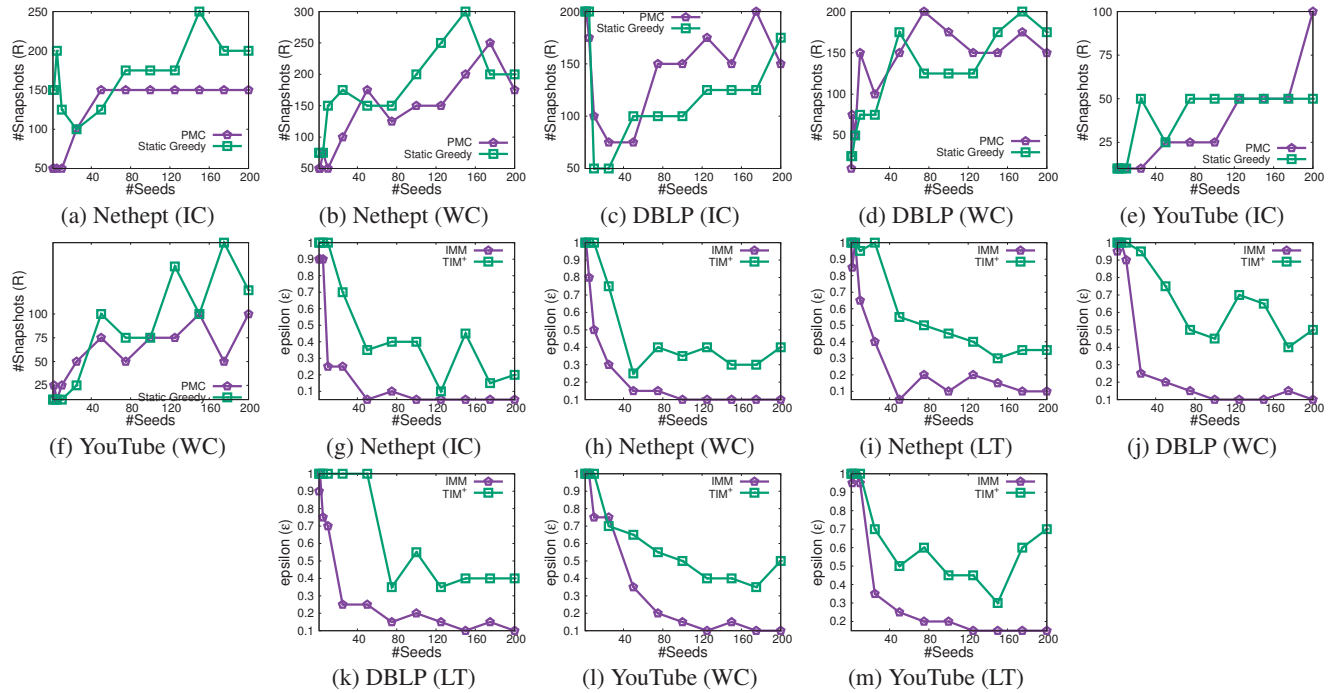


(a) Nethept (IC) (b) Nethept (WC) (c) Nethept (LT) (d) DBLP (IC) (e) DBLP (WC)

(f) DBLP (LT) (g) YouTube (IC) (h) YouTube (WC) (i) YouTube (LT) (j) IMRANK, Nethept (IC)

(k) IMRANK, Nethept (WC) (l) IMRANK, DBLP (IC) (m) IMRANK, DBLP(WC) (n) IMRANK, YouTube (IC) (o) IMRANK, YouTube (WC)

Figure 15: **Results of our experiments to identify the optimal parameter values for the parameters outlined in Table 2.**



(a) Nethept (IC) (b) Nethept (WC) (c) DBLP (IC) (d) DBLP (WC) (e) YouTube (IC)

(f) YouTube (WC) (g) Nethept (IC) (h) Nethept (WC) (i) Nethept (LT) (j) DBLP (WC)

(k) DBLP (LT) (l) YouTube (WC) (m) YouTube (LT)

Figure 16: **Identifying the optimal parameter values for the parameters outlined in Table 2**