A Numerical Evaluation of the Accuracy of Influence Maximization Algorithms

Hautahi Kingi*
Li-An Daniel Wang
Tom Shafer
Minh Huynh
Mike Trinh
Aaron Heuser
George Rochester
Antonio Paredes

ABSTRACT

We develop an algorithm to compute exact solutions to the influence maximization problem using concepts from reverse influence sampling (RIS). We implement the algorithm using GPU resources to evaluate the empirical accuracy of theoretically-guaranteed greedy and RIS approximate solutions. We find that the approximations yield solutions that are remarkably close to optimal — usually achieving greater than 99% of the optimal influence spread. These results are consistent across a wide range of network structures.

KEYWORDS

Networks, Influence Maximization, GPU computing

1 INTRODUCTION

Influence maximization (IM) in networks is a well-studied problem with applications in viral marketing, epidemiology, and public health among many others. The basic problem attempts to select a set of initial seed nodes in a network to maximize the resulting expected spread of influence initiated at those nodes.

Finding a solution to this problem suffers from a combinatorial explosion in the number of candidate seed sets as the size of the network or seed set increases. For example, in a relatively small network of 1,000 nodes, there are approximately 8 trillion different possible candidates of seed sets of size 5. The large computational burden involved in solving this NP-hard problem prompted substantial research over the last decade-and-a-half aimed at developing approximate solutions. These approximations can broadly be classified into two groups — theoretically-guaranteed algorithms that are proven to obtain solutions within a given factor of the optimal solution, and heuristics that sacrifice theoretical guarantees for speed.

This article focuses on the former. All theoretically-guaranteed approximation algorithms in the literature [7, 16, 20, 23, 31] provide solutions that achieve an expected spread of influence within a factor of $(1 - 1/e) \approx 0.63$ of the optimal solution with a given probability. This "gold-standard" is common across many computational fields because it derives from theory about how the maxima of sub-modular functions can be approximated via greedy hill-climbing algorithms using the "pigeon-hole" principle. This theoretical guarantee, however, leaves a lot of room for the actual empirical accuracy of these algorithms. In other words, where in the $[0.63, 1]$-interval does an approximate solution fall? Answering this question of course requires the ability to compute the optimal solution, the impracticality of which has motivated the literature in the first place.

However, two recent advances now render feasible the computation of optimal solutions for non-trivial networks. The first advance, which is theoretical, is the introduction of random reverse reachable sets to the IM problem by [7], which dramatically improved the computational speed of theoretically-guaranteed algorithms by eliminating the need for repeated Monte Carlo simulations for each candidate seed node. These have come to be known as Reverse Influence Sampling (RIS) algorithms. We show that the same concept can be leveraged to more efficiently compute exact solutions. The second advance is the dramatic improvement in computing power since the original greedy algorithm was developed.

This article combines these two advances to distribute our "embarrassingly parallel" RIS-based exact solution algorithm across multiple threads of a GPU. We compute the exact optimal solutions to the IM problem across a wide range of 100-node networks to assess the empirical performance of theoretically-guaranteed algorithms. We show that, perhaps surprisingly, the theoretically guaranteed methods achieve a spread that is within 99% of the exact optimum solution, which is consistent across a range of network structures.

Upon completion of an initial draft of this article, we were directed towards a paper by Li et al. [24], which attempts to answer the same question. Like us, they exploit the concept of random reverse reachable sets and also find the unexpected result that theoretically-guaranteed heuristics perform near-optimally. However, unlike us, they solve for the exact problem using a novel linear programming method. Instead, our GPU-driven technique retains the basic structure of the RIS algorithms. Furthermore, whereas [24] implement their method on a limited number of networks, we evaluate how the accuracy of the approximations depend on network structure.

The article proceeds as follows. Section 2 discusses the key concepts required to understand the methods in this article, including a formal statement of the IM problem, a description of the two general approaches to computing theoretically-guaranteed approximate solutions and an explanation of key network properties. Section 3 outlines the related literature and Section 4 develops our approach to the computation of exact solutions. Section 5 presents our results. Section 6 concludes.

*Corresponding author. hautahikingi@gmail.com

2 PRELIMINARIES

2.1 Influence Maximization Problem

Let a network be represented by a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges representing connections between two nodes. Let $S \subseteq V$ denote the "seed set," which is a subset of nodes selected to initiate an influence propagation process. Let $\sigma : P(V) \mapsto \mathbb{R}$ denote the spread function that returns the expected number of nodes activated by the influence propagation mechanism initiated from any given set of seed nodes S. The seminal work of Kempe et al. [20] formulated Influence Maximization (IM) as the following combinatorial optimization problem:

$$\max_{S \subseteq V} \sigma(S) \text{ s.t. } |S| \leq k$$

Any algorithm solving this problem takes the graph G, an integer k, and the spread function $\sigma(\cdot)$ as input and generates a seed set S of size at most k, with the intention that the expected number of nodes influenced by S is as large as possible. At the core of the computational burden and NP-hardness of this problem is that the domain of S has $\binom{n}{k}$ elements, where $n = |V|$ is the number of nodes in the graph.

As mentioned, the spread function is defined by a propagation model that determines the diffusion of influence across a network. Among the many potential candidates for the network diffusion process in the literature, we focus on one of the most prominent — the Independent Cascade (IC) model. Under the IC model, the process unfolds in discrete time steps. When $v \in V$ first becomes active, it has a single opportunity to activate each currently inactive neighbor w, which it does with probability p —a parameter of the system. If w has multiple newly activated neighbors, their attempts are sequenced in an arbitrary order. Whether or not v succeeds, it cannot make any further attempts to activate its neighbors in subsequent rounds. The process ends when no new nodes are activated.

2.2 Theoretically Guaranteed Approximations

The IM literature has focused on the development of efficient algorithms and heuristics to achieve approximate solutions to the problem. These algorithms can be broadly categorized according to whether or not they achieve a theoretically-guaranteed performance bound. Those with proven performance guarantees can further be classified into methods based on Monte Carlo simulations and those based on Reverse Influence Sampling (RIS). In this article, we compare the exact solutions to the approximate solutions obtained from a representative algorithm from each class. Furthermore, we exploit key features of the RIS methods to construct our exact solutions. Each of these methods is described in turn below.

2.2.1 Monte Carlo Methods. Kempe et al. [20] proposed a simple greedy hill-climbing algorithm, which selects the seed set S by iteratively selecting successive nodes, each time choosing the node that provides the largest marginal increase in the spread function σ until k nodes are selected. More formally, the node added to the seed set in time step $t \in 1, \ldots, k$, v_t^*, is determined as follows

$$v_t^* = \underset{v \in V \setminus S_{t-1},\, t \leq k}{\arg\max} \sigma\left(S_{t-1} \cup v\right) - \sigma\left(S_{t-1}\right)$$

Kempe et al. [20] prove that this simple algorithm is guaranteed to find a seed set with an expected influence spread that is at least a constant fraction $1 - 1/e - \epsilon \approx 0.63$ of the optimum solution. A sketch of the pseudocode used to implement this Greedy algorithm is presented in Algorithm 1.

There are two computationally burdensome components to the Greedy algorithm. The first is the node choice component described by the for-loop beginning at line 4 of Algorithm 1, which searches every node in the graph as a potential candidate for the next seed node, and ultimately determines the number of times $\sigma(\cdot)$ must be evaluated. The second computationally intensive component is the evaluation of $\sigma(\cdot)$ itself in line 5, which is #P-hard [9] and is therefore usually estimated with Monte-Carlo simulations, each of which requires approximately $O(m)$ time to simulate the propagation process across each edge in the graph, where $m = |E|$ is the number of edges in the graph. 10,000 iterations are typically performed. Assuming *nsim* Monte Carlo simulations for each evaluation of $\sigma(\cdot)$, the Greedy algorithm requires $O(kmn \cdot nsim)$ total running time.

It is important to note that there are two approximations occurring here. The $1/e$ component of the error term is attributable to the approximation caused by the iterative construction of the seed set rather than the consideration of the entire domain of seed set candidates, which is represented in the two nested for-loops beginning on lines 2 and 4 in Algorithm 1. The ϵ term, on the other hand, results from the fact that $\sigma(\cdot)$ must be approximated via repeated Monte Carlo simulations, which are performed implicitly in line 5 and run in approximately $O(m \cdot \text{POLY}(\epsilon^{-1}))$ time. It is the $1/e$ component which is of interest, because ϵ decreases with the number of Monte Carlo simulations performed and can therefore be reduced to any arbitrarily small value by the researcher.[1]

Algorithm 1 Pseudocode for Greedy Algorithm

Input: graph $G = (V, E)$, $p \in [0, 1]$, integer k
Output: size-k seed set S

```
1:  S = ∅
2:  for (i in 1 : k) do
3:      max = 0
4:      for (v in V \ S) do
5:          spread = σ(S ∪ v) − σ(S)
6:          if spread > max then
7:              max, best = spread, v
8:          end if
9:      end for
10: end for
```

This greedy algorithm has become somewhat of a benchmark case in the literature, given that no other algorithms achieve a greater theoretical performance guarantee. For this reason, we compare the approximate solution derived using this algorithm with the exact optimal solution.

[1] "Exact" solutions in this article will technically be $(1 - \epsilon)$ approximations, although in practice ϵ is very small.

2.2.2 Reverse Influence Sampling (RIS) Methods. Although the greedy procedure is conceptually very simple and orders of magnitude faster than computing an exact solution, it is still notoriously inefficient, which has prompted a large literature dedicated to finding faster approximate solutions. The fastest theoretically-guaranteed approximations are currently the class of RIS algorithms, which exploit the concept of a random reverse reachable set introduced to the IM problem by [7].

A reverse reachable (RR) set for an arbitrary node $v \in V$ is generated by first sampling a graph, g, from the distribution generated by removing each edge $e \in E$ according to its propagation probability $1 - p_e$ and then taking the set of nodes in g that can "reach" v.[2] A random reverse reachable (RRR) set is an RR set for a node selected uniformly at random from V. An example of the construction of an RR set is presented in Figure 1, which shows the sampling of the edges from the original network to produce an instance g and the resulting RR set of node D surrounded by the dashed lines.

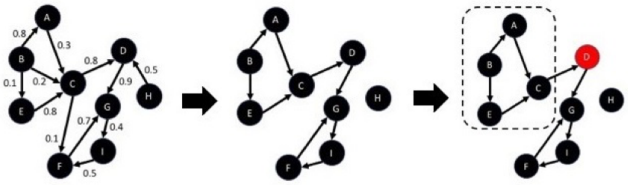

Figure 1: Example of a Reverse Reachable Set

Intuitively, if a node u appears in an RR set of another node v, then a diffusion process from a seed set containing u has some probability of activating v because there is a directed path from u to v. This connection between RR sets and node activations is formalized in a lemma, which states that the probability that a diffusion process from any seed set S will activate any node v is equal to the probability that S overlaps an RR set for v (at least one node in S is contained within a RR set for v).[3] This lemma implies a two-step approximation algorithm presented as pseudocode in Algorithm 2. [7] showed that this algorithm returns a theoretically guaranteed $(1 - 1/e - \epsilon)$-approximate solution with a known probability.

The first step generates a set $\mathcal{R}$ of θ independent RRR sets.[4] This was referred to as the "BuildHypergraph" step by [7]. The second step solves the maximum coverage problem of selecting k nodes to cover the maximum number of RRR sets in $\mathcal{R}$ using a greedy algorithm. In practice, the seed set is constructed by iteratively choosing the node that appears in the most RRR sets and upon the selection of each node, the RRR sets featuring that node are removed. This corresponds to the "BuildSeedSet" step in [7].

[2]We assume $p_e = p, \forall e$ throughout.

[3]This is Lemma 1 in, for example, [18].

[4]The value of θ determines both the runtime of the algorithm and ϵ. However, the relationship between θ and ϵ is a function of the optimal solution, as shown in Theorem 4.1. The literature has focused on determining increasingly tighter values of θ to reduce the runtime through various techniques like limiting the total number of edges examined during the generation process to a pre-defined threshold [7], using Chernoff bounds [34] and adopting martingale methods [33], among others [18, 31]. We focus on the two computational steps common to all RIS methods and set $\theta = 100,000$. Because this affects both the approximate and exact solutions equally, the proportional difference between the solutions is approximately independent of θ so long as it ensures $\epsilon << e$.

Algorithm 2 Pseudocode for RIS Algorithm

Input: graph $G = (V, E)$, $p \in [0, 1]$, integer k, integer θ
Output: size-k seed set S

STEP 1: Generate set of RRRSs
1: $\mathcal{R} = \emptyset$
2: **for** (i in $1 : \theta$) **do**
3: $\mathcal{R}$.append(RRRS(G, p))
4: **end for**

STEP 2: Maximum Coverage Algorithm
5: $S = \emptyset$
6: **for** (i in $1 : k$) **do**
7: **for** (v in $V \setminus S$) **do**
8: coverage = $\mathcal{F}_{\mathcal{R}}(S \cup v) - \mathcal{F}_{\mathcal{R}}(S)$
9: **if** coverage > max **then**
10: max, best = coverage, v
11: **end if**
12: **end for**
13: $S = S \cup$ best
14: **end for**

This approach works because for any seed set S, the fraction of RRR sets in $\mathcal{R}$ covered by S, denoted by $\mathcal{F}_{\mathcal{R}}(S)$, is an unbiased estimator of the spread $\sigma(S)$. Therefore, a seed set S_k^* that covers a large number of RRR sets in $\mathcal{R}$ is likely to have a large expected influence, which makes S_k^* a good solution to the IM problem. Intuitively, if we select a node v uniformly at random, determine the set of nodes that would have influenced v and then repeat this process multiple times, nodes that appear often as "influencers" are likely good candidates for the most influential nodes. This intuition is formalized by Lemma 2.1, which states that the error produced by estimating $R(S)$ with the fraction of RRR sets within $\mathcal{R}$ covered by S is bounded.[5]

Lemma 2.1. *Suppose $\mathcal{R}$ represents a collection of θ RRR sets, where θ satisfies the following bound:*

$$\theta \geq (8 + 2\epsilon) \cdot n \cdot \frac{l \log n + \log \binom{n}{k} + \log 2}{OPT \cdot \epsilon^2}$$

Then for any set S of at most k nodes, the following inequality holds with at least $1 - n^{-l}/\binom{n}{k}$ probability:

$$|n \cdot \mathcal{F}_{\mathcal{R}}(S) - \sigma(S)| < \frac{\epsilon}{2} OPT$$

where $\mathcal{F}_{\mathcal{R}}(S)$ is the fraction of RRR sets in $\mathcal{R}$ covered by S.

The key to the improved computational performance of the RIS algorithms over the Greedy procedure described in Algorithm 1 is that it does not repeat the spread computation procedure to incrementally construct a solution. Instead, all the Monte Carlo simulations/samplings are performed up front, and then the entire seed set is selected using the resulting set $\mathcal{R}$. This effectively replaces the $O(nk)$ instances of the spread computation in line 5 of Algorithm 1 with a counting procedure in line 8 and the preprocessing step in lines 1-4 of Algorithm 2, which eliminates the

[5]Examples of various versions of this lemma in the literature are Lemma 3 in [34], Lemma 3 in [33], and Lemma 5 in [31].

need to simulate repeated propagation cascades. Intuitively, the method is fast because it avoids the "wasted" spread computations of the greedy algorithm and most of its successors because the generated RRR sets are used to inform the spread of all nodes within the network.

2.3 Network Structure

Despite their enormous variety, networks tend to show remarkably similar structural characteristics. This surprising uniformity has prompted significant interest in the effect of network topologies on various phenomena and the development of generative frameworks to formally model these properties.

One such common property is the "small world" phenomenon in which the average distance between nodes is usually short and scales logarithmically with the number of nodes. This is captured in the prototypical random graph framework developed by Erdős and Rényi [14], in which links form between each pair of nodes with a fixed number of nodes n with probability q. As the value of q grows and the graph becomes less sparse, the diameter of the graph shrinks. Like many network phenomena, the shrinkage of the diameter changes non-linearly with q as the graph undergoes a phase transition from many equally sized clusters to one giant component.

Another common network property is the presence of cliques, so that two nodes that are both neighbors of the same third node are more likely to also be neighbors of one another. This effect is quantified by the clustering coefficient [30, 35] and can be studied using a framework developed by Watts and Strogatz [35], which begins with a regular ring lattice with n nodes and m edges and attempts to randomly "rewire" each edge of the graph sequentially with probability β. If an edge is selected for rewiring, a new destination node is selected at random if an edge connecting the target and the source does not already exist.

The Watts-Strogatz model is convenient for analyzing changes in clustering within a network. The model accounts for clustering while retaining the short average path lengths of the Erdos-Renyi model. The underlying lattice structure of the model produces a locally clustered network, while the randomly rewired links dramatically reduce the average path lengths. Varying the value of β allows us to interpolate between a network structure that is very close to an Erdos-Renyi network $G(n, q)$ with $q = K/n - 1$ at $\beta = 1$, where K is the degree of each node in the regular lattice, to a regular ring lattice at $\beta = 0$ with high clustering but large diameter.[6] Again, however, these changes do not occur uniformly. As β increases from zero, the diameter drops precipitously from its initial value and quickly approaches values typical of random graphs. On the other hand, the clustering coefficient also decreases with β, but much more slowly.

The models above produce approximately Gaussian degree distributions. However, many networks exhibit a right-skewed or "scale-free" degree distribution in which a significant number of nodes have a very large degree [4]. This results in the power-law distribution $P(d) = Cd^{-\gamma}$ for the probability that a node has d connections to other nodes, where γ generally ranges between 2 and 3.

Section 5 explores whether the approximation accuracy of the Greedy and RIS algorithms differs across various network properties using the generative models described above.

[6] The $\beta = 1$ version is not identical to the Erdos-Renyi model because it enforces each node to have at least $K/2$ connections whereas there is no restriction on edges for a given node in Erdos-Renyi.

3 RELATED LITERATURE

The problem of finding the most influential set of nodes to target for a diffusive process was first posed by Domingos and Richardson [12, 13] and formalized into the combinatorial problem by [20]. Many faster algorithms followed, most of which were heuristics with solutions that were not guaranteed to achieve any performance bound [9, 10, 19, 21, 29]. This article investigates the empirical accuracy of the relatively few theoretically-guaranteed approximations algorithms [7, 16, 23].

Our research question and approach are most closely related to recent work by [24]. Like us, they exploit the concept of random reverse reachable sets and also find the unexpected result that the theoretically-guaranteed heuristics perform near-optimally on the networks they evaluate. However, unlike us, they solve for the exact problem using a novel linear programming method, which can scale to extremely large graphs with billions of nodes whereas our GPU-driven technique is limited to relatively small sized networks in the hundreds of nodes. However, we also systematically investigate whether the accuracy of the approximations differ across different network structures. This practice of investigating how a network phenomenon varies with respect to network structure has a long history in network science, including epidemic and disease spreading [8, 28, 32], a network's vulnerability to attack [2, 6, 11, 27], the ability to predict the spread of a contagion [8], and non-equilibrium phase transitions [26].

This article is also related to a small literature broadly concerned with the practical implementation of IM algorithms and in particular the choice faced by a practitioner in choosing an appropriate algorithm ex-ante. For example, Akbarpour et al. [1] ask what conditions or network structures require extensive IM algorithms, and show that randomly choosing $k+1$ nodes can often outperform optimally chosen k nodes. It would be useful to know when such simple heuristics, rather than complex IM approaches, suffice to compute influential nodes. Our results suggest that network structure likely plays a minimal role in guiding this decision.

This article also helps to confirm the legitimacy of the general approach in the IM literature of comparing spreads achieved by a given method to an established algorithm with proven performance bounds. These comparisons are more meaningful when the quality of a given algorithm is known, which depends on the actual performance rather than a lower bound.

Finally, a key feature of our method is the use of GPU resources. Although a large literature investigates how to parallelize fundamental network algorithms [3, 5, 17, 22], there are relatively few papers that explicitly describe the implementation of parallelization techniques in the context of IM [15, 21, 25]. In these cases, high performance computing resources tend to be used to expand the

applicability of existing algorithms to larger graphs or to demonstrate the distributed nature of a novel algorithm. There has been little focus on the use of parallelization to compute exact solutions.

4 EXACT SOLUTION METHOD

To compare the empirical accuracy of the approximation procedures described in Section 2.2, we must first compute an exact solution to the NP-hard IM problem. This is not trivial, as evidenced by the significant effort attributed to the development of approximation algorithms. We therefore rely on two key insights to make this exercise tractable. The first is the RIS-Exact algorithm. The second is its embarrassingly parallel structure, which allows us to distribute independent iterations across multiple threads within a GPU architecture. Both insights are described in more detail below.

4.1 RIS-Exact Algorithm

Algorithm 3 presents the RIS-Exact algorithm. It retains the same 2-step structure of the RIS approximation algorithm described in Section 2.2, and indeed the first step is identical. The second step, however, is modified to comprehensively loop over all candidate seed sets rather than to construct the seed set iteratively. This replaces the nested for-loops beginning on lines 6-7 in Algorithm 2, which runs in $O(nk)$ time, with one larger loop beginning on line 7 in Algorithm 3 that computes the coverage of all possible seed sets and runs in $O\binom{n}{k}$ time. This simple adaptation achieves an exact solution to the IM problem, as described in Theorem 4.1.

Theorem 4.1. *Suppose θ satisfies the following bound:*

$$\theta \geq (8 + 2\epsilon) \cdot n \cdot \frac{l \log n + \log \binom{n}{k} + \log 2}{OPT \cdot \epsilon^2}$$

Algorithm 3 returns a $(1 - \epsilon)$-approximate solution to the IM problem with at least $1 - n^{-l}$ probability.

Proof. Lemma 2.1 states that

$$|n \cdot \mathcal{F}_{\mathcal{R}}(S) - \sigma(S)| < \frac{\epsilon}{2} OPT$$

holds with at least $1 - n^{-l}/\binom{n}{k}$ probability for any given size-k set S. It follows from Boole's inequality that the above holds simultaneously for all size-k sets with at least $1 - n^{-l}$ probability.

Now let $S_k^{\mathcal{R}+}$ be the seed set obtained from Algorithm 3. Lemma 2.1 implies that

$$\sigma(S_k^{\mathcal{R}+}) \geq n\mathcal{F}(S_k^{\mathcal{R}+}) - \frac{\epsilon}{2} OPT$$

Now let S_k^0 be the exact solution to the IM problem. Although this set maximizes $\sigma(\cdot)$, we know from the definition of $S_k^{\mathcal{R}+}$ that $\mathcal{F}(S_k^{\mathcal{R}+}) \geq \mathcal{F}(S_k^0)$, so it follows that

$$\sigma(S_k^{\mathcal{R}+}) \geq n\mathcal{F}(S_k^0) - \frac{\epsilon}{2} OPT$$

Applying Lemma 2.1 once again implies that

$$\begin{aligned} \sigma(S_k^{\mathcal{R}+}) &\geq \sigma(S_k^0) - \frac{\epsilon}{2} OPT - \frac{\epsilon}{2} OPT \\ &= (1 - \epsilon) \cdot OPT \end{aligned}$$

as required. □

Algorithm 3 Pseudocode for RIS-Exact Algorithm

Input: graph $G = (V, E)$, $p \in [0, 1]$, integer k, integer θ
Output: size-k seed set S

STEP 1: Generate set of RRRSs
1: $\mathcal{R} = \emptyset$
2: **for** (i in $1 : \theta$) **do**
3: $\mathcal{R}$.append(RRRS(G, p))
4: **end for**

STEP 2: Exhaustive Search
5: candidates, max = set of all $\binom{n}{k}$ node combinations, 0
6: **for** (set in candidates) **do**
7: coverage = $\mathcal{F}_{\mathcal{R}}$(set)
8: **if** coverage > max **then**
9: max, S = coverage, set
10: **end if**
11: **end for**

4.2 GPU Implementation

RIS-Exact retains the "combinatorial explosion" property required to compute an exact solution, but replaces the Monte Carlo simulations in each round with a procedure that counts the frequency of occurrence of events, which is a relatively cheap calculation. This is the same reason why RIS approximation methods outperform their Monte Carlo-based counterparts in terms of efficiency. However, Step 2 of Algorithm 3 still requires significant computational resources. Fortunately, it is "embarrassingly parallel" and can be be distributed across multiple computing nodes to speed up computation.

We experimented with various high performance computing configurations on Amazon Web Services, including a 72-core shared-memory architecture and a distributed setting with 19 c5.2xlarge instances for a total of 152 virtual CPUs. We settled upon a configuration with an Nvidia Tesla K80 GPU implemented on a p2.xlarge instance of an EC2 "Deep Learning Base AMI (Amazon Linux) Version 19.1 (ami-00a1164673faf2ac3)" instance. To understand the magnitude of the speed-up, running RIS-Exact on a 50-node network with a seed set size of $k = 4$ on a standard desktop with a 2.7 GHz Intel Core i5 processor and 8GB of RAM took 66 minutes. The same run on a 72-core shared-memory setup took 3 minutes while our GPU-based framework reduced the run time to 6 seconds.

The procedures were implemented in Python using the Numba library to access and execute tasks on the Nvidia CUDA cores. Efficient CUDA-aware computations require compact vector-like object representations. For Step 1, we encoded $\mathcal{R}$ as a $\theta \times n$ binary array, where each row represents an RRR set and the presence of a node in that set is indicated by the corresponding entry set to `TRUE`. Each row of the array is distributed to independent GPU threads. Similarly in Step 2, the array object is size $1 \times \binom{n}{k}$ with each entry representing the number of RRR sets in $\mathcal{R}$ covered by the corresponding seed set.[7]

[7]The code to generate the results is available at https://github.com/hautahi/IM-Evaluation.

5 NUMERICAL ANALYSIS

We compare approximate solutions to optimal solutions by constructing an approximation factor $\delta = R(S_{APP})/R(S_{OPT}) \in [0, 1]$, where S_{APP} is the seed set obtained via Greedy or RIS and S_{OPT} is the seed set obtained via RIS-Exact. A larger δ, which represents the fraction of an optimal solution achieved by the approximation, indicates a more accurate estimation.

We focus our experiments on the three classes of networks discussed in Section 2.3 - the Erdős and Rényi [14] network, the Watts and Strogatz [35] network, and a network with node degrees d sampled from a scale-free distribution $P(d) = Cd^{-\gamma}$ with C set to achieve a similar network density to the other graph instances. These frameworks allow us to isolate a particular component of network structure and systematically study how δ varies with the structural parameters of the models. For example, varying the Erdos-Renyi probability of link formation between a pair of nodes, q allows us to study the effect of edge density and network diameter on δ. Similarly, varying the re-wiring parameter β in Watts-Strogatz networks allows us to investigate the effect of clustering on δ. It has also been shown that the scale-free parameter γ crucially determines a number of network phenomena, so it is also of interest to directly examine how it affects δ.

We considered 29 different parameter configurations across the three frameworks ($q \in [0.1, 0.9]$, $\beta \in [0, 0.9]$, and $\gamma \in [1.5, 4]$). For each configuration, we generated 10 graph instances and ran each algorithm for a range of propagation probabilities p between 0.01 to 0.7. We also experimented with a range of network sizes (50-500) and seed set sizes (2-7) that kept the number of candidate seed sets for RIS-Exact under 100 million and computation time under 30 minutes per algorithm. We found little variation in δ with n or k so we settled on presenting results for networks of size $n = 100$ with a seed set size of $k = 4$, each of which required run times of approximately 180 seconds per solution. In total, we simulated solutions for 1,880 different combinations of network configurations, propagation probabilities and graph instances, which required a total computation time of about 4 days. We set θ equal to 1 million and used 10,000 Monte Carlo simulations to estimate σ in the Greedy algorithm.

There are two main numerical results. The first is that the greedy and RIS approximation procedures achieve almost identical results to the exact optimal solution, which is consistent with the findings of [24]. Fundamentally, both approximations in practice achieve close to the upper-bound of their theoretically-guaranteed solution range of $[0.63, 1]$. Figure 2 plots the mean and 95% confidence bands of the approximation factor δ for both approximation algorithms across all three network types, where the mean is taken over all network configuration parameters, propagation probabilities and graph instances. It demonstrates that the approximation algorithms yield solutions that almost always achieve a spread well above 99% of the optimal solution, regardless of network structure.

Figure 2 also appears to show that the RIS algorithm is superior to the greedy algorithm. Although this is possible, because theory does not guarantee the methods yield identical results, we caution against this interpretation for three reasons. First, the RIS and RIS-Exact solutions are directly comparable because they both act on the same set $\mathcal{R}$. There is therefore no random simulation error in δ for RIS, unlike greedy. Second, we cannot ensure that the error term ϵ described in Section 2.2 is identical between the RIS and greedy solutions, so the observed difference in δ may simply reflect a larger ϵ value for the greedy procedure. Third, the scale of the axes suggest that the difference is more salient than what it is.

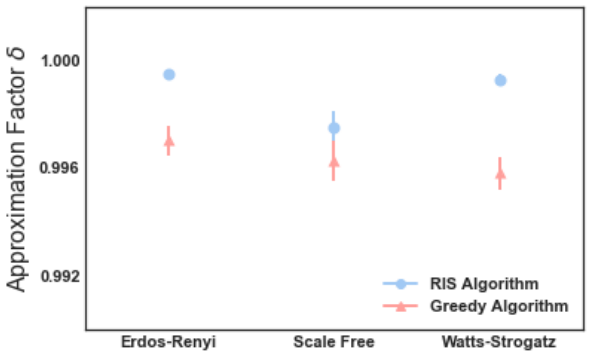

Figure 2: IM Approximation Algorithms are Almost Optimal

The second main numerical result is that there is very little relationship between network structure (at least those related to clustering, diameter, density, and degree distribution) and the accuracy of the approximation algorithms. The only relatively significant relationship we uncovered is presented in Figure 3, which shows the relationship between the mean δ for each approximation method (averaged across propagation probabilities and graph instances with 95% confidence bands) and the Erdos-Renyi link-forming parameter q. As network density increases through an increase in q, the approximate solutions tend to improve.

One potential explanation for this is that the probability of a node having degree d in an Erdos-Renyi network is equal to q^d. As q increases, higher degree nodes are more likely. In the absence of clustering, influential nodes tend to therefore be spread evenly across the network, which avoids the interdependence among the influence of nodes that tends to make the iterative construction of the seed set diverge. There is no such relationship between δ and, say, the rewiring parameter of the Watts-Strogatz network because the constant node-to-edge ratio of Watts-Strogatz instances ensures that changes in β do not affect the expected degree of its nodes.

6 CONCLUDING REMARKS

We explored how a seminal and a state-of-the-art theoretically-guaranteed influence maximization (IM) approximation algorithm perform in comparison to an optimal solution, and whether this performance differed with respect to network structure. To do so, we proposed RIS-Exact - a fast algorithm that exploits reverse influence sampling and GPU computing resources to calculate the exact solution to the IM problem. We applied RIS-Exact to a collection of network instances drawn from alternative configurations of Erdos-Renyi, Watts-Strogatz and scale-free graph frameworks.

We found that both approximations perform remarkably well and generally achieve at least 99% of the optimal solution. We

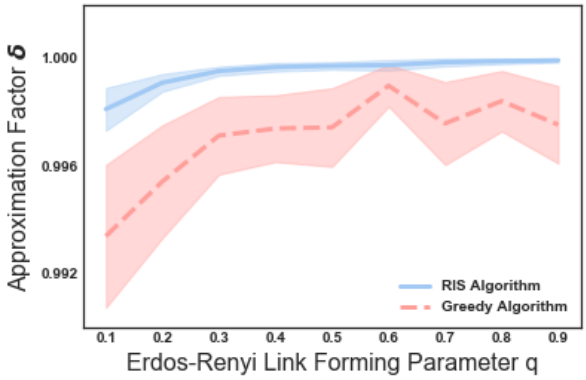

Figure 3: Accuracy Increases with Network Density

found little relationship between network properties and accuracy other than an apparent positive correlation between accuracy and network density.

Our results suggest a possibility that there is something inherent to how influence propagates across a network that renders standard sub-modular function results to be too conservative. We leave this theoretical investigation to future work.

REFERENCES

[1] Mohammad Akbarpour, Suraj Malladi, and Amin Saberi. 2018. Diffusion, Seeding, and the Value of Network Information. In *Proceedings of the 2018 ACM Conference on Economics and Computation*. ACM, 641–641.

[2] Réka Albert, Hawoong Jeong, and Albert-László Barabási. 2000. Error and attack tolerance of complex networks. *nature* 406, 6794 (2000), 378.

[3] David A Bader and Kamesh Madduri. 2008. SNAP, Small-world Network Analysis and Partitioning: an open-source parallel graph framework for the exploration of large-scale networks. In *2008 IEEE International Symposium on Parallel and Distributed Processing*. IEEE, 1–12.

[4] Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *science* 286, 5439 (1999), 509–512.

[5] Jiri Barnat, Petr Bauch, Lubos Brim, and Milan Ceska. 2011. Computing Strongly Connected Components in Parallel on CUDA. *2011 IEEE International Parallel Distributed Processing Symposium* (2011), 544–555.

[6] Béla Bollobás and Oliver Riordan. 2004. Robustness and vulnerability of scale-free random graphs. *Internet Mathematics* 1, 1 (2004), 1–35.

[7] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. 2014. Maximizing social influence in nearly optimal time. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 946–957.

[8] Duan-Bing Chen, Rui Xiao, and An Zeng. 2014. Predicting the evolution of spreading on complex networks. *Scientific reports* 4 (2014), 6108.

[9] Wei Chen, Chi Wang, and Yajun Wang. 2010. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1029–1038.

[10] Wei Chen, Yajun Wang, and Siyu Yang. 2009. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 199–208.

[11] Reuven Cohen, Keren Erez, Daniel Ben-Avraham, and Shlomo Havlin. 2000. Resilience of the internet to random breakdowns. *Physical review letters* 85, 21 (2000), 4626.

[12] Pedro Domingos and Matt Richardson. 2001. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 57–66.

[13] Pedro Domingos and Matt Richardson. 2002. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 61–70.

[14] Paul Erdős and Alfréd Rényi. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci* 5, 1 (1960), 17–60.

[15] Sainyam Galhotra, Akhil Arora, and Shourya Roy. 2016. Holistic influence maximization: Combining scalability and efficiency with opinion-aware models. In *Proceedings of the 2016 International Conference on Management of Data*. ACM, 743–758.

[16] Amit Goyal, Wei Lu, and Laks VS Lakshmanan. 2011. Celf++: optimizing the greedy algorithm for influence maximization in social networks. In *Proceedings of the 20th international conference companion on World wide web*. ACM, 47–48.

[17] Pawan Harish and PJ Narayanan. 2007. Accelerating large graph algorithms on the GPU using CUDA. In *International conference on high-performance computing*. Springer, 197–208.

[18] Keke Huang, Sibo Wang, Glenn Bevilacqua, Xiaokui Xiao, and Laks Lakshmanan. 2017. Revisiting the stop-and-stare algorithms for influence maximization. *Proceedings of the VLDB Endowment* 10 (05 2017), 913–924. https://doi.org/10.14778/3099622.3099623

[19] Kyomin Jung, Wooram Heo, and Wei Chen. 2012. Irie: Scalable and robust influence maximization in social networks. In *2012 IEEE 12th International Conference on Data Mining*. IEEE, 918–923.

[20] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 137–146.

[21] Jinha Kim, Seung-Keol Kim, and Hwanjo Yu. 2013. Scalable and parallelizable processing of influence maximization for large-scale social networks?. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE, 266–277.

[22] Dominique LaSalle and George Karypis. 2013. Multi-threaded graph partitioning. In *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*. IEEE, 225–236.

[23] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. 2007. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 420–429.

[24] Xiang Li, J David Smith, Thang N Dinh, and My T Thai. 2017. Why approximate when you can get the exact? optimal targeted viral marketing at scale. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 1–9.

[25] Xiaodong Liu, Mo Li, Shanshan Li, Shaoliang Peng, Xiangke Liao, and Xiaopei Lu. 2013. IMGPU: GPU-accelerated influence maximization in large-scale social networks. *IEEE Transactions on Parallel and distributed Systems* 25, 1 (2013), 136–145.

[26] J. Marro and R. Dickman. 2005. *Nonequilibrium Phase Transitions in Lattice Models*. Cambridge University Press. https://books.google.com/books?id=80YF69jbczYC

[27] Michael Molloy and Bruce Reed. 1995. A critical point for random graphs with a given degree sequence. *Random structures & algorithms* 6, 2-3 (1995), 161–180.

[28] Cristopher Moore and Mark EJ Newman. 2000. Epidemics and percolation in small-world networks. *Physical Review E* 61, 5 (2000), 5678.

[29] Flaviano Morone and Hernán A Makse. 2015. Influence maximization in complex networks through optimal percolation. *Nature* 524, 7563 (2015), 65.

[30] Mark EJ Newman. 2001. Clustering and preferential attachment in growing networks. *Physical review E* 64, 2 (2001), 025102.

[31] 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 *Proceedings of the 2016 International Conference on Management of Data*. ACM, 695–710.

[32] Romualdo Pastor-Satorras and Alessandro Vespignani. 2001. Epidemic dynamics and endemic states in complex networks. *Physical Review E* 63, 6 (2001), 066117.

[33] Youze Tang, Yanchen Shi, and Xiaokui Xiao. 2015. Influence maximization in near-linear time: A martingale approach. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 1539–1554.

[34] Youze Tang, Xiaokui Xiao, and Yanchen Shi. 2014. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 75–86.

[35] Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of âĂŸsmall-worldâĂŹnetworks. *nature* 393, 6684 (1998), 440.