ASIM: A Scalable Algorithm for Influence Maximization under the Independent Cascade Model

Sainyam Galhotra, Akhil Arora, Srinivas Virinchi, and Shourya Roy
Xerox Research Centre India, Bangalore
{sainyam.galhotra, akhil.arora, srinivas.virinchi, shourya.roy}@xerox.com

ABSTRACT

The steady growth of graph data from social networks has resulted in wide-spread research in finding solutions to the influence maximization problem. Although, TIM [4] is one of the fastest existing algorithms, it cannot be deemed *scalable* owing to its exorbitantly high memory footprint. In this paper, we address the scalability aspect – *memory consumption* and *running time* of the influence maximization problem. We propose *ASIM*, a scalable algorithm capable of running within practical compute times on commodity hardware. Empirically, *ASIM* is $6-8$ times faster when compared to *CELF++* [1] with similar memory consumption, while its memory footprint is ≈ 200 times smaller when compared to TIM.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications – *Data Mining*

Keywords: Social network; Influence maximization; Viral marketing; Greedy algorithm; Running time; Scalability

1. INTRODUCTION

Social networks have become pervasive owing to the exponential growth in their popularity. The scale at which these networks operate today is humongous – *Facebook*, *Twitter* etc. have over billions of nodes and trillions of edges. This wide-spread reach paves the way for a host of applications with huge impact. The influence maximization problem with applications in *viral marketing* is one such example. More formally, given a network $G(V, E)$; $|V| = n$, $|E| = m$, with edge weights $(p(e) \mid e \in E)$ denoting the pair-wise influence probabilities, and a budget constraint k the objective of influence maximization is to select a set S of k seed-nodes ($|S| = k$) with the ability to maximize the spread of information over this network.

Kempe et al. [3] in their seminal work proved that finding an optimal solution for the influence maximization problem is NP-Hard and were the first to prove that a simple greedy algorithm can provide the best approximation guarantees in polynomial time. They incorporated the use of two fundamental diffusion models – Independent Cascade (IC) and Linear Threshold (LT) for information propagation. However, the algorithm proposed by them had two sources of inefficiency. The first is that it took $O(kmn)$ time to produce a solution, while the second one is that it requires an additional factor of a large number of Monte Carlo (MC) simulations ($\approx 10K$) to obtain the expected value of the *spread*.

Considerable amount of work has been done to cater to the first aspect – optimizing the running time of this greedy algorithm, with *CELF++* [1] being the most efficient of all, but there hasn't been much work in improving the second. More recently, Tang et al. [4] have come up with an algorithm (TIM)[1] that runs in $O((k+l)(m+n)\log n/\epsilon^2)$ expected time and produces a $(1-\frac{1}{e}-\epsilon)$-approximate solution, where ϵ is a constant, with probability as high as $1-n^{-l}$. While this is the fastest known algorithm for influence maximization it cannot be termed *scalable* as it has a high memory footprint. The worst case space complexity of TIM is $O(n^2 \log \binom{n}{k}/\epsilon^2)$, which can be very high for small values of ϵ. For example, the memory footprint of TIM can be as high as 100 GB for a graph with a million nodes and close to 3 million edges (Details in Section 3). This huge requirement is tough to be honored by commodity hardware.

In this paper, we propose an efficient algorithm *ASIM* which provides the best tradeoff between *memory-consumption* and *running-time* and is capable of handling real-world large scale networks on moderately sized machines. We argue that our algorithm can be efficiently parallelized since each MC simulation is independent of the other. Moreover, a single iteration of *ASIM* takes $O(kd(m+n))$ (Details in Section 2) which is faster than TIM thus effectively it can perform better on overall time[2] while keeping the memory footprint $150-200$ times smaller when compared to the latter.

2. ALGORITHM

In this section, we present our algorithm *ASIM* to address the influence maximization problem. This algorithm takes a graph G, number of seeds k and depth value d as input and returns a set of seed nodes S. It is intuitive that the probability of a node v to get activated by a seed node u, assuming the standard diffusion models as proposed in [3], is dependent upon the number of simple paths from u to v.

[1]For notation and details please refer [4].

[2]Considering the ease of availability of multiple processing units (cores) in a single machine when compared to large amount of memory (RAM). Details in Section 3.

WWW 2015 Companion, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3473-0/15/05.
http://dx.doi.org/10.1145/2740908.2742725.

Algorithm 1 Seed Selection using ASIM

Input: Graph $G = (V, E)$, #seeds ($k = |S|$) and depth (d)
Output: Seed set S

```
1: S, C ← ∅
2: for i = 1 to k do
3:    max, maxId ← 0
4:    Score ← AssignScore(G, C, d)
5:    for each u ∈ V do
6:       if Score[u] > max then
7:          max ← score; maxId ← u
8:       end if
9:    end for
10:   S ← S ∪ {maxId}; C ← C ∪ F(maxId)
11: end for
```

Goyal et al. [2] used a similar idea to propose an algorithm for the linear threshold (LT) model. *ASIM* exploits this aspect of information propagation by assigning a score to each node (u) of the graph. $Score[u]$ ($\forall u \in V$) is defined as the weighted sum of the number of simple paths of length at most depth[3] ($d \mid d \leq \mathcal{D}$) starting from u. The weight for each path is defined as the product of probabilities $p(e)$ of the edges composing that path. The score assigned to a node u tries to mimic closely the expected value of the spread when u is chosen as a seed node.

At each iteration, our algorithm selects the node with the maximum score as the seed node and updates the set C with the nodes activated by this seed. At any given iteration, the set C contains all the nodes activated $\mathcal{F}(s)$ by the seed nodes $s \in S$. In line 6 of the algorithm, we assign a score to each node v of the graph G as explained above. In order to ensure that the set of nodes activated by each selected seed node are disjoint, *AssignScore* neglects the contribution of all the paths containing any previously activated node $c \in C$ in the score calculation for the subsequent iterations.

Paths of length d from a node u can be calculated as the sum of all paths of length $d-1$ from its neighbors. Owing to this observation, the time taken by the algorithm to assign scores to each node of the graph is $O(d(m+n))$ because for each iteration over d, it looks at the adjacency list of each node to compute the updated score. Hence the overall time complexity of *ASIM* is $O(kd(m+n))$.

3. EXPERIMENTS

All the simulations were done using the Boost graph library (http://www.boost.org) in C++ on an Intel(R) Xeon(R) 32-core machine with 2.4 GHz CPU and 100GB RAM running Linux Ubuntu 12.04. We adopt the C++ implementation of CELF++ and TIM made available by the respective authors. We present results on real graphs, taken from the arXiv (http://www.arxiv.org) and SNAP database (snap.stanford.edu/data/). All the experiments were conducted on the following datasets: 1) NetHEPT (15K nodes and 62K edges), 2) HepPh (12K nodes and 118K edges), 3) DBLP (317K nodes and 1M edges) and 4) YouTube (1M nodes and 3M edges). These datasets are undirected, however they were made directed by taking for each edge the arcs in both the directions. We use the IC model [3] with each edge being uniformly assigned a probability of 0.1. As a conventional practice, the spread is calculated as an average over $10K$ MC simulations[4]. We don't compare *ASIM* with SIMPATH, as we present results on the IC model only[5] while SIMPATH is specifically for the LT Model.

[3]Depth determines the accuracy of the score assignment. $\mathcal{D}$ is the diameter of the graph.

Dataset	Running Time (min)			Memory (MB)		
	CELF++	ASIM	Gain	CELF++	ASIM	Gain
NetHEPT	5352.25	648.33	**8.25x**	23.26	10.5195	2.2x
HepPh	9746.74	1355	**7.2x**	24.60	14.0391	1.75x
DBLP	88216.69	13166.67	**6.7x**	138.19	159.09	0.87x

Table 1: Comparison between *CELF++* and *ASIM*, $k = 100$ and $d = 1$.

Dataset	Running Time (min)			Memory (MB)		
	TIM	ASIM	Gain	TIM	ASIM	Gain
DBLP	783.1	6500	0.12x	35234.75	159.09	**221x**
YouTube	NA	19666.67	∞	NA	553.94	∞

Table 2: Comparison between *TIM* and *ASIM*, $k = 50$, $d = 1$ and $\epsilon = 0.1$.

Table 1 compares the running time and memory consumption of *ASIM* with *CELF++*. The results for the YouTube dataset using *CELF++* have not been reported since it required > 200 hours for 400 iterations and was still not completed. Table 2 shows a similar comparison of *ASIM* with *TIM*. Although, TIM is ≈ 10 times faster when compared to *ASIM* for the DBLP dataset, we argue that this difference can be easily mitigated (as discussed in Section 1) by running *ASIM* in parallel on 10 cores while ensuring the memory gain to be the same. The memory consumed by *TIM* was $>$ 100GB for $\epsilon = 0.1$, $k = 50$ on YouTube, thus we could not report the results for the same.

4. CONCLUSIONS

In this paper, we addressed the problem of influence maximization in social networks from a scalability standpoint. Since majority of the influence maximization algorithms are not scalable ([3, 4] and the references therein), there are efficiency concerns in real-world scenarios with huge graphs. Consequently, we designed an efficient algorithm that uses graph topology and connectivity, and runs in linear time. Moreover, the space complexity of our algorithm is linear which is orders of magnitude better when compared to the current state of the art. Our empirical studies on real world social network datasets showed that our algorithm scales well, portraying significant improvements in terms of both running time and memory consumption, and is practical for large real graphs.

5. REFERENCES

[1] 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.

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

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

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

[4]These instances were run in parallel on 32-cores for CELF++ and ASIM. However for a fair comparison with TIM we report the total time taken.

[5]Owing to the space constraints