Influence Maximization: A Time-Space Efficient Algorithm

IOP Conf. Series: Materials Science and Engineering **533** (2019) 012048 doi:10.1088/1757-899X/533/1/012048

Influence Maximization: A Time-Space Efficient Algorithm

Ganming Xia

Beijing University of Posts and Telecommunications, Beijing 100876, China

Gnmingxia@gmail.com

Abstract. Influence maximization is the problem of finding a small subset of nodes (seed nodes) in a social network that could maximize the spread the influence. The algorithm problem of computing influence estimation and influence maximization have been extensively studied for decades. In this paper, we studied the reverse influence sampling method proposed by Byogs et al. in 2014, and some other improved algorithms. We proposed a new algorithm TESA (Time-Space Efficient Algorithm). On large social networks, TESA saves more space than existing algorithms and runs faster. We used data sets tested in the literature to evaluate TESA experimentally and showed that it was superior to the most advanced solutions in terms of running time and memory footprint.

1. Introduction

Recently, many large social networking sites, such as Facebook, Friendster, etc., have achieved great success, mainly because they are able to connect people, and the results are very good. They also link small, separate social networks. At the same time, they are becoming a huge communication and marketing platform, where a lot of information can be spread to a very wide range in a very short time. In this paper, we present our work to address one of the challenges of effectively identifying influential individuals in large social networks.

Generally speaking, social networks are relatively large, so operation time and operation space are two main factors to be considered in solving the problem of maximum influence. Kempe et al. showed that the influence maximization problem is NP-hard [1], and Chen et al. showed for the influence estimation problem that computing the exact influence of a single seed is #P-hard [2]. Moreover, as Feige proved in 1998, the problem is hard to approximate to anything better than $1 - (1 - 1/s)^s$ of the optimum for a seed set of size s [3].

In order to capture the propagation of influence spread, Kempe et al. [1] introduced the Independent Cascade (IC) model [4]: an independent random variable is assigned to each directed edge (u, v); the variable reflects the level of influence from u to v. Starting from a vertex, in each step, information spreads to vertex neighbors with the probability equal to level of influence of the vertex over each neighbors. Kempe et al. Showed that IM on the IC model encodes the classic maximum coverage problem and therefore is NP-hard [5]. Chen et al. [6] shwoed for the IE problem that computing the exact influence of a single seed is #P-hard. Kempe et al. [1] showed that IM on the IC model is monotone and submodular, and therefore, a Greedy algorithm produces provably-good quality solutions. More precisely, the influence of the approximate Greedy solution with given number of seeds is $(1 - 1/ e - \epsilon)$ of the optimal solution, for any $\epsilon > 0$ [7]. IC becomes a standard model of influence spread, and we are using it for our algorithm.

According to the research results of Kempe et al., other improved algorithms have been published successively in this theory, such as [2, 8, 9, 10, 11, 12]. However, there are still many problems in dealing with large-scale social networks.

In 2014, Brogs et al. [13] proposed a different approach: Reverse Influence Sampling (RIS) method. The main idea of RIS is to select a random node from the existing nodes and represent it with v, and then determine how many nodes can affect v nodes. If a vertex u usually affects different randomly selected vertices, then u is a good candidate for the most influential vertices. This can be done by simulating the influence process using the IC model in the graph, and the edge direction in the graph is reversed. RIS is a fast algorithm for IM, obtaining the near-optimal approximation factor of $(1 - 1/e - \epsilon)$, for any $\epsilon > 0$, in time $O((m + n)k\epsilon^{-2} \log(n))$, where n is the number of vertices, m is the number of edges, and k is the number of seeds. Borgs et al. showed RIS runtime to be optimal (up to a logarithmic factor) [5]. Although this algorithm solves the problem of maximizing influence well, there are still some shortcomings. In the common IM algorithm and its improved algorithm, there are two main problems: one is not running fast enough, and the other is a large memory footprint.

In this paper, we deeply analyzed the space saving algorithm proposed by Popova et al. [5]. The algorithm has significant effect on space saving, but the running time is long and many useless subgraphs are generated. We propose a TESA (time-space efficient algorithm) based on this algorithm, which allows to scale-up computing of IM and IE to massive graphs with billions of edges and faster. We compared the time space performance of TESA with state-of-the-art algorithms. TESA takes less time and less space for the same graph.

2. Preliminaries

We describe in this section some notations and definition.

2.1. Notations

Let $G = (V, E, p)$ be a directed graph, and vertex is set as $| V | = n$, edge Set $| E | = m$, and $p: E \rightarrow [0,1]$ is the existence of the probability of the edge. S is the seed set.$\sigma(S)$ is the influence spread of a seed set S under the Independent Cascade (IC) model [14].

2.2. IM and IE problems

Influence Estimation Problem (IE). Given a graph $G = (V ,E,p)$ and a seed set $S \subseteq V$, compute the influence spread $\sigma (S)$ of S.

Influence Maximization Problem (IM). Given a graph $G = (V ,E,p)$ and an integer k, find a seed set $S \subseteq V$ of size k that maximizes $\sigma (S)$.

2.3. Social networks

A social network represents a social structure made up of individuals or organizations and their relations (friendship, co-authorship of scientific papers, co-appearance in a movie, and following-followers et al.). Social networks are described busing a Graph $G(V, E)$ where V is the set of nodes representing the individuals or organizations and E is the set of edges representing the social ties.

3. Proposed Data Structures

Although the existing greedy algorithm has a good result for solving IM problems, it still needs a great cost for large networks. We proposed a new algorithm on the existing RIS algorithm and its improvement algorithm, which can well solve the problem of running space and time.

3.1. RIS algorithm

RIS method is the generation of subgraphs by arbitrary selected nodes in the graph, which is called the influencing set. If a vertex often appears in the influencing set, then the vertex is the candidate for the optimal influence in the graph. Moreover, RIS is shown in [13] to be runtime-optimal (up to a logarithmic) with respect to network size.

The weight of the hypergraph is defined as the number of graph edges "touched" by RIS [5]. RIS randomly selects the edges to follow with a given edge probability p. During the search, each side that hits the node is counted as a "touch" and helps calculate the weight. Note that the edge is considered "touched" whether the search selects it or not. An approximation of how large the weight should be to ensure an optimal solution is defined in [13].

3.2. The Time Space Efficient Algorithm

In this part, we mainly introduce the algorithm, TESA. Based on the existing RIS algorithm and its improved algorithm, we proposed an improved algorithm to make the algorithm more efficient in time and space. We show the pseudo-code of the algorithm in algorithm 1, followed by a detailed description of the main features of the algorithm.The main idea of TESA algorithm is that we classify all the nodes in the graph by using the principle of 80/20. In the real world, the most influential people are often the minority. 20 percent have more influence than 80 percent. A node with a larger degree indicates that the node has a greater influence in the graph. We count all the nodes in the graph, select 20 percent of the nodes before the degree to form a candidate set. Then, RIS method was used to select any node in the candidate set to generate the subgraph. Meanwhile, we adopted

Algorithm 3 TESA
Input: directed graph G, precision $\epsilon \in (0,1)$, number of seeds k
Output: seeds set $S \subseteq V$ of size k, spread $\sigma(S)$

$c \leftarrow 4(1+\epsilon)(1+1/k)$
$R \leftarrow cmk\epsilon^{-2}\log(n)$
$H \leftarrow$ BuildHypergraph(R)
$Average \leftarrow$ AverageDegree(G)
return GetSeeds(H)
procedure BuildHypergraph(R)
$sk\_degree \leftarrow 0$
$sk\_num \leftarrow 0$
$averageDegree \leftarrow 0$
    while $H\_weight < R$ **do**
        $v \leftarrow$ random vertex of 20% maximum degree of G^T
        $sk \leftarrow$ DFS in G^T starting from v
        $sk\_num = sk\_num + 1$
        for each $u \in sk$ **do**
            $count(u) \leftarrow count(u) + 1$
            $sk\_degree \leftarrow sk\_degree + G^T.outdegree(u)$
            $averageDegree \leftarrow$ averageDegree + $G^T.outdegree(u)$
        if $sk\_cardinality > 2$ and $averageDegree > average$ **then**
            append sk to hypergraph H
        $H\_weight \leftarrow H\_weight + sk\_degree$
    return H
procedure GetSeeds(H)
    $S \leftarrow \emptyset$
    $\sigma(S) \leftarrow 0$
    for $i = 1,\ldots,k$ **do**
        $vi \leftarrow \text{argmax}_v \{count(v)\}$
        $S.insert(v_i)$
        $\sigma(S) \leftarrow \sigma(S) + count(v_i) * n/sk\_num$
        scan H
        if $vi \in sk_j$ **then**
            for each $u \in sk_j$ **do**
                $count(u) \leftarrow count(u) - 1$

```
33:             count (v_i ) ← 0
34:         output S, σ (S)
35:   podedure AverageDegree(G)
36:         total ← 0
37:         degree ← G.outdegrees()
38:         i ← 0
39:         while degree.hasNext() do
40:             total ← total + degree.next()
41:             i ← i + 1
42:             if i == n – 1 then
43:                 break
44:         return total / n
```

the DFS (Depth First Search) algorithm in the process of generating the subgraph. But we don't see all the nodes as the same. To consider a general situation in real world networks, every node is different. For a specific spreading probability *p,* and *p* is usually very small in real world networks. When *p* is small, the multi-hop neighbors of *v* can be ignored, and only the nearest neighbors are counted toward the degree [7]. Therefore, when we use DFS to calculate the subgraph, we only calculate the candidate node three-hop neighbors. This is more in line with the actual situation, and can save time and improve the calculation efficiency when calculating the subgraph.

The major differences of TESA algorithm comparing to RIS and other RIS-based algorithms.

(1) We use webgraph, a highly efficient, and actively maintained graph compression framework [15].

(2) We use the Pareto principle, where influential people are often only 20 percent of the total population. So we used Pareto principleto constructed a candidate set.

(3) We use DFS algorithm to calculate the influence spread of the seed node, and we only calculate three-hop neighbors, which greatly accelerates the calculation speed and saves space.

4. Experimental Results

We have conducted extensive experiments on several real-world graphics and have tested our IM and IE solutions. At the same time, for the sake of simplicity, we only include the most interesting and problem-solving results in IM and its analysis. Due to the space constraints, we only present results for three real world graphs. The datasets are available from the Laboratory for Web Algorithmics [16, 17] (http://law.di.unimi.it/datasets.php).

Although the size of the network we consider is in the medium range, each node in the network can be sampled ultiple times, and the number of edges used by all our algorithms is huge.

Table 1. Datasets ordered by m.

Dataset	N	M
UK100K	100,000	3,050,615
CNR2000	325,557	3,216,152

Equipment.The experiments were conducted on a desktop computer with processor 4.0 GHz Intel Core i7 (8-core), RAM 16GB 2400MHz DDR4.

Parameters.The parameters we used in the experiment are as follows. *k* is the seed size. *p* is the probability of edge existence. In these tests, *k* takes values of 5, 10, 25, and *p* takes values of 0.1, 0.01, 0.001.

Arora et al. [18] comprehensively evaluated and tested the most advanced IM algorithms. Arora et al. tested several well-known IM algorithms, including CELF++, TIM, IRIE, and PMC, and compared their performance, runtime, and memory consumption. The algorithms tested by Arora et al. were released before May 2016, so they did not contain some of the most interesting and promising IM

algorithms. We decide to test our TESA vs new IM algorithm, NoSingles, and perform a comparative analysis of the results.

Fig.1 shows testing results when TESA was compared to NoSingles [5] while processing IM for graph uk100, a medium size web graph with 100k vertices and 3M edges. Both algorithm use RIS method, with the lower bound on the hypergraph weight as defined in Theorem 4.1 in [13]. Parameters used by both algorithms were identical throughout testing: Brogs et al. number of calculated seeds varied: k = 5, 10, 25, and the probability of edge existence was taken as p = 0.1,0.01,0.001. This ensures that the IM solutions computed by TESA and NoSingles have the same approximation guarantees.

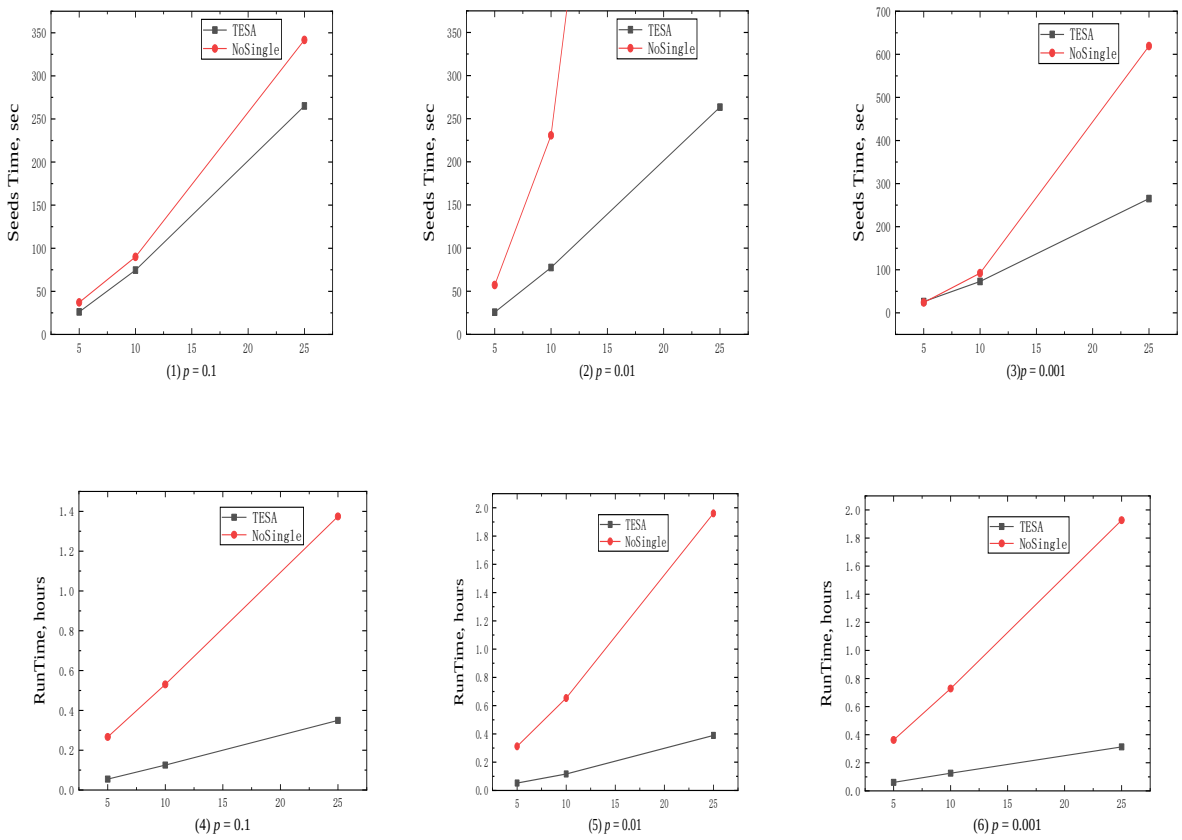

Figure 1. TESA vs NoSingles, varying k in uk100

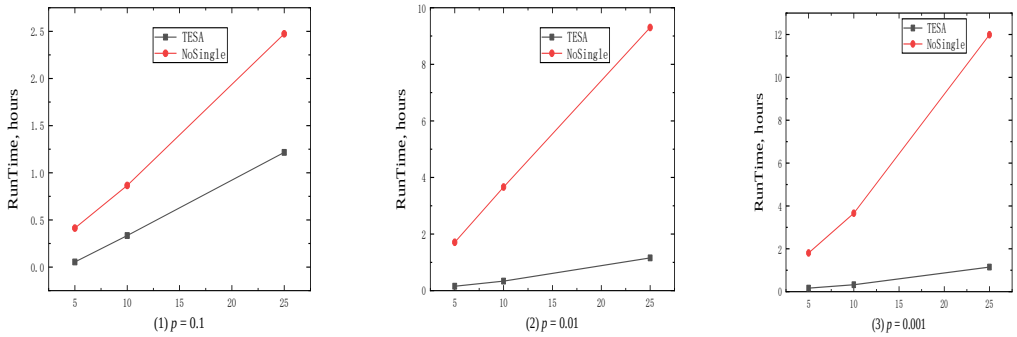

Figure 2 . TESA vs NoSingles, varying k in cnr2000Charts in Fig. 1 and Fig.2 demonstrate that:

(1) Space consumed by TESA is smaller than NoSingles, for all tested k.

(2) On two datasets, time taken by the whole processing (Total Time), is lower for TESA, for all tested *k*, and
(3) Time taken by the seed calculation (Seeds Time), is lower for TESA.

RunTime. The running tie is mainly composed of the following two parts, one is the time required to build the hypergraph, and the other is to calculate the seeds set. Fig. 1 (4), (5), and (6) shows that TESA spends overall less time for calculating IM Solutions, for all *k*. On the other hand, we have adopted a new method to calculate the hypergraph, because in the actual social network, the probability of information propagation is relatively low. When the information is transmitted multiple times, the probability of being accepted is very low [7], so we only compute the hypergraph obtained by the node in the finite number of propagation. TESA reduces the number of nodes that need to be calculated and also reduce memory.

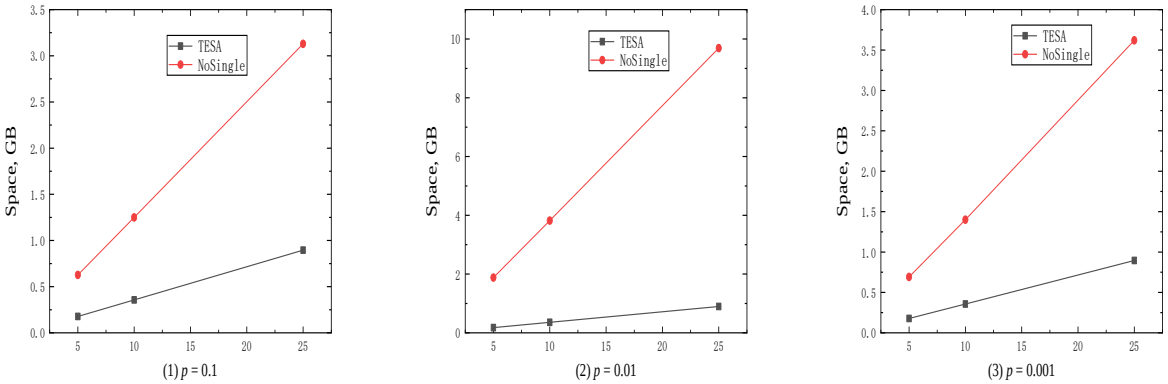

Figure 3. TESA vs NoSingles, varying k in uk100

Seeds Time. Fig. 1 (1), (2), and (3) shows that the time for computing seeds is much lower for TESA than for NoSingles. The main reason is that the number of hypergraph nodes generated by the TESA is relatively small, and have a good influence. Seeds time consists of two parts, one is to calculate the most influential node, the calculation time is the same, the second part is to update other nodes connected with the most influential node in the social network. There are fewer nodes in the hypergraph generated by TESA, so it takes less time to update the hypergraph, which makes the seeds time of TESA less than NoSingles.

Space. Space consumption is shown in Fig. 3 (1), (2). In our testing, we assigned the same p for all graph edg-es. From the Fig. 3 we can see that the TESA uses less memory than the NoSingles algorithm. The main reason is that we only calculate the finite neighbors to form the subgraph. Reduce memory while ensuring a large sprea-d range.

5. Conclusions And Future Research

We have implemented a new algorithm, which is faster and takes less memory than the existing RIS algorithm and the improved algorithm based on RIS. We tested th-e performance of the new alrgorithm on a larger graph and compared the test results. Without compromising t-heoretical guarantees, we significantly reduced the run-ning time an required memory, enabling millions of edge graphics to run on regular desktop computers. The main research in the future is to speed up the algorithm while greatly reducing the memory footprint of the data.

References

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

[2] W. Chen, Y. Wang, and S. Yang. Efcient influence maximization in social networks. In KDD, pages 199–208, 2009.
[3] U. Feige. A threshold of ln n for approximating set cover. Journal of the ACM (JACM), 45(4):634–652, 1998.
[4] J. Goldenberg, B. Libai, and E. Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. Marketing Letters, 12(3):211–223, 2001.
[5] Popova D, Ohsaka N, Kawarabayashi K, et al. NoSingles: a space-efficient algorithm for influence maximization[C]//Proceedings of the 30th International Conference on Scientific and Statistical Database Management. ACM, 2018: 18.
[6] W. Chen, C. Wang, and Y. Wang. 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, pages 1029–1038, 2010.
[7] Wang X, Zhang X, Zhao C, et al. Maximizing the spread of influence via generalized degree discount[J]. PloS one, 2016, 11(10): e0164393.
[8] K. Huang, S. Wang, G. S. Bevilacqua, X. Xiao, and L. V. S. Lakshmanan. Revisiting the stop-and-stare algorithms for influence maximization. PVLDB, 10:913–924, 2017.
[9] H. T. Nguyen, M. T. Thai, and T. N. Dinh. A billion-scale approximation algorithm for maximizing beneft in viral marketing. IEEE/ACM Trans. Netw., 25(4):2419–2429, Aug. 2017.
[10] N. Ohsaka, T. Akiba, Y. Yoshida, and K.-i. Kawarabayashi. Fast and accurate influence maximization on large networks with pruned monte-carlo simulations. In AAAI, pages 138–144, 2014.
[11] Y. Tang, Y. Shi, and X. Xiao. Influence maximization in near-linear time: A martingale approach. SIGMOD ’15, New York, NY, USA.
[12] Y. Tang, X. Xiao, and Y. Shi. Influence maximization: Near-optimal time complexity meets practical efciency. SIGMOD ’14, New York, NY, USA.
[13] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms, pages 946–957, 2014.
[14] Popova D, Khot A, Thomo A. Data Structures for Efficient Computation of Influence Maximization and Influence Estimation[J]. 2018.
[15] P. Boldi and S. Vigna. The webgraph framework i: compression techniques. In WWW, 2004.
[16] Borgs C, Brautbar M, Chayes J, et al. Maximizing social influence in nearly optimal time[C]//Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, 2014: 946-957.
[17] P. Boldi and S. Vigna. The WebGraph framework I: Compression techniques. In Proceedings of the 13th International Conference on World Wide Web, pages 595–602, 2004.
[18] Arora, S. Galhotra, and S. Ranu. Debunking the myths of influence maximization: An in-depth benchmarking study. In Proceedings of the 43rd ACM SIGMOD International Conference on Management of Data, pages 651–666, 2017.