Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 10, No. 1, 2022, 25-41.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Research paper

Optimizing CELF Algorithm for Influence Maximization Problem in Social Networks

Mohsen Taherinia[1], Mahdi Esmaeili[1*] and Behrouz Minaei-Bidgoli[2]

1. Department of Computer Engineering, Kashan Branch, Islamic Azad University, Kashan, Iran.
2. School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran.

Article Info

Article History:
Received 09 August 2021
Revised 22 September 2021
Accepted 02 November 2021

DOI:10.22044/JADM.2021.10964.2254

Keywords:
CELF Optimization, Influence Maximization, Social Networks Analysis, Greedy Algorithm, Diffusion Model.

**Corresponding author: m.esmaeili@iaukashan.ac.ir (M. Esmaeili).*

Abstract

The influence maximization problem in social networks aims to find a minimal set of individuals in order to produce the highest influence on the other individuals in the network. In the last two decades, a lot of algorithms have been proposed to solve the time efficiency and effectiveness challenges of this NP-Hard problem. Undoubtedly, the CELF algorithm (besides the naive greedy algorithm) has the highest effectiveness among them. Of course, the CELF algorithm is faster than the naive greedy algorithm (about 700 times). This superiority has led many researchers to make extensive use of the CELF algorithm in their innovative approaches.
However, the main drawback of the CELF algorithm is the very long running time of its first iteration since it has to estimate the influence spread for all nodes by the expensive Monte-Carlo simulations, similar to the naive greedy algorithm. In this paper, a heuristic approach is proposed, namely optimized-CELF algorithm, in order to improve this drawback of the CELF algorithm by avoiding the unnecessary Monte-Carlo simulations. It is found that the proposed algorithm reduces the CELF running time, and subsequently, improves the time efficiency of the other algorithms that have employed CELF as a base algorithm. The experimental results on the wide spectral of real datasets show that the optimized-CELF algorithm provides a better running time gain, about 88-99% and 56-98% for k=1 and k=50, respectively, compared to the CELF algorithm without missing effectiveness.

1. Introduction

In the last two decades, the world has witnessed the explosion of online social networks. Today, the social networks play a vital role in the dissemination of information among the people extensively and rapidly. The people within the social networks are influenced by each other via their relationships (friends, acquaintances, and colleagues). This social influence can change the people's thoughts, ideas, opinions, and behaviors [1]. Many companies and organizations have exploited this opportunity and have developed various applications such as marketing, advertising, and recommender systems. [2]. For example, consider a company that intends to promote a new product to the market with the lowest budget. One solution is to find a small set of influential individuals inside the social network to freely trial the product. It is expected that these influential individuals will recommend the new product to their close individuals in a cascading mechanism. Thus a new product will be promoted in the market through the power of “word of mouth” in viral marketing.

This problem, which is called the influence maximization problem, is a classic and hot topic in social network analysis that has been attracted much attention from the researchers. The goal is to find a set of k nodes so that if the information propagation process starts from this initial set, then more individuals receive the information eventually, and the influence spread will be maximized throughout the network. The influence maximization problem has been introduced by Domingos and Richardson in viral marketing [3]. Kemp *et al*. have formulated this problem and proved that obtaining an optimal solution of the influence maximization problem is *#NP-hard* under the Linear Threshold (LT) and Independent Cascade (IC) diffusion models [4]. They presented a greedy hill-climbing algorithm with *O(knmR)* time complexity that provided a solution with 63% approximate of optimal solution (*n* and *m* are the number of nodes and edges in the network, *k* is the size of the target seed set, and *R* is the number of Monte-Carlo simulations). Their algorithm suffers from two major challenges. First, computing the exact influence spread of seed set *S (δ(S))* is *#P-hard* under the LT and IC models. This challenge is handled by estimating the influence spread using the Monte-Carlo (MC) simulations. Secondly, this algorithm is quadratic in the number of nodes, which makes it inefficient [1]. So far, many efforts have been made in order to improve the time efficiency dilemma of this greedy algorithm. One of the outstanding proposed approaches is Cost-Effective Lazy Forward (CELF) by Leskovec *et al*. [5], which has been used in many subsequent works. They exploited the submodularity property of the objective function to reduce the time complexity of the greedy algorithm. The CELF algorithm became up to 700 times faster than the greedy algorithm by avoiding the unnecessary evaluation of the influence spread of nodes. However, it still has to estimate the influence spread for all nodes in the first round of the algorithm by the expensive MC-simulation. This computational overhead causes the CELF algorithm to be inefficient.

Novelty:

This paper aims to tackle this time inefficiency of the CELF algorithm by proposing a heuristic approach. The main idea is to reduce the influence spread calculations in the first round by a pruning technique. Firstly, algorithm 2 constructs the Minimal Spanning Node (MSN) set composed of the core nodes of the network. Then we just estimate the influence spread of the MSN nodes instead of all network nodes using the MC-simulations. In this way, most of the graph nodes are eliminated from expensive MC simulations (96% on average practically). The influence spread of the remind nodes is easily calculated based on the obtained influence spread of the MSN set nodes according to (3). The practical results indicate that the optimized-CELF algorithm provides a better running time than the CELF algorithm, while produces the target seed set with the same quality. Since the CELF algorithm is orthogonal to the algorithms that optimize the estimating of the influence spread, the proposed algorithm can be combined with them in order to achieve a scalable algorithm.

The rest of this paper is organized as what follows. The related works on the influence maximization are reviewed in section 2. Section 3 describes the related definitions of the influence maximization. The optimized-CELF algorithm is introduced in section 4. Section 5 reports the experimental results. The performance evaluation of the optimized-CELF algorithm is discussed in section 6. Finally, section 7 concludes this paper with future works.

2. Literature Review

The influence maximization problem could be traced back to Domingos and Richardson [3]. It was formulated as a discrete optimization problem by kemp *et al*. [4]. They proved that the optimization of this problem was *#NP-hard*. Thus they developed a greedy approximation framework that provided a near-optimal solution with a theoretical guarantee within $\left(1-1/e\right)$. However, it suffers from the time inefficiency issue due to the huge computational overhead of MC-simulations. Leskovec *et al*. have proposed the Cost-Effective Lazy Forward (CELF) in order to improve the time inefficiency issue of the naïve greedy algorithm [5]. It utilized the sub-modularity property of the influence spread function to reduce the number of unnecessary MC-simulations. The practical results demonstrated that the CELF optimization provided the same performance as the naïve greedy algorithm, 700 times faster. The CELF optimization was further improved to the CELF++ algorithm [6] by Goyal *et al*. The CELF++ strategy estimated the influence spread for two successive iterations of the naïve greedy algorithm. It was about 35-55% faster than CELF due to reducing the number of spread estimation calls. Chen *et al*. have proposed the NewGreedy algorithm [7], an improved version of the naïve greedy algorithm. It reused the previous results of the MC-simulations to compute the influence spread for all candidate nodes in the same iteration. The StaticGreedy approach, presented by Cheng *et al*. [8], took a few MC snapshots at first. Next, *k* nodes with the highest marginal gain on all sampled snapshots were picked as the seed set nodes. Due to the inevitable time complexity of the StaticGreedy algorithm in the worst case, a pruning method was applied to decrease its running time in the staticGreedyDU algorithm [8]. UBLF [9] has provided an upper bound of the influence spread function using the matrices, introduced by Zhou *et al*. UBLF reduced the MC-simulations calls, more than 95%. Tang *et al*. [10] have proposed a Two-Phase algorithm (TIM). In its first phase, a lower bound was computed on the influence function in order to estimate the parameter φ. The second phase selected φ numbers of reverse reachability from the graph. Finally, k nodes were selected that covered the maximum number of reverse reachability. Also they proposed an improved version of TIM, namely TIM+, in order to speed up the parameter estimation procedure. IMM [11] was a two-phase method similar to TIM, which was proposed by Tang *et al*. The IMM approach utilized advanced estimation techniques such as the martingales method. High memory consumption was one of the most critical challenges of TIM, TIM+, and IMM.

Unfortunately, these improved versions of the naïve greedy algorithm were still inefficient due to huge MC-simulations, despite providing the theoretical guarantee on the influence spread [12]. Hence, a wide spectral of heuristic algorithms had been proposed to trade-off between the influence spread and the running time. The most intuitive idea was to select the nodes based on their degree. Degree Centrality, Betweenness Centrality, Closeness Centrality, Eigenvector Centrality [13], PageRank [14], and Hits [15] were the most commonly heuristic approaches based on the node degree. Since these approaches did not consider the overlap of the influence between the different seeds, the overestimation challenge would occur. The Degree-Discount algorithm was proposed by Chen *et al*. [7] to take into account the overlap of the nodes' influence. After selecting one node as the seed node, the influence scores of its neighbors are reduced by a factor. Goyal *et al*. have proposed the SimPath algorithm [16], which counts all simple paths starting from each node within the seed set to estimate the influence spread. The SimPath algorithm exploits the CELF optimization [5] for more efficiency. The main idea of SP1M [17], which was introduced by Kimura *et al*., was the node activation by considering the shortest path from node u to node v. Chen *et al*. have developed the MIA and PMIA models inspired by the SP1M idea [18]. The MIA model limited the propagation area into a local tree structure. The PMIA model modified the in-arborescence influence after selecting a seed node to prevent blocking the influence of the subsequent seeds by the current seed. IPA was a path-based algorithm by Kim *et al*., which employed a parallelization method to estimate the influence spread for a more efficiently [19]. Chen *et al*. have developed the LDAG framework [20]. It built the local DAGs for each node to estimate the influence spread accurately and efficiently. Narayanam *et al*. have introduced the SPIN algorithm [21], in which the information propagation is conducted similar to what happens in a coalition alliance. The SPIN algorithm ranked the nodes according to their Shapley value, and selected the *top-k* nodes as the target seed set. IRIE, which was developed by Jung *et al*., was a robust extension of the PageRank algorithm [22]. In order to avoid the influence overlaps occurrence, an influence estimation method was suggested to compute and subtract the extra influence of a seed. Group-PR was a generalization of PageRank based on a greedy framework, proposed by Liu *et al*. [23]. This model employed a collection of nodes instead of a single node to estimate the influence spread. The main idea of IMRank, which was presented by Cheng *et al*., was to discover the self-consistent ranking of any initial ranking [24]. Then IMRank was selected *top-k* nodes as the seed set nodes. Ohsaka *et al*. have proposed a snapshot-based sampling approach, namely the Pruned-MC [25]. Its efficiency and effectiveness were improved by exploiting an index structure on snapshots along with the pruning technique. The main idea of EaSyIM, which was developed by Galhotra *et al*., was to count the simple paths to compute the influence spread of each node [26]. This model employed the IRIE algorithm in order to compute the global influence. Although these heuristic approaches could improve the time efficiency of the naïve greedy algorithm, they cannot provide any worst-case bound, and are theoretically weak.

Increasing the size of the online social has bolded the scalability issue of the IM problem more than past [27]. In order to tackle this issue, many researchers have exploited the community structures as a successful strategy [28]. Wang *et al*. have used the community structures to improve the scalability of the influence maximization problem, the first time [29]. They invented the Community Greedy Algorithm (CGA), in which the graph was partitioned into small communities. CGA detected the appropriate community by dynamic programming in order to select the seed set nodes.

Table 1. Specification of outstanding influence maximization algorithms.

Algorithm	Authors	Yea	Diffusion model	Time complexity	CELF-based	Approximation
Greedy [4]	Kempe et al.	200	IC, LT	$O(krnm)$		$1-1/e-\varepsilon(r)$
CELF [5]	Leskovec et al.	200	IC, LT	$O(krnm)$	✓	$1-1/e-\varepsilon(r)$
SP1M [17]	Kimura et al.	200	IC	$O(knm)$		$1-1/e^{\alpha}$
NewGreedy [7]	Chen et al.	200	IC	$O(krm)$		$1-1/e-\varepsilon(r)$
Deg-Dis [7]	Chen et al.	200	IC, WC	$O(k\log n+m)$		Not guaranteed
CGA [29]	Wang et al.	201	IC	$O(E+(Z-M)NT_P+MKT_P+K\|C_P\|T_P)$		---
MIA/PMIA [18]	Chen et al.	201	IC	$O(nt_{i\theta}+kn_{0\theta}\ n_{i\theta}\ (n_{i\theta}+\log n))$		$1-1/e^{\alpha}$
LDAG [20]	Chen et al.	201	LT	$O(nt_{\theta}+kn_{\theta}\ m_{\theta}\ (m_{\theta}+\log n))$		Not guaranteed
CELF++ [6]	Goyal et al.	201	IC, LT	$O(krnm)$	✓	Not guaranteed
SimPath [16]	Goyal et al.	201	LT	$O(klnP_{\theta})$	✓	Not guaranteed
SPIN [21]	Narayanam et al.	201	LT	$O(tr(n+m)+n\log n+kn)$		Not guaranteed
IRIE [22]	Jung et al.	201	IC-N	$O(k(n_{o\theta}\ k+\ m))$		Not guaranteed
staticGreedy [8]	Cheng et al.	201	IC	$O(\varepsilon^{-2}kmn^{2}\log(n,k))$		$1-1/e-\varepsilon(r)$
IPA [19]	Kim et al.	201	IC	$O\left(c^{-1}nO_{v}\ n_{uv}+k2\left(c^{-1}O_{v}\ n_{uv}+(c-1)\right)\right)$	✓	Not guaranteed
TIM/TIM+ [10]	Tang et al.	201	IC, LT	$O(\varepsilon^{-2}k(m+n)\log(n))$		$1-1/e-\varepsilon(r)$
Group-PR [23]	Liu et al.	201	IC	$O(krm)(\text{Linear}),O(m+k^{2}n)(\text{Bound})$		Not guaranteed
IMRank [24]	Cheng et al.	201	IC	$O(nTd_{max}\log d_{max})$		Not guaranteed
Pruned-MC [25]	Ohsaka et al.	201	IC	$O(\varepsilon^{-2}kmn^{2}\log(n,k))$		$1-1/e-\varepsilon(r)$
UBLF [9]	Zhou et al.	201	IC, LT	$O(krnm)$		$1-1/e-\varepsilon(r)$
CINEMA [30]	Li et al.	201	C^2, C^3	$O(k'm'n'+\ kTRm')$		$---$
IMM [11]	Tang et al.	201	IC, LT	$O(\varepsilon^{-2}k(m+n)\log(n))$		$1-1/e-\varepsilon(r)$
INCIM [31]	Bozorgi et al	201	LT	$---$	✓	$---$
EaSyIM [26]	Galhotra et al.	201	IC, LT	$O(kD(n+m))$		Not guaranteed
HybridIM [32]	Ko et al.	201	WC	$---$	✓	$---$
IMPC [33]	Shang et al.	201	WC	$O(n+m+k(m+k_{max}))$	✓	$---$

CGA was not suitable for handling large social networks. Li *et al*. have developed an algorithm based on the community structure using the concept of conformity, called CINEMA [30]. CINEMA selects the target seed nodes inside the detected communities with the maximum marginal gain. In INCIM, which has been proposed by Bozorgi *et al*. [31], a new graph is constructed so that its nodes are the communities of the main graph. In the new graph, each node (community) could be considered as a diffusion module. INCIM employs the CELF strategy in order to reduce the time complexity. HybridIM is a combination framework of the path-based and community detection techniques, which has been developed by Ko *et al*. [32]. For each extracted community, a separate queue is created based on the nodes' marginal gain. Then the CELF algorithm [5] is applied to each queue directly. Shang *et al*. have presented a new model based on multi-neighbor, namely IMPC [33]. IMPC spreads the influence throughout the network in two separate phases: 1) multi-neighbor potential-based seeds expansion 2) intra-community influence propagation. IMPC employs the CELF strategy [5] in order to accelerate the proposed algorithm.

As mentioned earlier, many researchers have employed the naïve greedy algorithm [4] or CELF [5] (its optimized version). For example, the following approaches use the CELF strategy directly: CELF++ [6], SimPath [16], IPA [19], [34], INCIM [31], HybridIM [32], IMPC [33], and etc. As mentioned in the introduction section, the CELF algorithm meets the time efficiency dilemma in its first iteration. By improving this dilemma in this paper, other approaches that have exploited the CELF algorithm will be improved subsequently. Therefore, a significant progress will be achieved in the scalability of the influence maximization problem. Table 1 illustrates the outstanding influence maximization algorithms.

3. Preliminaries

This section formulates the influence maximization problem, and presents an overview of the naïve greedy and the CELF algorithms.

Table 2. Frequently used notations.

Notation	Description
$G(V,E)$	Social network G with node set V and edge set E
m, n	Number of nodes and edges in G, i.e. $\|V\|,\|E\|$
S	Initial seed set for diffusion process
k	Size of the seed set S, i.e., $\|S\|$
R	Number of Monte-Carlo simulations
M	Diffusion Model
$P_{u,v}$	Activation probability of node u on node v in the IC
$\delta_{G,M}(S)$	Expected influence spread of seed set S in a diffusion propagation process under M in G
W	Core node set of G. (MSN set)
$\Delta(v \mid S)$	Marginal influence spread (marginal gain) of node v upon seed set S
Δ_{max}	Maximum marginal gain value
$b_{v,u}$	Influence weight of directed edge (v,u) from node v to node u
$nbr^{in}(v)$	Set of in-neighbor of node v
$nbr^{out}(v)$	Set of out-neighbor of node v
$deg(v)$	Degree of node v
$deg^{in}(v)$	In-degree of node v
$deg^{out}(v)$	Out-degree of node v

Also its related concepts are defined such as sub-modularity and monotonicity. Besides, two of the most widely-used diffusion models in influence maximization will be introduced. Table 2 lists the frequently used notations in the rest of this paper. Also the # symbol means the number of elements.

3.1. Diffusion Model

A diffusion model is a conceptual framework of the information propagation, which refers to spreading the ideas or information within a social graph. It aims to simulate the spread of information (social influence) across people (nodes) through links in a social network.

In a diffusion model, each node $v \in V$ can adopt one state: active state or inactive state. The active state of node v indicates that node v can propagate the information toward its neighbors. The inactive state of node v indicates that the information cannot be propagated via node v. The node state can be changed from inactive to active when it receives the information but is not possible in vice versa. Initially, all nodes $v \in V$ are inactive. Then the k nodes are added to the seed set S, and their state will be switched to active. The information propagation procedure is triggered from the seed set S. Each node of S can activate its neighbors in separated time steps. The newly activated nodes can activate their neighbors as well. This procedure will be terminated when no new node is activated. Various diffusion models adopt different mechanisms for switching the node state from inactive to active. The Independent Cascade (IC) and Linear Threshold (LT) models are the most common diffusion models in social networks that capture the information propagation procedure [4].

3.1.1. Independent Cascade (IC) Model

The mechanism of the IC model is as follows. Suppose that the set S_{t-1} includes the activated nodes in the time step $t-1$. Each node $u \in S_{t-1}$ has a single chance to activate each inactive out-neighbor v with activation probability $P_{u,v}$ in the time step t. If node u succeeds, node v will be activated in step $t+1$; else, it will remain inactive. Regardless of the activation result of u on its neighbors, node u has no opportunity to activate the nodes in the next time steps. The diffusion process continues until no more activation is possible [4].

3.1.2. Linear Threshold (LT) Model

In the LT model, the activation of node v depends on its active in-neighbors. In this model, a random number θ_v in the range [0,1] is assigned to each node v as its activation threshold. Also weight $w_{u,v}$ is assigned as the effect of u on v for each directed edge from node u to node v, which satisfies $\sum_{u \in neighbor^{in}(v)} w_{u,v} \leq 1$. The node v will be activated if the sum of the weight of active in-neighbors v is at least θ_v; i.e., $\sum_{u \in neighbor^{in}(v) \cap u \in S} w_{u,v} \geq \theta_v$. This spreading process will be finished when no new node is activated [4].

3.2. Influence Maximization Problem

Definition 1. (Influence spread) Given the social network $G(V,E)$, initial seed set $S \subseteq V$, diffusion model M, and an integer number k as budget, the influence spread of seed set S with k nodes is defined as the expected number of activated nodes by seed set S under stochastic diffusion model M in graph G, denoted by $\delta_{G,M}(S)$.

Kempe *et al.* have shown that the influence spread function $\delta_{G,M}(S)$ satisfies three important properties: non-negatively, monotonicity, and sub-modularity [35], [36].

Definition 2. (Non-negatively) An influence spread function $\delta(\cdot)$ is non-negative if $\delta(S) \geq 0$ for any $S \subseteq V$.

Definition 3. (Monotonicity) An influence spread function $\delta(\cdot)$ is monotone if $\delta(S) \leq \delta(T)$ for any $S \subseteq T \subseteq V$.

Definition 4. (Sub-modularity) An influence spread function $\delta(\cdot)$ is sub-modular if $\delta(S \cup \{v\}) - \delta(S) \geq \delta(T \cup \{v\}) - \delta(T)$ for any $S \subseteq T \subseteq V$ and $v \in V \backslash T$.

The monotonicity property says that adding more activated nodes to a seed set S does not decrease its influence spread. The sub-modularity property means that the marginal gain of a new node is diminished as the seed set size grows.

Definition 5. (Influence maximization problem) Given the social network *G(V,E)*, diffusion model *M*, and an integer number *k* as budget, the aim is to select a set of graph nodes with *k* elements is called S^* so that if the information propagation is triggered from S^*, then the influence spread of this set on the graph *G* under stochastic diffusion model *M* will be maximized, i.e. [4]:

$$\delta_{G,M}\left(S^*\right) = argmax_{|S|\le k\ ,S\subseteq V}\ \delta_{G,\mathrm{M}}\left(S\right) \qquad (1)$$

3.2.1. Greedy Algorithm

Kempe *et al*. have proved that computation of optimum solution for influence maximization problem is *#NP-hard* under the LT and IC diffusion models [4]. In order to overcome the NP-hardness, they presented a greedy algorithm according to three properties of the influence spread function (non-negatively, monotonicity, and sub-modularity).

Theorem 1. The influence spread function *δ(·)* under the IC model is non-negative, monotone, and sub-modular [4].

Theorem 2. The influence spread function *δ(·)* under the LT model is non-negative, monotone, and sub-modular [4].

Theorem 3. If S^* is an optimal solution of influence maximization problem, and *S* is a set obtained by the greedy algorithm, then *S* satisfies $\delta(S) \ge (1\text{-}1/e)\times\delta(S^*)$ for the non-negative, monotone, and sub-modular influence spread function *δ(·)*[36].

These theorems guarantee that the greedy algorithm approximates the optimal solution with the lower bound ratio *(1-1/e) ≈ 0.63123* by evaluating the influence spread function [4]. The original greedy algorithm starts with an empty seed set *S*, and then iteratively picks node *u* with the maximum marginal gain (influence spread) as seed node until the *k* nodes are added to the seed set *S*.

Definition 6. (Marginal gain) The marginal influence spread of node *v* upon seed set *S* is defined as:

$$\Delta\left(v\,|S\right) = \delta_{G,M}\left(\mathrm{S}\cup\{\mathrm{v}\}\right) - \delta_{G,M}\left(S\right) \qquad (2)$$

Since the activated nodes by node *v* may overlap with the activated nodes by seed set *S*, the influence spread function is not extensible, i.e. $\delta_{G,M}\left(\mathrm{S}\cup\{\mathrm{v}\}\right) \ne \delta_{G,M}\left(S\right) + \delta_{G,M}\left(v\right), v \notin S$. So, both $\delta_{G,M}(S)$ and $\delta_{G,M}(S\cup\{v\})$ must be calculated from scratch, resulting in a huge computation overhead. However, evaluating the exact marginal gain $\delta_{G,M}(S)$ is *#P-hard* under both the IC and LT models [18], [20], which cannot be completed in a polynomial time. Hence, it is estimated by conducting a huge number of MC-simulations (e.g. $R \approx 10{,}000$) to achieve an accurate estimation of $\delta_{G,M}(S)$. The time complexity of the MC-simulation is *O(mR)* So the time complexity of the naïve greedy algorithm is *O(knmR)*, where *R* is the number of MC-simulations. Algorithm 1 shows the naïve greedy algorithm in detail.

Algorithm 1: Naïve greedy algorithm

```
Input: Graph $G(V,E)$, number of seed nodes $k$, diffusion model $M$
Output: seed set $S$
1:   Initialize $S=\emptyset$, $R=10,000$
2:   while $|S|<k$ do
3:       for each $v \in V\text{-}S$ do
4:           $sum=0$
5:           for $i=1$ to $R$ do
6:               $sum=sum + \delta(S\cup\{v\})$ // δ(·) is estimated by M
7:           end of for
8:           $\Delta(v|S)=sum / R$
9:       end of for
10:      $S=S \cup \{argmax_{v\in V\text{-}S}\ \Delta(v|S)\}$
11:  end of while
12:  output $S$
```

3.2.2. CELF Algorithm

Although the naïve greedy algorithm perverts the NP-hardness of the influence maximization problem [4], it must call the influence spread evaluation procedure (MC-simulation) for *O(kn)* times to pick the *top-k* nodes, where *n* is the number of graph nodes. Leskovec *et al*. have proposed the Cost-Effective Lazy Forward (CELF) algorithm to overcome the naïve greedy algorithm inefficiency [5]. The CLEF algorithm significantly reduces the number of influence spread estimations by exploiting the sub-modularity property of influence spread function *δ(·)*.The intuition behind CELF is that the marginal gain of node *v* in the current iteration cannot exceed its marginal gain in the previous iterations. More formally, suppose that *S* is the current seed set, and $\Delta(v|S)$ is the marginal gain of *v* with respect to *S*. According to the sub-modularity property of the influence function, $\Delta(v|S)$ is an upper bound for any $\Delta(v|T)$ for $S\subseteq T\subseteq V$. Based on this property, the CELF algorithm estimates $\Delta(v|\emptyset)$ using MC-simulations for all nodes in its first round. One node with the highest marginal gain is added to seed set S. Then $\Delta(v|\emptyset)$ can be used as an upper bound as follows. In the rest of iteration, the CELF algorithm re-estimates $\Delta(v|S)$ for nodes $v \in V\backslash S$ using MC-simulations in the descending order of their upper bounds. Instead of estimating $\Delta(v|S)$ for all nodes, CELF terminates the current iteration whenever the highest upper bound of the unestimated nodes is already smaller than the highest upper bound of the estimated nodes. In other words, $\Delta(v\,|\,S)$ is re-estimated only for the front node of queue. Then the queue is resorted in the descending order. If a node remains at the front of queue, it is selected as the seed set node. Although the CELF optimization could not improve the worst-case time complexity of the naïve greedy algorithm, the practical results demonstrate that the CELF algorithm improves the time efficiency of the naïve greedy algorithm to 700 times faster [5].

Although the CELF algorithm could improve the time efficiency of the naïve greedy algorithm by avoiding recomputation of $\delta(S)$, its first iteration is very time-consuming because it has to call the influence spread estimation procedure (MC-simulations) for all nodes of the graph, similar to the naïve greedy algorithm. This issue arises a huge computational overhead in the first iteration. In the next section, this dilemma will be tackled by proposing a new optimization strategy.

4. Optimized-CELF Algorithm

As mentioned in the previous sections, Leskovec *et al*. have exploited the sub-modularity property of the influence function, developing the CELF optimization [5] in order to reduce the running time of the naïve greedy algorithm. The main idea is that the marginal gain $\Delta(v\,|\,S)$ of a node v in the previous iterations is more than its marginal gain in the current iteration. This optimization can significantly reduce the number of estimation calls on the influence spread of nodes. However, the first iteration of the CELF algorithm is still very time-consuming because it has to make $|V|$ calls of the influence spread estimation for all nodes by expensive MC simulations. In other words, the function $\delta(\cdot)$ must be called $|V|$ times, exactly similar to the naive greedy algorithm. Hence, in this paper, a novel approach is proposed in order to reduce the number of calling function $\delta(\cdot)$ in the first iteration of CELF. The main idea is the estimation of influence spread $\delta(\cdot)$ only for the core nodes of the graph instead of all graph nodes. The influence spread of the other nodes is estimated based on the obtained influence spread of the MSN nodes. Thus by avoiding huge MC-simulations, the long-running time of the first iteration will be reduced significantly. The core nodes is defined in definition 7.

Algorithm overview:

1. *Constructing a set of core nodes of graph (MSN set).*
2. *Estimating the influence spread $\delta(\cdot)$ for MSN set nodes using MC-simulations.*
3. *Estimating the influence spread of the other nodes based on the obtained influence spread of MSN nodes according to (3).*
4. *Generating a queue of nodes in the descending order of their influence value.*
5. *Selecting the front node of the queue as the first node of seed set S.*
6. *Repeating the following steps until $|S| \le k$.*
7. *Estimating the influence spread $\delta(\cdot)$ for the front node v of the queue.*
8. *Resorting queue in the descending order of influence value.*
9. *If node v remains at the front of the queue, it is selected as seed set S nodes, and go to step 6; else, go to step 7.*

Definition 7. (Core nodes) Given the directed graph *G(V,E)*, the core node set of graph *G* is defined as a set of graph nodes so that all nodes of graph *G* could be accessible via them at most in one distance.

Algorithm 2 constructs a minimal set of core nodes, which span all graph nodes. The input of this algorithm is directed graph *G*, and its output is the MSN set. At each iteration, one unvisited node with the highest degree is selected as a core node and added to the MSN set. Thereafter, the selected node along with its neighbors are marked to the visited node. This procedure is repeated until all nodes are visited.

Algorithm 2: ***Core nodes (MSN)***

```
Input: Directed graph G(V,E)
Output: W
1:  W=∅
2:  for each v ∈ V do
3:        visited[v] = False
4:  end of for
5:  Sort all nodes in list V in descending order of their degree
6:  for each v ∈ V do // taken in descending order by degree
7:        if visited[v] == False then
8:           W = W ∪{v}
9:           visited[v] = True
10:          for each u ∈ neighbor(v) do
11:             visited[u] = True
12:          end of for
13:       end of if
14: end of for
15: return W
```

Algorithm 2 works as follows. Lines 2–4 set the visited attribute of every node to a false value. Line 5 sorts all nodes in the descending order of their degree at the list *V'*. For loop of lines 6–14 iterates the following procedure is carried out for unvisited nodes: firstly, it adds the unvisited node v (with the highest degree) to set *W*. Then it changes the visited attribute of node v and its neighbors to a true value. Finally, algorithm 2 returns set *W* as the core node set.

Consider the following example in Figure 1. Figure 1(a) shows the input graph G, which has 9 nodes and 14 edges. The sorted nodes in the descending order of degree are as follow {#8, #4, #6, #0, #2, #7, #5, #3, and #1}. As shown in Figure 1(b), node #8, which has the biggest degree, is the first node chosen by algorithm 2 and added to core node set *W*. Then this algorithm changes the visited attribute of the following nodes to true (node #8 along with its neighbors #3, #4, #5, and #6, shown with red color). In the next step in Figure 1(c), node #4 has the biggest degree among the unvisited nodes #4, #0, #2, and #1. Thus node #4 is added to core node set $\mathcal{W}$. The visited attribute of node #4 and its neighbors #2 and #3 are changed to visited = true. As shown in Figure 1(d), node #0, which has the biggest degree among the unvisited nodes {#0 and #1}, is the last node chosen by algorithm 2 because after changing the visited attribute of node #0 along with its neighbors #1, and #2 to true, there will no unvisited nodes. Finally, the set $\mathcal{W}$, which is the core node set produced by algorithm 2, contains the three nodes #8, #4, and #0. In other words, all nodes will be accessible at most with one distance via the core node set {#8, #4, and #0}.

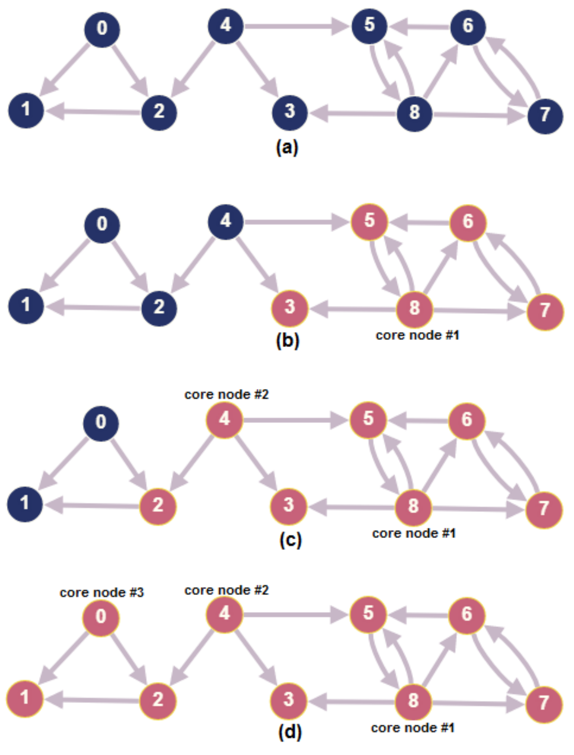

Figure 1. Procedure of selecting core nodes by algorithm 2

Time Complexity of algorithm 2:

Given the directed graph *G(V,E)*, let n and m be the number of nodes and edges in the graph *G*, respectively. The overhead for initialization is *O(n)* in lines 2-5. The sort procedure in line 5 takes *O(n log n)* time. Lines 6-14 require $O(\sum_{v\in V} degree(v)) = O(m)$ time because it checks all nodes (line 6) along with their neighbors (line 10). Thus the total time complexity of algorithm 2 is $O(n + n \log n + m) \approx O(n \log n + m)$ in the worst case. Since this algorithm selects the nodes according to the highest degree in each round and ignores the neighbors of the selected nodes, its time complexity is better than the worst-case complexity. As mentioned earlier, algorithm 2 repeatedly selects the unvisited node *v* with the maximum degree to push on the set *W*, and then eliminates node *v* along with its neighbor nodes from the graph *G*. This selection strategy based on the maximum degree cause that most nodes are removed in each iteration. As a result, the size of output set *W* will be minimal. Intuitively, since algorithm 2 pick up the nodes based on the maximum degree, the core nodes of the graph will be obtained efficiently and effectively. The practical results demonstrate that the size of the set $\mathcal{W}$ is very small rather than the size of the graph. So far, we estimated the influence spread $\delta(\cdot)$ only for the nodes of the set *W* in the first round of our proposed algorithm. However, the influence spread of all nodes must be provided for the next rounds. Hence, the influence spread of the other nodes is estimated based on the following principle. The expected influence spread of node $v \in V$ is linearly dependent on the influence of its neighbors under the IC model, i.e.:

$$E\left[\delta(v)\right] = 1 + \sum_{u \in nbr^{out}(v)} b_{v,u} \times E\left[\delta(u)\right] \qquad (3)$$

where $\delta(u)$ is the influence spread of node *u*; $b_{v,u}$ is the weight of the directed edge from node *v* to node *u*, $b_{v,u} = 1/degree^{in}(u)$. The term “1” in (3) is the influence of node v on itself. For example, consider the graph shown in Figure 2. Suppose that the expected influence spread value of the node v_4 is equal to 1. According to (3), the expected influence spread value of the node v_1 can be calculated as:

$$E\left[\delta(v_1)\right] = 1 + \left(b_{v_1,v_2}.\delta(v_2) + b_{v_1,v_3}.\delta(v_3)\right)$$
$$= 1 + \left(b_{v_1v_2}.\left(1 + b_{v_2v_4}.\delta(v_4)\right) + b_{v_1v_3}.\left(1 + b_{v_3v_4}.\delta(v_4)\right)\right)$$
$$= 1 + \left(0.2 \times (1 + 0.3 \times 1) + 0.4 \times (1 + 0.5 \times 1)\right) = 1.86$$

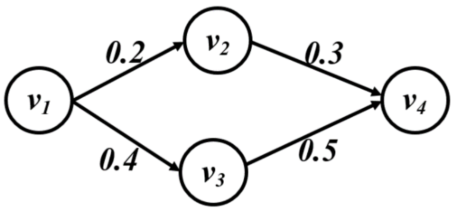

Figure 2. Influence calculation.

Note that the expected influence spread value of nodes *b* and *c* was computed to be equal to 1.3 and 1.5, respectively, according to (3). In this example, node v_1 with the highest influence spread is selected as seed set node.

In other words, for each node $v \in V$, if the influence spread of out-neighbors of node *v* is provided, then the influence spread of node *v* will be obtained directly, without calling the expensive function $\delta(\cdot)$.

In fact, (3) calculates the influence spread value as a linear combination of the influences of nodes. However, there is no concern about the overestimation of nodes' influence on each other $\delta_{G,M}(S \cup \{v\}) = \delta_{G,M}(S) + \delta_{G,M}(v)$ because (3) is used only in the first iteration of the algorithm, and there is no overlap among seeds when $S=\emptyset$. The practical results indicates that the proposed optimized-CELF algorithm significantly increases the speed of the first iteration of basic CELF with maintenance accuracy. The optimized-CELF implementation is elaborated in algorithm 3. It seems that the first iteration of the algorithm has the extra overhead but it is relatively insignificant compared to the achieved running time gain.

The optimized-CELF algorithm is divided into two general parts. In lines 1-33, the first round of the algorithm is performed to select the first node of seed set *S*. The remaining nodes of seed set *S* are selected in lines 34-51. $\Delta(v|S)$ and *visited[v]* vectors are initialized for all $v \in V$ by 0 and false values, respectively (line 1). $\Delta(v|S)$ indicates the influence spread (marginal gain) of node *v* with respect to *S*. Algorithm 2 detects the minimal spanning nodes (core nodes) and puts them in the set *W* (line 2). Lines 4 -16 estimates the influence spread $(\delta(\cdot))$ for nodes $v \in W$ using MC-simulation under the IC model. Also in the synchronous procedure, any node $u \in neighbor^{in}(v)$ that had not been visited so far will be added to the queue $Q_{in\text{-}nbr}$ to calculate its influence spread in the future steps. In the while loop (lines 17-26), until queue $Q_{in\text{-}nbr}$ is not empty, the front node of $Q_{in\text{-}nbr}$ is removed, and its influence spread is calculated according to (3). Also in the synchronous procedure, any node $u \in neighbor^{in}(v)$ that had not been visited so far, will be added to queue $Q_{in\text{-}nbr}$. In the next step, all nodes along with their influence spread value are added to queue Q_{CELF} (lines 27 - 29). Line 30 sorts Q_{CELF} according to the nodes' influence spread value in the descending order. The front node of Q_{CELF} (that has the highest marginal gain) is removed and added to the seed set *S* as the first influential node (lines 31-33). Lines 34-51 demonstrate the iterative procedure of the basic CELF algorithm to find the other seed set nodes as follows. The influence spread of the front node *v* of Q_{CELF} is re-estimated by MC simulation. After resorting queue Q_{CELF}, if node *v* remains at the front of queue Q_{CELF}, it is added to seed *S* as the node with the highest marginal gain in the current round. This procedure will be terminated when the number of seed nodes is satisfied.

Algorithm 3: Optimized-CELF

```
Input: Directed graph $G(V,E)$, Number of seed nodes $k$, Diffusion Model $M$
Output: seed set $S$
1:  Initialize $S = \emptyset, R = 10,000$
    Initialize $\Delta(v|S) = 0$, $visited[v] = False$ for each $v \in V$
2:  $W$ = Construct the MSN set of graph $G$ by Algorithm 2
3:  for each $v \in W$ do
4:      $sum = 0$
5:      for $i = 1$ to $R$ do
6:          $sum = sum + \delta(v)$ // $\delta(\cdot)$ is estimated by diffusion model $M$
7:      end of for
8:      $\Delta(v|S) = sum / R$
9:      $visited[v] = True$
10:     for each $u \in neighbor^{in}(v)$ do
11:         if $visited[u] == False$ then
12:             $Q_{in\text{-}nbr}.Push(u)$
13:             $visited[u] = True$
14:         end of if
15:     end of for
16: end of for
17: while $Q_{in\text{-}nbr}.IsEmpty() == False$ do
18:     $v = Q_{in\text{-}nbr}.pop()$
19:     $sum = 0$
20:     for each $u \in neighbor^{out}(v)$ do
21:         $sum = sum + \Delta(u|S) \times (1/degree^{in}(v))$     //by (3)
22:     end of for
23:     $\Delta(v|S) = 1 + sum$
24:     for each $u \in neighbor^{in}(v)$ do
21:         if $visited[u] == False$ then
22:             $Q_{in\text{-}nbr}.Push(u)$
23:             $visited[u] = True$
24:         end of if
25:     end of for
26: end of while
27: for each $v \in V$ do
28:     $Q_{CELF}.Push(v)$
29: end of for
30: sort $Q_{CELF}$ in descending order based on $\Delta(v|S)$
31: $v = Q_{CELF}.POP()$
32: $S = S \cup \{v\}$
33: $\Delta_{max} = \Delta(v|S)$
34: while $|S| \leq k$ do
35:     $satisfied = False$
36:     while $satisfied == False$ do
37:         $v = Q_{CELF}.Front()$
38:         $sum = 0$
39:         for $i = 1$ to $R$ do
40:             $sum = sum + \delta(S \cup \{v\})$ // $\delta(\cdot)$ is estimated by diffusion model $M$
41:         end for
42:         $\Delta(v|S) = (sum / R) - \Delta_{max}$
43:         resort $Q_{CELF}$ in descending order based on $\Delta(v|S)$
44:         if $v == Q_{CELF}.Front()$ then
45:             $satisfied = True$
46:         end of if
47:     end of while
48:     $v = Q_{CELF}.POP()$
49:     $S = S \cup \{v\}$
50:     $\Delta_{max} = \Delta_{max} + \Delta(v|S)$
51: end of while
52: return $S$
```

Table 3. Statistical properties of six real datasets.

Networks statistics	Hamsterster friendships	Facebook (NIPS)	Erdős	Wikipedia talk	DBLP	Google+ (NIPS)
# Nodes	1,858	2,888	6,927	7,586	12,590	23,628
# Edges	12,534	2,981	11,850	47,070	49,759	39,242
Maximum degree	272	769	507	5,157	714	2,771
Average degree	13.5	2.1	3.4	12.4	7.9	3.321 65
Diameter	14	9	4	8	10	8
Gini coefficient	0.61	0.51	0.65	0.87	0.66	0.659 896
Network format	Undirected	Undirected	Undirected	Directed	Directed	Directed
Category	Online social network	Online social network	Co-authorship network	Communication network	Citation network	Online social network

Optimized-CELF Algorithm Time Complexity: Let n and m be the number of nodes and edges in the graph, respectively, n' and m' be the number of nodes and edges in the graph restricted to core node set W, respectively, and R be the number of MC simulations. The original CELF algorithm takes $O(nmR)$ in its first iteration. This is equivalent to lines 2-26 of the optimized-CELF algorithm. The optimized-CELF algorithm first calls algorithm 2 with time complexity $O(n \log n + m)$ in line 2. The loop in lines 3-16 requires $O(n'(R+\sum_{v\in W} deg(v))) = O(n'R + m')$ time. Suppose that n'' is the size of V-W. Thus lines 17-26 require $O(n'' \sum_{v\in V-W} deg(v)) \approx O(m)$. Thus the total time complexity of the optimized-CELF algorithm is $O((n \log n + m)+(n'R + m') + (m))$. Since $m' \ll m$, therefore, it takes $O(n \log n + 2m + n'R) \approx O(n \log n + m + n'R)$, which is the near-linear time complexity versus the original CELF algorithm with time complexity $O(nmR)$.

5. Experiments and Results

In this section, several experiments were conducted on six real datasets in order to verify the performance of the optimized-CELF algorithm. The optimized-CELF algorithm was compared with the CELF algorithm on two aspects: influence spread (effectiveness) and running time (time efficiency). In this work, the naïve greedy algorithm was not considered since it had the same influence spread results with more running time compared to the CELF algorithm. The main goal of the experiments was to prove that the optimized-CELF algorithm could improve the time efficiency of the CELF algorithm, while maintaining effectiveness. All simulations (our proposed algorithm and original algorithm) were implemented by the C++ language using the SNAP libraries and Standard Template Library (STL) and executed on a windows machine with a single processor Intel Core-i5-6400 (2.70 GHz) and 16GB memory. These codes are available at https://github.com/mohsentaherinia/OptimizedCELF website.

5.1. Experimental Setup

5.1.1. Datasets

In order to achieve the comparable results, six real datasets with various sizes were employed.

Hamsterster friendships [37]**:** This is an online social network from hamsterster.com, which contains the user–user friendships.

Facebook (NIPS) [38]**:** This online social network contains Facebook friendships between the individuals.

Erdős [37]**:** This is a co-authorship network around Paul Erdős, which was assembled by the Pajek project. This network contains the individuals who have written the papers with Paul Erdős.

Wikipedia talk [38]**:** This is a directed communication network of Esperanto Wikipedia. Each node in the network is a user, and an edge from user u to user v indicates that the user u wrote a message to the user v.

DBLP [38]**:** This citation network contains a database of scientific publications. Each node in this directed network is a publication, and each edge represents a citation of a publication by another publication.

Google+ (NIPS) [37]**:** Google+ is a directed social network, which contains the circles of the relationship between the individuals. A directed edge in this network shows that one user has the other user in his circles.

Table 3 summarizes the statistical properties of these datasets. All datasets are available at the SNAP library website maintained by Jure Leskovec https://snap.stanford.edu/data/ or the http://networkrepository.com/ website.

5.1.2. Diffusion Model

This paper uses the independent cascade model as a diffusion model since it is extensively studied in a similar work. The propagation probability from node u to node v is $P_{u,v} = 0.01$.

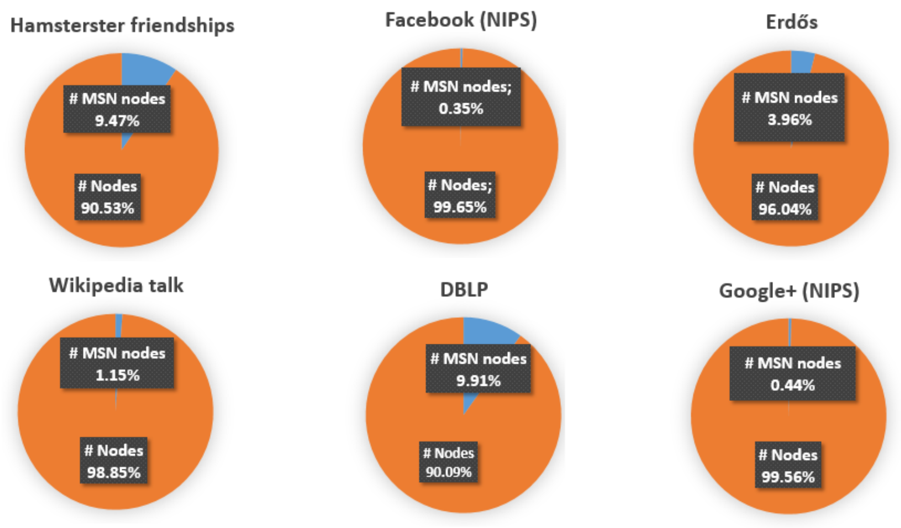

Figure 3. Comparison of size of MSN set *vs.* size of the graph for all datasets.

5.1.3. Evaluation Metrics

Efficiency and effectiveness are the most common evaluation criteria in the influence maximization problem. The elapsed time for the complete performing of the optimized-CELF is accumulated in order to evaluate the algorithm time efficiency. Also to evaluate its effectiveness, the influence spread values were estimated for various seed sizes from 1 to 50. The influence spread value was equal to the number of eventually activated nodes under the IC diffusion model. In addition to the two mentioned criteria, this paper analyzes the size of the MSN set as well.

5.2. Experimental Result

In order to estimate the influence spread more accurately, we set $R = 10{,}000$ as the number of MC-simulations. Also the following algorithms were executed for $k=1$ to 50.

5.2.1. Minimal Spanning Node Analysis

As mentioned in the previous sections, one of the most outstanding innovations of this paper is the detection of core nodes of the network by algorithm 2 effectively and efficiently. These nodes construct one set called minimal spanning nodes (MSN set), which is used to estimate the influence spread using the MC-simulation in the next step. Table 4 shows the size of the generated MSN set for each dataset.

Figure 3 compares the number of nodes in the MSN set with the number of graph nodes in order to show the performance of algorithm 2 better. As it could be seen, the ratios of the MSN nodes to the graph nodes were 9.47%, 0.35%, 3.96%, 1.15%, 9.91%, and 0.44% for Hamsterster, Facebook, Erdős, Wikipedia talk, DBLP, and Google+, respectively. The Facebook, Wikipedia talk, and Google+ datasets have the lowest percentage of nodes. It could be intuitively stated that the denser graphs with lower diameters could produce the small MSN set. In general, the MSN size to graph size ratio fell into the range of 0.35% to a maximum of 9.91%. As one can see in the next section, performing the MC- simulation on 0.35% to 9.91% of network nodes instead of all network nodes will produce wonderful effects on the running time gain.

5.2.2. Influence Spread Analysis

Figure 4 shows the influence spread of the optimized-CELF and CELF for six real datasets. The x-axis indicates the number of seed set nodes from 1 to 50. The y-axis represents the number of nodes that are eventually activated. As expected, both algorithms have the same performance in all datasets. In other words, the optimized-CELF produces high-quality seed sets similar to CELF with more speed. This equal performance goes back to the same strategy of the two algorithms in selecting the seed nodes, which is the node selection based on the maximum marginal gain.

Table 4. Size of the MSN set for six real datasets.

Dataset	# Nodes	# MSN nodes
Hamsterster friendships	1,858	176
Facebook (NIPS)	2,888	10
Erdős	6,927	274
Wikipedia talk	7,586	87
DBLP	12,590	1248
Google+ (NIPS)	23,628	105

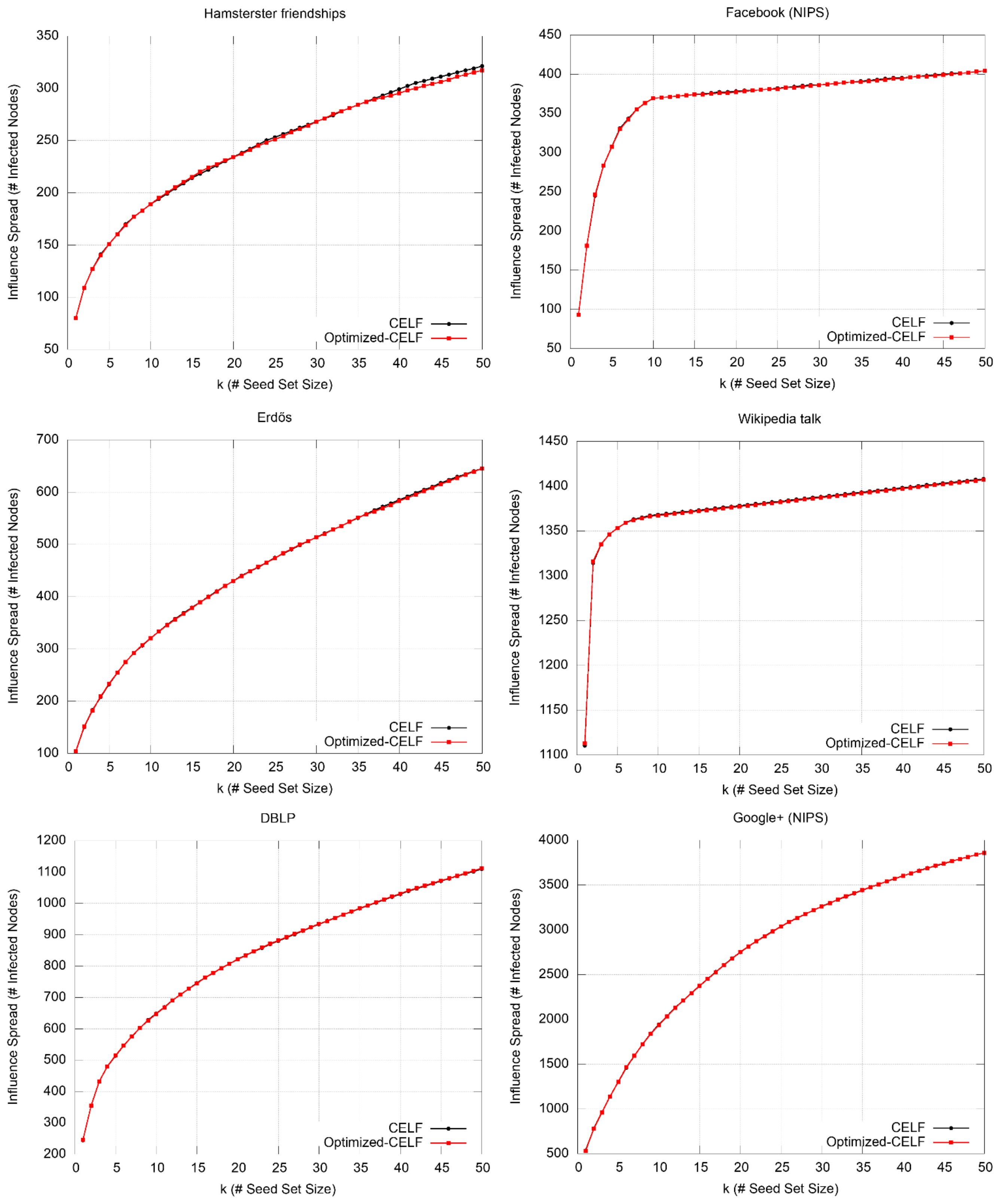

Figure 4. Comparison of influence spread of optimized-CELF and CELF algorithms on six real datasets for k = 1 to 50.

There is a slight increase or decrease in the influence spread value in some datasets. For example, in the Hamsterster dataset, the optimized-CELF increases the influence spread maximum +0.11% (when *k=17*), and it decreases the influence spread maximum -0.27% (when *k=45*). In other words, the influence spread fluctuates between +0.11% and -0.27% by optimized-CELF.

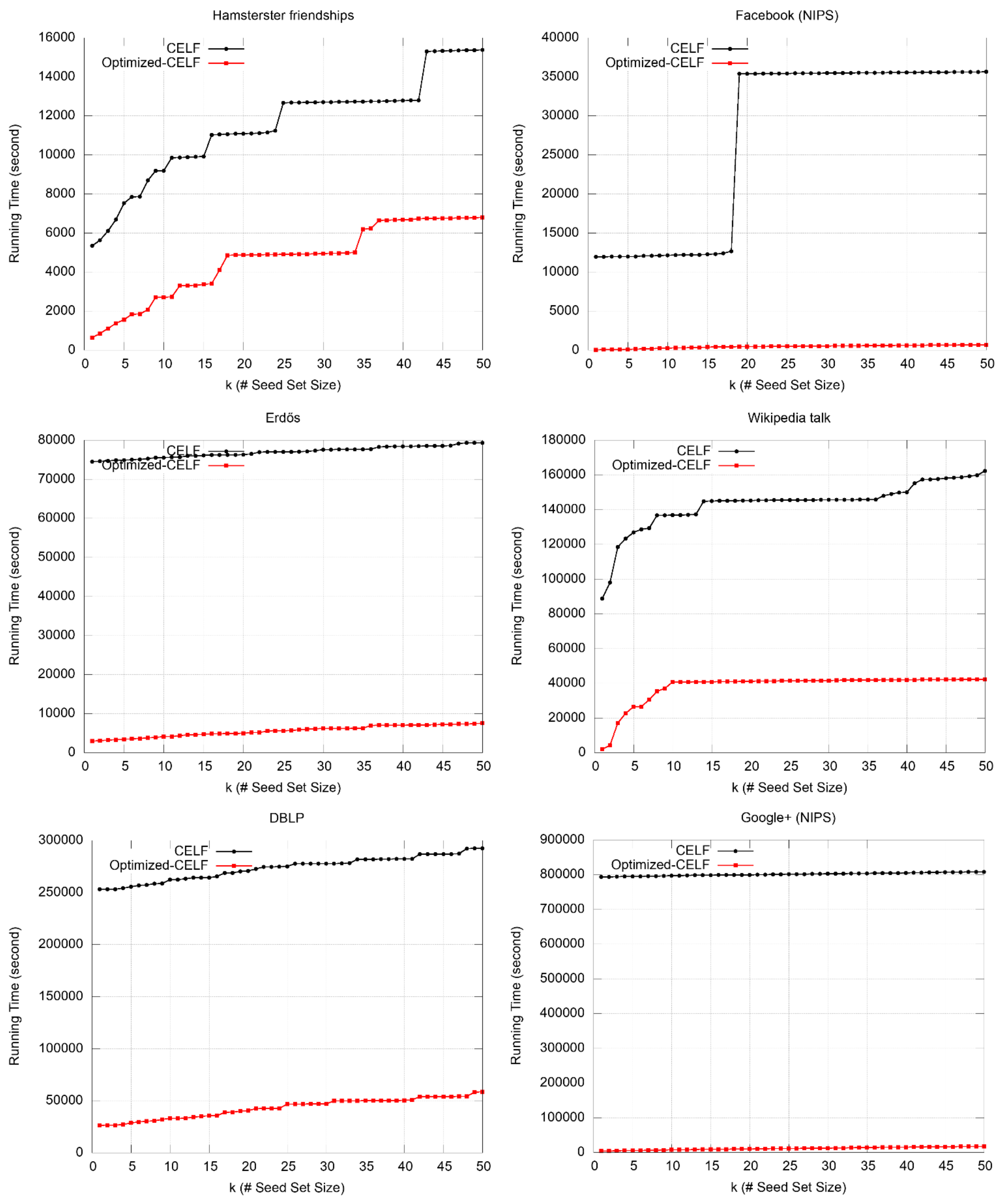

Figure 5. Comparison of running time of optimized-CELF and CELF algorithms on six real datasets for k = 1 to 50.

These slight changes are due to the stochastic nature of the diffusion model and the MC-simulation, which can be ignored.
Other DSs did not change significantly in terms of effectiveness.

5.2.3. Running Time Analysis

Figure 5 demonstrates the running time of the optimized-CELF and CELF algorithms for all datasets. The x-axis represents the number of seed set nodes from 1 to 50. The y-axis indicates the running time in seconds.

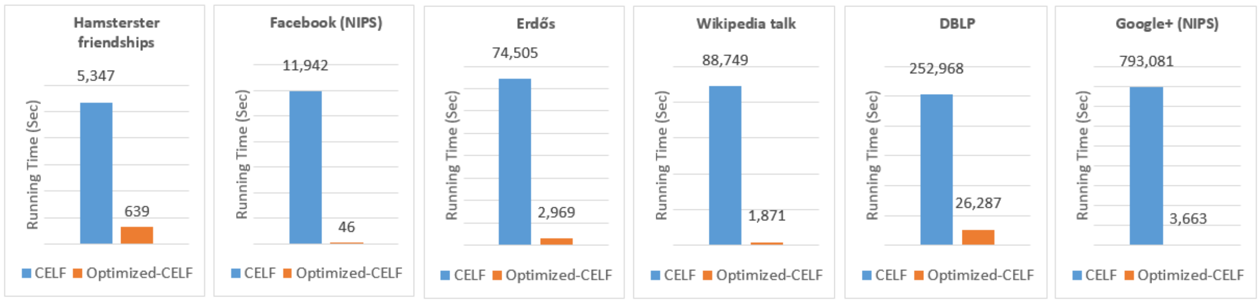

Figure 6. Comparison of running time of optimized-CELF and CELF algorithms on six real datasets when k = 1.

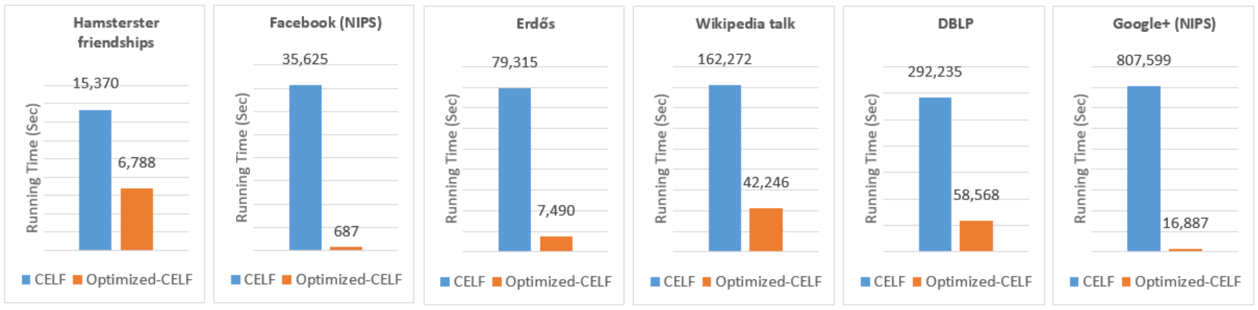

Figure 7. Comparison of running time of optimized-CELF and CELF algorithms on six real datasets when k = 50.

It is clear that the optimized-CELF algorithm is faster than the CELF algorithm in all datasets. The main goal of the proposed algorithm is to reduce the running time of the first iteration of the CELF algorithm. However, since the subsequent iterations are directly dependent on the first iteration, the running time of the following iterations is also improved significantly.

In more detail, Tables 5 and 6 focus on the running time gain in all datasets for *k=1* and *k=50*, respectively.

Table 5. Running time gain of optimized-CELF for k = 1.

Dataset	CELF	Optimized-CELF	Gain
Hamsterster friendships	5,347	639	**88.05%**
Facebook (NIPS)	11,942	46	**99.61%**
Erdős	74,505	2,969	**96.02%**
Wikipedia talk	88,749	1,871	**97.89%**
DBLP	252,968	26,287	**89.61%**
Google+ (NIPS)	793,081	3,663	**99.54%**

Table 6. Running time gain of optimized-CELF for k = 50.

Dataset	CELF	Optimized-CELF	Gain
Hamsterster friendships	15,370	6,788	**55.84%**
Facebook (NIPS)	35,625	687	**98.07%**
Erdős	79,315	7,490	**90.56%**
Wikipedia talk	162,272	42,246	**73.97%**
DBLP	292,235	58,568	**79.96%**
Google+ (NIPS)	807,599	16,887	**97.91%**

The third column of Tables 5 and 6 demonstrates the achieved running time gain by optimized-CELF compared to CELF.

Table 5 verifies that the achieved time gain falls into the range between 88.05% to 99.61% in all datasets (for *k=1*).

In other words, in the first iteration of the optimized-CELF algorithm, the achieved time gain is equal to 95% on average in all datasets. In Table 6 (for *k=50*), the lowest time gain is 55.84% in the Hamsterster dataset, and the highest time gain is 98.07% for the Facebook dataset.

It is visible that the optimized-CELF achieves 83% improvement in time gain on average in all datasets (for *k=50*). Figures 6 and 7 were also plotted for a better visual differentiation of the time gain of optimized-CELF in all datasets for *k=1* and *k=50*, respectively.

6. Discussion

As it could be seen from the results obtained in the previous section, the optimized-CELF algorithm could improve the time efficiency of the basic CELF algorithm (95% when *k=1* and 82% when *k=50*, in the average of all datasets). This running time gain occurs while the effectiveness of the basic CELF algorithm is not diminished. The main reason for this runtime improvement is that in the first round of the optimized-CELF algorithm, the influence spreads were estimated only for the MSN set instead of all nodes. Of course, the size of the MSN set plays a vital role in the time efficiency of the optimized-CELF algorithm. Intuitively, a denser network with a lower diameter produces a small MSN set, and subsequently, achieves a better running time gain. Fortunately, it was seen that in some cases such as the Facebook and Google+ datasets, despite the very small MSN set, the efficiency of optimized-CELF did not decrease compared to CELF. On the other hand, since one node with the highest marginal gain was selected as the seed set node in each iteration, the quality of generated seed set by the proposed algorithm would not be changed compared to the basic CELF algorithm.

7. Conclusions and future directions

Undoubtedly, the Greedy CELF algorithm, as the best algorithm in the influence maximization problem in terms of effectiveness, can generate a high-quality seed set among all the presented algorithms. Hence, the researchers have employed it extensively in order to invent new models to solve the various aspects of the influence maximization problem. One of the main drawbacks of CELF is the time inefficiency of its first iteration because it has to run the expensive MC-simulation to estimate the influence spread of all nodes in its first round.

In this paper, a new algorithm was proposed in order to tackle this challenge of CELF. The main idea was to construct a set of the core nodes of the network, namely minimal spanning nodes (MSN). Then the influence spreads were estimated only for the MSN set instead of all nodes of the network. The influence spreads of the other nodes were calculated according to (3). Therefore, many expensive MC-simulations would be avoided, and the running time would be reduced. In order to verify the time efficiency and accuracy of the optimized-CELF algorithm, various experiments were conducted. The experimental results demonstrated that the optimized-CELF algorithm and the CELF algorithm performed similarly in terms of accuracy (the equal influence spread); however, the optimized-CELF algorithm outperformed the CELF algorithm in terms of efficiency (running time).

There are several promising directions for future research works:

- ✓ Extending to other diffusion models such as the LT, WC, SIR, and TR models.
- ✓ Embedding the proposed algorithm to the CELF++ algorithm, which uses the CELF approach directly.
- ✓ Applying the optimized-CELF algorithm to other approaches that exploit the CELF algorithm such as SimPath, IPA, INCIM, HybridIM, and IMPC.

Acknowledgments

This research work did not receive any specific grant from the funding agencies in the public, commercial or not-for-profit sectors. The authors would like to thank the anonymous reviewers for their valuable comments that helped improve the quality of this paper.

References

[1] Y. Li, J. Fan, Y. Wang, and K.-L. L. Tan, “Influence Maximization on Social Graphs: A Survey,” *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 10, pp. 1852–1872, Oct. 2018.

[2] K. Li, L. Zhang, and H. Huang, “Social Influence Analysis: Models, Methods, and Evaluation,” *Engineering*, vol. 4, no. 1, pp. 40–46, Feb. 2018.

[3] P. Domingos and M. Richardson, “Mining the network value of customers,” in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001, pp. 57–66.

[4] D. Kempe, J. Kleinberg, and É. Tardos, “Maximizing the spread of influence through a social network,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003, pp. 137–146.

[5] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. Vanbriesen, and N. Glance, “Cost-effective outbreak detection in networks,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007, pp. 420–429.

[6] A. Goyal, W. Lu, and L. V. S. Lakshmanan, “CELF++: Optimizing the greedy algorithm for influence maximization in social networks,” in *Proceedings of the 20th International Conference Companion on World Wide Web, WWW 2011*, 2011, pp. 47–48.

[7] W. Chen, Y. Wang, and S. Yang, “Efficient influence maximization in social networks,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009, pp. 199–207.

[8] S. Cheng, H. Shen, J. Huang, G. Zhang, and X. Cheng, “StaticGreedy: Solving the scalability-accuracy dilemma in influence maximization,” in *International Conference on Information and Knowledge Management, Proceedings*, 2013, pp. 509–518.

[9] C. Zhou, P. Zhang, W. Zang, and L. Guo, “On the Upper Bounds of Spread for Greedy Algorithms in Social Network Influence Maximization,” *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 10, pp. 2770–2783, Oct. 2015.

[10] Y. Tang, X. Xiao, and Y. Shi, “Influence maximization: Near-optimal time complexity meets practical efficiency,” in *Proceedings of the ACM SIGMOD International Conference on Management of*

Data, 2014, pp. 75–86.

[11] Y. Tang, Y. Shi, and X. Xiao, “Influence maximization in near-linear time: A martingale approach,” in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2015, vol. 2015-May, pp. 1539–1554.

[12] S. Banerjee, M. Jenamani, and D. K. Pratihar, “A survey on influence maximization in a social network,” *Knowl. Inf. Syst.*, vol. 62, no. 9, pp. 3417–3455, Sep. 2020.

[13] S. Peng, Y. Zhou, L. Cao, S. Yu, J. Niu, and W. Jia, “Influence analysis in social networks: A survey,” *J. Netw. Comput. Appl.*, vol. 106, pp. 17–32, Mar. 2018.

[14] L. Page, L. Page, S. Brin, R. Motwani, and T. Winograd, “The PageRank Citation Ranking: Bringing Order to the Web,” -, 1998.

[15] J. M. Kleinberg, “Authoritative sources in a hyperlinked environment,” *J. ACM*, vol. 46, no. 5, pp. 604–632, 1999.

[16] A. Goyal, W. Lu, and L. V. S. Lakshmanan, “SIMPATH: An Efficient Algorithm for Influence Maximization under the Linear Threshold Model,” in *2011 IEEE 11th International Conference on Data Mining*, 2011, pp. 211–220.

[17] M. Kimura, K. Saito, R. Nakano, and H. Motoda, “Extracting influential nodes on a social network for information diffusion,” *Data Min. Knowl. Discov.*, vol. 20, no. 1, pp. 70–97, Jan. 2010.

[18] W. Chen, C. Wang, and Y. Wang, “Scalable influence maximization for prevalent viral marketing in large-scale social networks,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010, pp. 1029–1038.

[19] J. Kim, S. K. Kim, and H. Yu, “Scalable and parallelizable processing of influence maximization for large-scale social networks?,” in *Proceedings - International Conference on Data Engineering*, 2013, pp. 266–277.

[20] W. Chen, Y. Yuan, and L. Zhang, “Scalable influence maximization in social networks under the linear threshold model,” in *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2010, pp. 88–97.

[21] R. Narayanam and Y. Narahari, “A shapley value-based approach to discover influential nodes in social networks,” *IEEE Trans. Autom. Sci. Eng.*, vol. 8, no. 1, pp. 130–147, Jan. 2011.

[22] K. Jung, W. Heo, and W. Chen, “IRIE: Scalable and robust influence maximization in social networks,” in *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2012, pp. 918–923.

[23] 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 2014 - Proceedings of the 2014 ACM International Conference on Information and Knowledge Management*, 2014, pp. 171–180.

[24] S. Cheng, H.-W. Shen, J. Huang, W. Chen, and X.-Q. Cheng, “IMRank: Influence Maximization via Finding Self-Consistent Ranking,” *SIGIR 2014 - Proc. 37th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, pp. 475–484, Feb. 2014.

[25] N. Ohsaka, T. Akiba, Y. Yoshida, and K. Kawarabayashi, “Fast and accurate influence maximization on large networks with pruned Monte-Carlo simulations,” in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014, pp. 138–144.

[26] S. Galhotra, A. Arora, and S. Roy, “Holistic Influence Maximization: Combining Scalability and Efficiency with Opinion-Aware Models,” *Proc. ACM SIGMOD Int. Conf. Manag. Data*, vol. 26-June-20, pp. 743–758, Feb. 2016.

[27] N. Sumith, B. Annappa, and S. Bhattacharya, “Influence maximization in large social networks: Heuristics, models and parameters,” *Futur. Gener. Comput. Syst.*, vol. 89, pp. 777–790, Dec. 2018.

[28] M. Zarezade, E. Nourani, and A. Bouyer, “Community Detection using a New Node Scoring and Synchronous Label Updating of Boundary Nodes in Social Networks,” *J. AI Data Min.*, vol. 8, no. 2, pp. 201–212, 2020.

[29] Y. Wang, G. Cong, G. Song, and K. Xie, “Community-based Greedy Algorithm for Mining top-K Influential Nodes in Mobile Social Networks,” in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010, pp. 1039–1048.

[30] H. Li, S. S. Bhowmick, A. Sun, and J. Cui, “Conformity-aware influence maximization in online social networks,” *VLDB J.*, vol. 24, no. 1, pp. 117–141, Jan. 2015.

[31] A. Bozorgi, H. Haghighi, M. Sadegh Zahedi, and M. Rezvani, “INCIM: A community-based algorithm for influence maximization problem under the linear threshold model,” *Inf. Process. Manag.*, vol. 52, no. 6, pp. 1188–1199, Nov. 2016.

[32] Y. Y. Ko, K. J. Cho, and S. W. Kim, “Efficient and effective influence maximization in social networks: A hybrid-approach,” *Inf. Sci. (Ny).*, vol. 465, pp. 144–161, Oct. 2018.

[33] J. Shang, H. Wu, S. Zhou, J. Zhong, Y. Feng, and B. Qiang, “IMPC: Influence maximization based on multi-neighbor potential in community networks,” *Phys. A Stat. Mech. its Appl.*, vol. 512, pp. 1085–1103, Dec. 2018.

[34] S. Jendoubi, A. Martin, L. Liétard, H. Ben Hadji, and B. Ben Yaghlane, “Two evidential data based models for influence maximization in Twitter,”

Knowledge-Based Syst., vol. 121, pp. 58–70, Apr. 2017.

[35] G. Corneuejols, M. L. Fisher, and G. L. Nemhauser, “Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms.,” *Manage. Sci.*, vol. 23, no. 8, pp. 789–810, Apr. 1977.

[36] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, “An analysis of approximations for maximizing submodular set functions-I,” *Math. Program.*, vol. 14, no. 1, pp. 265–294, Dec. 1978.

[37] R. A. Rossi and N. K. Ahmed, “The Network Data Repository with Interactive Graph Analytics and Visualization,” in AAAI, 2015. [Online]. Available: https://networkrepository.com/. [Accessed: 25-Oct-2020].

[38] J. Leskovec and A. Krevl, “SNAP Datasets,” Stanford, Jun-2014. [Online]. Available: http://snap.stanford.edu/data. [Accessed: 25-Oct-2020].

بهینه‌سازی الگوریتم CELF برای مسئله بیشینه‌سازی تأثیر در شبکه‌های اجتماعی

محسن‌طاهری‌نیا[۱]، مهدی‌اسماعیلی[۱،*] و بهروز مینایی‌بیدگلی[۲]

[۱] گروه کامپیوتر، دانشگاه آزاد اسلامی واحد کاشان، کاشان، ایران.

[۲] دانشکده مهندسی کامپیوتر ، دانشگاه علم و صنعت ایران، تهران، ایران.

ارسال ۲۰۲۱/۰۸/۰۹؛ بازنگری ۲۰۲۱/۰۹/۲۲؛ پذیرش ۲۰۲۱/۱۱/۰۲

چکیده:

هدف مسئله‌ی بیشینه‌سازی تأثیر در شبکه‌های اجتماعی یافتن مجموعه‌ی کمینه‌ای از افراد یک شبکه اجتماعی است بطوریکه بیشترین تأثیر را بر سایر افراد در شبکه داشته باشند. در دو دهه گذشته، الگوریتم‌های زیادی برای حل چالش‌های کارایی زمانی و اثربخشی این مسئله‌ی NP-Hard پیشنهاد شده‌است. بدون شک در میان آنها، الگوریتم CELF در کنار الگوریتم حریصانه دارای بالاترین میزان اثربخشی بوده است. البته، الگوریتم CELF کارایی زمانی در حدود ۷۰۰ مرتبه سریعتر از الگوریتم حریصانه از خود نشان داده است. این برتری باعث شده است که بسیاری از محققان از الگوریتم CELF در رویکردهای نوآورانه خود استفاده کنند. با این حال معضل اصلی الگوریتم CELF، زمان بسیار طولانی اولین تکرار آن است. زیرا مانند الگوریتم حریصانه، مجبور به اجرای شبیه‌سازی‌های سنگین مونت کارلو به منظور تخمین میزان تأثیر همه گره‌ها است. در این مقاله، یک رویکرد ابتکاری به نام optimized-CELF، برای بهبود این معضل الگوریتم CELF (با اجتناب از شبیه‌سازی های غیرضروری مونت کارلو) پیشنهاد شده‌است. الگوریتم پیشنهادی زمان اجرای الگوریتم CELF را به شدت کاهش داده و متعاقباً بازده زمانی الگوریتم‌های دیگر را که از CELF به عنوان الگوریتم پایه استفاده می‌کنند، بهبود می‌بخشد. نتایج تجربی بر روی طیف وسیعی از مجموعه‌داده‌های واقعی نشان دادند که الگوریتم optimized-CELF در مقایسه با الگوریتم CELF بدون از دست‌دادن میزان اثربخشی، به بهره زمانی بهتری درحدود ۸۸ تا ۹۹ درصد برای k=1 و ۵۹ تا ۹۸ درصد برای k=50 دست می‌یابد.

کلمات کلیدی: بهینه‌سازی CELF، بیشینه‌سازی تأثیر، تحلیل شبکه‌های اجتماعی، الگوریتم حریصانه، مدل انتشار.