

FastCELF++: A Novel and Fast Heuristic for Influence Maximization in Complex Networks

Vitor Elisiário do Carmo, Vinícius da Fonseca Vieira, Rafael Sachetto Oliveira, and Carolina Ribeiro Xavier(✉)

Universidade Federal de São João del Rei, São João del Rei, Brazil
carolinaxavier@ufsj.edu.br

Abstract. Social networks reflect the relationships and interactions between individuals and have played a significant role in the spread of information, in which the communication of ideas and sharing of opinions happen all the time. There are various examples of how social networks can affect the behavior of individuals, like viral marketing, the spread of memes, and the propagation of fake news. This dynamic of information diffusion has motivated the research of several approaches to identify the prominent influencers in a network. The Influence Maximization Problem consists of identifying a subset S, called a seed set, of at most k elements to achieve the maximum (expected) propagation through a diffusion model, with S as the initial influencers on a network. It demonstrated that the influence maximization problem is an NP-hard optimization problem. Therefore, it is unfeasible to identify the subset S that ensures the most extensive diffusion due to its complexity. The most typical approach to this problem is using approximate algorithms, highlighting the Cost-Effective Lazy Forward (CELF), about 700 times faster than the greedy strategy proposed by Kemp et al., and CELF++, which presents a runtime gain of between 35 to 55% over CELF. This work modifies the two above-mentioned state-of-the-art algorithms, CELF and CELF++, replacing Monte Carlo simulations with functions to calculate the diffusion estimations (metamodels) for selecting a set of seeds. The adoption of well-known methods in the literature with metamodels can identify orders of magnitude faster, more influential individuals and, in some cases, even outperform the results of these methods in terms of propagation.

Keywords: Influence maximization · social networks · CELF · CELF++ · optimization

1 Introduction

The dynamics of information diffusion in networks have motivated the study of several approaches to identify the prominent influencers in a social network since the diffusion of such information can affect society and bring significant

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
O. Gervasi et al. (Eds.): ICCSA 2023, LNCS 13956, pp. 453–470, 2023.
https://doi.org/10.1007/978-3-031-36805-9_30

consequences, both good and bad. Considering viral marketing, for example, a company pays people to promote its products and services online to reach an audience that is often unaware of its brand. The company wants to make an advertising campaign as profitable as possible, and the choice of who will advertise it may directly impact the success of this campaign. Ideally, those selected to carry out the outreach are highly influential people.

This problem may be posed as Influence Maximization Problem (IMP) and has applications in several domains, in addition to those previously mentioned, such as personalized recommendations [16], classification of news feeds [7], selection of influential profiles in *Twitter* [1], and real-time ad targeting [13].

It has already demonstrated that the IMP is an NP-hard optimization problem [9]; therefore, due to its complexity, it is not feasible to find the subset S that guarantees the maximum diffusion in viable computational time. Thus, heuristic approaches are widely used, such as the greedy strategy (GK) proposed by [9].

In one of the most notable works on the subject, Leskovec *et al.* [10] propose the *Cost-Effective Lazy Forward* (CELF), around 700 times faster than the *GK*, for exploiting the property of propagation function submodularity for diffusion models. This algorithm selects a node with the greatest marginal gain in each iteration. The marginal gain of a selected node in each iteration must be greater than the previous ones. This algorithm manages to reduce the number of calls in the influence propagation evaluation.

Goyal *et al.* [6] propose CELF++, which tries to improve the CELF strategy by performing the influence propagation calculation for two steps of the algorithm simultaneously, but at an exceptionally high execution time. However, their experiments have shown gains in runtime between 35 to 55% on CELF.

Inspired by the two well-known algorithms, *Cost-Effective Lazy Forward* (CELF) [10] and CELF++ [6], this work presents a modification of them, using two metamodels named *Expected Diffusion Value* (EDV) [5] and *Local Influence Estimation* (LIE) [8] to replace the Monte Carlo simulations, aiming to reduce the computational cost and the execution time of the classical algorithms.

The results found by the experiments performed considering the methodology presented in this work show that the main methods in the literature combined with meta-models can identify the potential set of the most influential individuals in a much lower execution time than the original methods and even outperform the results of those methods in terms of the number of influenced individuals.

2 Background

2.1 Heuristics for the Solution of the IMP

One of the most effective heuristics for the estimation of influential nodes in networks for the Influence Maximization Problem (IMP) is the *DegreeDiscount*, proposed by Chen *et al.* [3] and designed originally for the Information Cascade (IC) model. For a specific diffusion probability p, *DegreeDiscount* tries to perform a deeper analysis of the local structure of the vertices. The general idea of the algorithm is that nodes with high degrees are good seeds for the IMP, but the

edges that connect to it will not take into account for the calculation of the degrees of its neighbors, i.e., after considering a node as a seed, the evaluation of the next nodes will discount the edges that connect them to the nodes already in the seed set.

In addition to greedy strategies, it is possible to find in the literature several approaches that try to find suitable solutions for IMP through meta-heuristics. For example, Jiang *et al.* [8] propose an approach based on *Simulated Annealing* for the IMP, with the difference of replacing Monte Carlo simulations by a very efficient objective function, called Expected Diffusion Value (EDV). EDV obtains a diffusion estimate by performing a simple calculation that considers the diffusion probability and the number of neighbors within a hop of the evaluated nodes. Following the same approach, Gong *et al.* [5] use a Discrete Particle Swarm algorithm combined with an objective function called *Local Influence Estimation* (LIE), which extends the EDV to consider the neighborhood of two-hop distances making it more sensitive to the obtained estimate. In order to simplify the reading, the term "metamodel" will be further used to refer to the diffusion estimation functions, such as EDV and LIE.

2.2 Calculation of the Propagation Estimate

According to [3,4] it is computationally difficult to calculate the exact value of σ, the number of active nodes, for an IC model, thus making it necessary to estimate this value. One way to calculate this estimate is to average the number of vertices activated over several IC runs. Let $\mathcal{A}$ be the set of active vertices at the end of an IC execution. As IC is a probabilistic model, it can obtain different active vertices set at each execution. So $|\mathcal{A}|$ is a random variable, with an expected value $\sigma_{IC}(S)$. Knowing this, one way to achieve greater precision on $\sigma_{IC}(S)$ is to extract the mean result of all runs of the IC. This type of estimate is known as the Monte Carlo Method, a tool to estimate values through sampling and simulation [14]. More generally, the diffusion estimate of a S set in a m diffusion model is obtained by the Eq. 1:

$$\sigma_m(S) = \frac{1}{\mathcal{R}} \sum_{i}^{\mathcal{R}} |\mathcal{A}_i|, \tag{1}$$

where r is the number of executions of the model m and $\mathcal{A}_i$ is the set of nodes activated in the diffusion process in the i^{-th} execution. Traditionally, it is necessary to carry out a large number of simulations to obtain a good accuracy in the estimate and usually the value of $\mathcal{R} = 10000$ [3,6,10].

2.3 Expected Diffusion Value (EDV)

The Expected Diffusion Value (EDV) is a local approximation method proposed by Jiang *et al.* [8] for the influence propagation in the IC model, and aims at an approximation of the diffusion calculated by Monte Carlo simulations.

Furthermore, in the same work where EDV is proposes, it is used as an objective function for a method based on *Simulated Annealing* (SAEDV) adapted for the IMP and has shown to be able to obtain good results with millions of iterations. Let G be a graph $G = (V, E)$ and S a set $S \subseteq V$ of size k. The EDV is calculated by the Eq. 2

$$EDV(S) = k + \sum_{i \in N_S^{(1)} \setminus S} (1 - (1-p)^{\tau(i)}), \tag{2}$$

where $N_S^{(1)}$ represents the direct neighbors of S, $i \in N_S^{(1)} \setminus S$ represents the neighbor vertices of i that are not part of S, p is the activation probability in the IC model and $\tau(i)$ represents the number of edges between the vertex i and the initial set of vertices S. When the vertex i has a high degree and many neighbors that do not belong to S, its contribution to the EDV is greater, because the chance of i activating some new vertex is greater and the term $(1 - (1-)p)^{\tau(i)})$, as observed in Eq. 2.

2.4 Local Influence Estimation (LIE)

Local Influence Estimation (LIE) [5] is also a local approximation method for influence propagation under the IC, but unlike the EDV which only considers the neighborhood one hop away from the vertex set, this method also covers the neighborhood two hops away. Like EDV, LIE was designed to be used as an objective function in a particle swarm optimization method, which achieved good results with an acceptable runtime under the IC, as proved by Tang *et al.* [17]. LIE can be described as an estimation function by Eq. 3

$$LIE(S) = k + (1 + \frac{1}{|N_S^{(1)} \setminus S|} \sum_{u \in N_S^{(2)} \setminus S} p_u^* d_u^*) \sum_{i \in N_S^{(1)} \setminus S} (1 - \prod_{(i,j) \in E, j \in S} (1 - p_{i,j}))), \tag{3}$$

where $N_S^{(1)}$ and $N_S^{(2)}$ represent the neighborhood of S at one and two hops away, respectively, p_u^* is the activation probability of a node u, which corresponds to the propagation probability p in the IC model or to the $\frac{1}{d_i^{(in)}}$ for the *Weighted Cascade* template, d_u^* represents the number of edges between the 1 and 2 jump neighborhoods of the u vertex.

2.5 Rank-Biased Overlap (RBO)

The *Rank-Biased Overlap* (RBO) is a coefficient that calculates the similarity of two ordered sets proposed by Webber *et al.* [18] to work with ordinal and undefined rankings. Since it is a measure based on ordering and intersection, this method is suitable to analyze the formation of the seeds' sets for each criterion and its order. RBO respects a numerical range between 0 and 1, where larger values imply a more significant similarity between the lists. The principle of RBO is that elements with different ratings have different weights, giving higher

weights to those with higher ratings. For example, the RBO coefficient of two lists S and T is calculated by Eq. 4:

$$RBO(S,T,w) = (1-w)\sum_{d=1}^{\eta} w^{d-1} A(S,T,d), \tag{4}$$

where η is the number of elements in the ranking list and the parameter w determines how steep the decline in weights is - the smaller the w, the more weighted the metric. In Eq. 4, $A(S,T,d)$ is the overlap value between two ranking lists S and T up to position d in *ranking*, calculated by Eq. 5.

$$A(S,T,d) = \frac{|S_{1:d} \cap T_{1:d}|}{|S_{1:d} \cup T_{1:d}|}, \tag{5}$$

where $S_{1:d}$ are the elements present in positions from 1 to d of list S, and $T_{1:d}$ indicates the same elements of list T.

3 Proposed Methods

The diffusion estimate of CELF and CELF++, as initially described, is calculated by Monte Carlo simulations, in which a high number of repetitions are necessary to obtain satisfactory results. Typically 10,000 repetitions are used. In this work, we address this aspect to explore for improvement.

The main idea is straightforward, although very effective: to replace Monte Carlo simulations with a metamodel. The modified CELF and CELF++ methods were now FastCELF and FASTCELF++, respectively. The EDV and LIE metamodels replace the Monte Carlo simulations in each proposed algorithm in this work. Thus, we have four heuristics to be investigated: FastCELF EDV, FASTCELF++ EDV, FastCELF LIE and FASTCELF++ LIE.

3.1 Calculating Diffusion Estimates with Metamodels

Equation 1 presents the calculation of $\sigma_m(S)$, which corresponds to the diffusion estimate of a set S for a model m through Monte Carlo simulations. Knowing that calculating σ is the most computationally expensive part of the original algorithms and aiming to maintain the same structure, we propose to replace σ with a function γ.

The generic representation of the propagation estimate of a set by a metamodel is $\gamma_{\bar{m}}(S,\ p)$, where S is the set of seeds, $\bar{m}$ is a metamodel, and p is the propagation probability. Thus, the propagation estimates calculated by the EDV and LIE metamodels are written according to the respective Eqs. 6 and 7:

$$\gamma_{EDV}(S,\ p) = EDV(S) \tag{6}$$

$$\gamma_{LIE}(S,\ p) = LIE(S) \tag{7}$$

which are equivalent to the Eqs. 2 and 3 presented in Sect. 2.

3.2 FastCELF

A pseudo-code for FastCELF is presented in the Algorithm 1. It is essential to highlight that the suggested modifications in this work can be found in lines 4 and 15, where $\sigma_m(\{u\})$ is replaced by $\gamma(\{u\}, p)$, with γ being any metamodel and p the diffusion probability. As the metamodels considered in this work require the parameter p to calculate the diffusion estimate, it is also a parameter for FastCELF.

Algorithm 1. FastCELF

```
 1: function FastCELF(G, k, γ, p)
 2:     S ← ∅, Q ← ∅
 3:     for each u ∈ V do
 4:         u.mg ← γ({u}, p)
 5:         u.flag ← 0
 6:         Q ← Q ∪ {u}
 7:     end for
 8:     heapify(Q)                                   ▷ based on u.mg
 9:     while |S| < k do
10:         u ← Q[0]
11:         if u.flag = |S| then
12:             S ← S ∪ {u}
13:             Q ← Q − {u}
14:         else
15:             u.mg ← γ(S ∪ {u}, p) − γ(S, p)
16:             u.flag ← |S|
17:         end if
18:         heapify(Q)                               ▷ based on u.mg
19:     end while
20:     return S
21: end function
```

3.3 FASTCELF++

Similarly to FastCELF, FastCELF++ is also parameterized by p, that represents the propagation probability considered for the metamodels. In the pseudo-code illustrated by Algorithm 2, the modifications regarding CELF++ can be found in lines 5, 6, 22 and 24, corresponding to the calls to the function σ. As in FastCELF, $\sigma_m(\{u\})$ is replaced by $\gamma(\{u\}, p)$, which is a single metamodel call, instead of the traditional Monte Carlo simulations. $\Delta_u^{\gamma}(cur_best, p)$ corresponds to the marginal gain of u with respect to the set cur_best, calculated by the metamodel γ with diffusion probability p. In line 22 of Algorithm 2, for example, the term $u.mg1 \leftarrow \Delta_u^{\gamma}(S, p)$ could be rewritten analogously to line 15 of Algorithm 1, that is:

$$u.mg1 \leftarrow \Delta_u^{\gamma}(S, p) \equiv u.mg1 \leftarrow \gamma(S \cup \{u\}, p) - \gamma(S, p)$$

Algorithm 2. FastCELF++

```
1: function FastCELF++(G, k, γ, p)
2:     S ← ∅, Q ← ∅
3:     last_seed ← NULL, cur_best ← NULL
4:     for each u ∈ V do
5:         u.mg1 ← γ({u}, p)
6:         u.mg2 ← Δ_u^γ(cur_best, p)
7:         u.prev_best ← cur_best
8:         u.flag ← 0
9:         Q ← Q ∪ {u}
10:    end for
11:    heapify(Q)                                   ▷ based on u.mg1
12:    while |S| < k do
13:        u ← Q[0]
14:        if u.flag = |S| then
15:            S ← S ∪ {u}
16:            last_seed ← u
17:            cur_best ← NULL
18:            Q ← Q − {u}
19:        else if u.prev_best = last_seed and u.flag = |S| − 1 then
20:            u.mg1 ← u.mg2
21:        else
22:            u.mg1 ← Δ_u^γ(S, p)
23:            u.prev_best ← cur_best
24:            u.mg2 ← Δ_u^γ(S ∪ {cur_best}, p)
25:        end if
26:        u.flag = |S|
27:        update cur_best
28:        heapify(Q)                               ▷ based on u.mg1
29:    end while
30:    return S
31: end function
```

3.4 Complexity Analysis

The CELF and CELF++ algorithms have an identical time complexity for the IC model, $\mathcal{O}(k\mathcal{R}\mathcal{N}\mathcal{M})$ [2,12], where k is the number of seeds to be selected, $\mathcal{R}$ is the number of Monte Carlo simulations, and $\mathcal{N}$ and $\mathcal{M}$ correspond to the number of vertices and edges of the network, respectively. The worst-case scenario of the IC model occurs when all vertices of the network are activated during the cascade process since all edges are visited in this case, so its complexity is $\mathcal{O}(\mathcal{M})$.

The EDV presented in Sect. 2 has time complexity $\mathcal{O}(k\bar{D})$, with $\bar{D}$ as the average degree of the network. Since it covers a larger neighborhood, the worst-case scenario of LIE, presented in Sect. 2, regarding time complexity is $\mathcal{O}(k\bar{D}^2)$, where $\bar{D}$ is the average degree of the network.

The outer loop of FastCELF (Algorithm 1) in lines 3–7 has time complexity $\mathcal{O}(\mathcal{N}\gamma_{\bar{m}})$, where $\mathcal{O}(\gamma_{\bar{m}})$ is the complexity of the metamodel $\bar{m}$. The loop of lines

9–19 is executed $\mathcal{O}(k\mathcal{N})$ times, since the condition of line 11 can take up to $\mathcal{O}(\mathcal{N})$ iterations to occur. The calculation of the marginal gain in line 15 has complexity $2\mathcal{O}(\gamma_{\bar{m}})$. So the resulting complexity of FastCELF for any metamodel $\bar{m}$ is $\mathcal{O}(\mathcal{N}\gamma_{\bar{m}}) + 2\mathcal{O}(k\mathcal{N}\gamma_{\bar{m}}) \equiv \mathcal{O}(k\mathcal{N}\gamma_{\bar{m}})$.

In FastCELF++ (Algorithm 2) the main difference is in the number of calls of the estimative, calculated by the metamodel $\gamma_{\bar{m}}$. In the loop of lines 4–10, lines 5 and 6 have one and two calls to $\gamma_{\bar{m}}$, respectively, executed $\mathcal{O}(\mathcal{N})$ times, resulting at a complexity of $3\mathcal{O}(\mathcal{N}\gamma_{\bar{m}})$. In the second loop (lines 12–29) there are four calls to $\gamma_{\bar{m}}$, two on line 22 and two on line 24. Analogous to Algorithm 1, the loop is executed $\mathcal{O}(k\mathcal{N})$ times. As a result, the FastCELF++ complexity for any $\bar{m}$ metamodel is $3\mathcal{O}(\mathcal{N}\gamma_{\bar{m}}) + 4\mathcal{O}(k\mathcal{N}\gamma_{\bar{m}}) \equiv \mathcal{O}(k\mathcal{N}\gamma_{\bar{m}})$. For comparison purposes, Table 1 summarizes the complexities described.

Table 1. Time complexity of algorithms: k is number of seeds to be selected, $\mathcal{R}$ is number of Monte Carlo simulations, $\mathcal{N}$ is number of vertices, $\mathcal{M}$ is the number of edges and $\gamma_{\bar{m}}$ represents any metamodel.

Algorithm	Time Complexity
Algorithm: CELF	$\mathcal{O}(k\mathcal{R}\mathcal{N}\mathcal{M})$
Algorithm: CELF++	$\mathcal{O}(k\mathcal{R}\mathcal{N}\mathcal{M})$
Algorithm 1: FastCELF	$\mathcal{O}(k\mathcal{N}\gamma_{\bar{m}})$
Algorithm 2: FastCELF++	$\mathcal{O}(k\mathcal{N}\gamma_{\bar{m}})$

Considering the EDV and LIE metamodels, described in Sect. 2, it is possible to replace $\gamma_{\bar{m}}$ by their respective complexities. Substituting the EDV complexity we have $\mathcal{O}(k\mathcal{N}k\bar{D}) \equiv \mathcal{O}(k^2\mathcal{N}\bar{D})$. For LIE, the resulting complexity is $\mathcal{O}(k\mathcal{N}k\bar{D}^2) \equiv \mathcal{O}(k^2\mathcal{N}\bar{D}^2)$. Although the proposed methods have quadratic complexity in k, the value of this term tends to be much smaller than $\mathcal{N}$ and D. Table 2 presents the time complexities of the heuristics with the metamodels presented in Sect. 2.

Table 2. Time complexity of algorithms with metamodels: k is the number of seeds to be selected, $\mathcal{N}$ is the number of vertices and $\bar{D}$ is the average degree of the network.

Algorithm	Time Complexity
Algorithm 1: FastCELF with EDV	$\mathcal{O}(k^2\mathcal{N}\bar{D})$
Algorithm 2: FastCELF++ with EDV	$\mathcal{O}(k^2\mathcal{N}\bar{D})$
Algorithm 1: FastCELF with LIE	$\mathcal{O}(k^2\mathcal{N}\bar{D}^2)$
Algorithm 2: FastCELF++ withcom LIE	$\mathcal{O}(k^2\mathcal{N}\bar{D}^2)$

4 Data and Experiments

4.1 Networks

The methodology proposed in this work was evaluated considering a varied set of real networks, ranging from different categories and contexts, like social networks, communication, co-authorship and collaboration networks. Table 3 describes the characteristics of this dataset. Each network was pre-processed in order to extract its giant component. Therefore, the values presented in Table 3 refer to the giant component of their respective networks.

Table 3. Characteristics of used networks: number of nodes ($|V|$); number of edges ($|E|$); medium degree ($\hat{k}$); average *clustering* coefficient ($\hat{c}$).

Network	Type	$\|V\|$	$\|E\|$	$\hat{k}$	$\hat{l}$	Category
netscience†	non-directed	379	914	4.82	0.741	Co-authoring
email†	non-targeted	1134	5451	9.62	0.220	Communication
soc-hamsterster*	non-targeted	2000	16097	16.09	0.539	Social
CA-GrQc*	non-targeted	4158	13422	6.45	0.556	Collaboration
CA-HepTh*	non-targeted	8638	25998	5.74	0.481	Collaboration
fb-pages-company†	non-targeted	14113	52310	7.38	0.239	Social
fb-pages-sport†	non-targeted	13866	86858	12.52	0.276	Social
CA-AstroPh*	non-targeted	17903	196972	22.00	0.632	Collaboration
CA-CondMat*	non-targeted	21363	91286	8.54	0.641	Collaboration
soc-epinions*	targeted	75877	508836	13.41	0.137	Social
soc-slashdot0902*	targeted	82168	870161	21.18	0.060	Social

The networks tagged with $*$ were collected from the database *SNAP*[1] [11] while the networks tagged with † were collected from the *Network Repository*[2] [15].

4.2 Experiments

In order to establish fair conditions for the experiments, a familiar environment was defined. The seed sets k range from 5 to 50, with increments of 5 for the calculation of influence propagation. To calculate the similarity between the pairs of seed sets, the maximum size of the sets was 50. *Independent Cascade* (IC) was adopted as the diffusion model. The activation probability p for both the IC and the metamodels was set to 0.01. In order to obtain a good approximation of the influence propagation by the IC, the number of Monte Carlo simulations adopted in this work was 10000 [3,6,10].

[1] http://snap.stanford.edu/data/index.html.
[2] http://networkrepository.com/.

The proposed algorithms FastCELF and FastCELF++ were evaluated using both EDV and LIE. Thus, this work has four novel methods capable of achieving competitive results in diffusion quality and executing in considerably lower time than the original algorithms. For comparison purposes, the DegreeDiscount algorithm was evaluated with the original algorithms, CELF and CELF++ and the proposed heuristics.

Each heuristic was used to select 50 seeds from each network and simulate the IC model with multiples of five seeds. Next, the similarities of the seeds were checked pairwise using a heatmap with the RBO to compare the sets, and finally, the diffusion process with the seeds was started. It was noteworthy that, in this diffusion process, Monte Carlo simulations are used for all methods, including the proposed ones, and the results were plotted as a line chart.

The final evaluation of the efficiency of all methods was done by the diffusion process (IC model), which uses Monte Carlo simulations to obtain good accuracy, as it is a stochastic process. So, the following figures show the results for the original way for simulation, with seeds selected by each method.

5 Results and Discussion

The execution times of the original algorithms for seed selection, CELF and CELF++, together with their respective modified versions, are presented in Fig. 1. The proposed methods (FastCELF and FastCELF++), when used together with EDV, were much faster, taking less than 5 s in the "slashdot0902" network, the largest network in the base, to select the seeds. The execution time was less than 1 s for the four heuristics in the smallest network. On the other hand, the relative inefficiency of CELF and CELF++ is notorious in a network with a few hundred vertices, taking about 2 min and 3 min, respectively. The execution time increases with the size of the networks, taking each of the two algorithms approximately 23 h to finish executing in the "fb-pages-company" network. For this network, it was impractical to execute both original CELF and CELF++.

The modified FastCELF and FastCELF++ versions that use LIE as an estimating function take slightly longer than the EDV versions in the experiments performed. It happens because the LIE considers a larger neighborhood than the EDV to calculate its estimate. However, even being less efficient in runtime, the modified versions considering LIE are still much faster than the original algorithms. Among the proposed methods, FastCELF++ LIE was the most time-consuming in the "fb-pages-company" network, taking 30 s to finish its execution, a substantial gain compared to 23 h for the original methods.

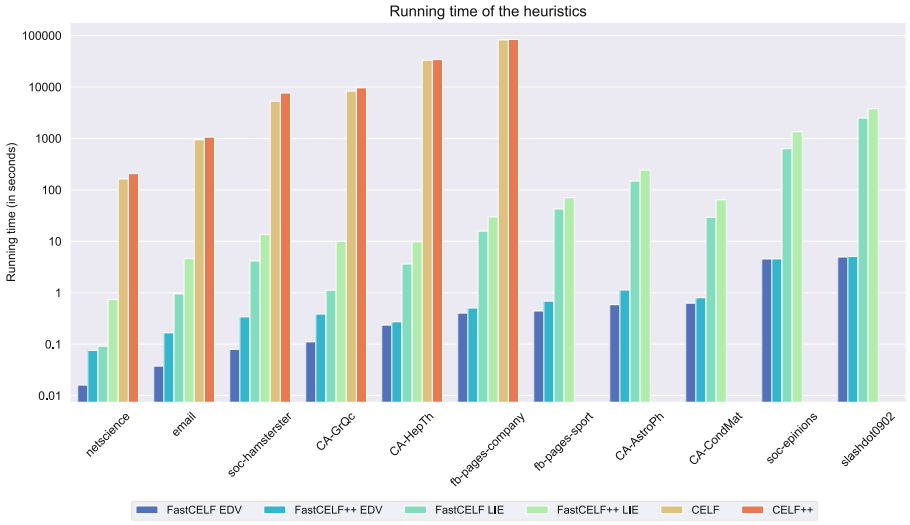

Fig. 1. Algorithm execution time for seed selection.

In terms of influence propagation, the results of experiments show that the proposed algorithms are competitive and outperform the original versions in all networks in experiments scenarios, even by a small margin. The Fig. 2 shows the results for netscience, email and soc-hamsterster.

Regarding the "email" network, the correlations between the seeds sets are high to very high, CELF and CELF++ chose the same set in the same order. One can see that all algorithms perform identically, varying the mean for their reached numbers in less than one unit.

In the "soc-hamsterster" network , the correlations are high, but even so it can be seen that the same three algorithms, *DegreeDiscount*, FastCELF++ LIE, FastCELF++ EDV show pairs of higher correlation. In this case FastCELF++ LIE outperforms *DegreeDiscount* by a small margin.

The first line of Fig. 3 shows the results of propagation considering the proposed heuristics and the correlation between their seeds' sets for the "CA-GrQc" network. *DegreeDiscount*, FastCELF++ LIE, FastCELF++ EDV methods, in that order, present the best results and outperform by far the CELF and CELF++ algorithms. Seed correlations show that high similarities reflect the performance of heuristics. CELF and CELF++ are similar to FastCELF algorithms that adopt EDV and LIE.

Diffusion results for "CA-HepTh" network can be seen in second line of Fig. 3. One can see that that FastCELF++ and *DegreeDiscount* outperformed other algorithms.

It is possible to observe in third line of Fig. 3 that for the "fb-pages-company" network, all algorithms work in a very similar way, which can be corroborated by

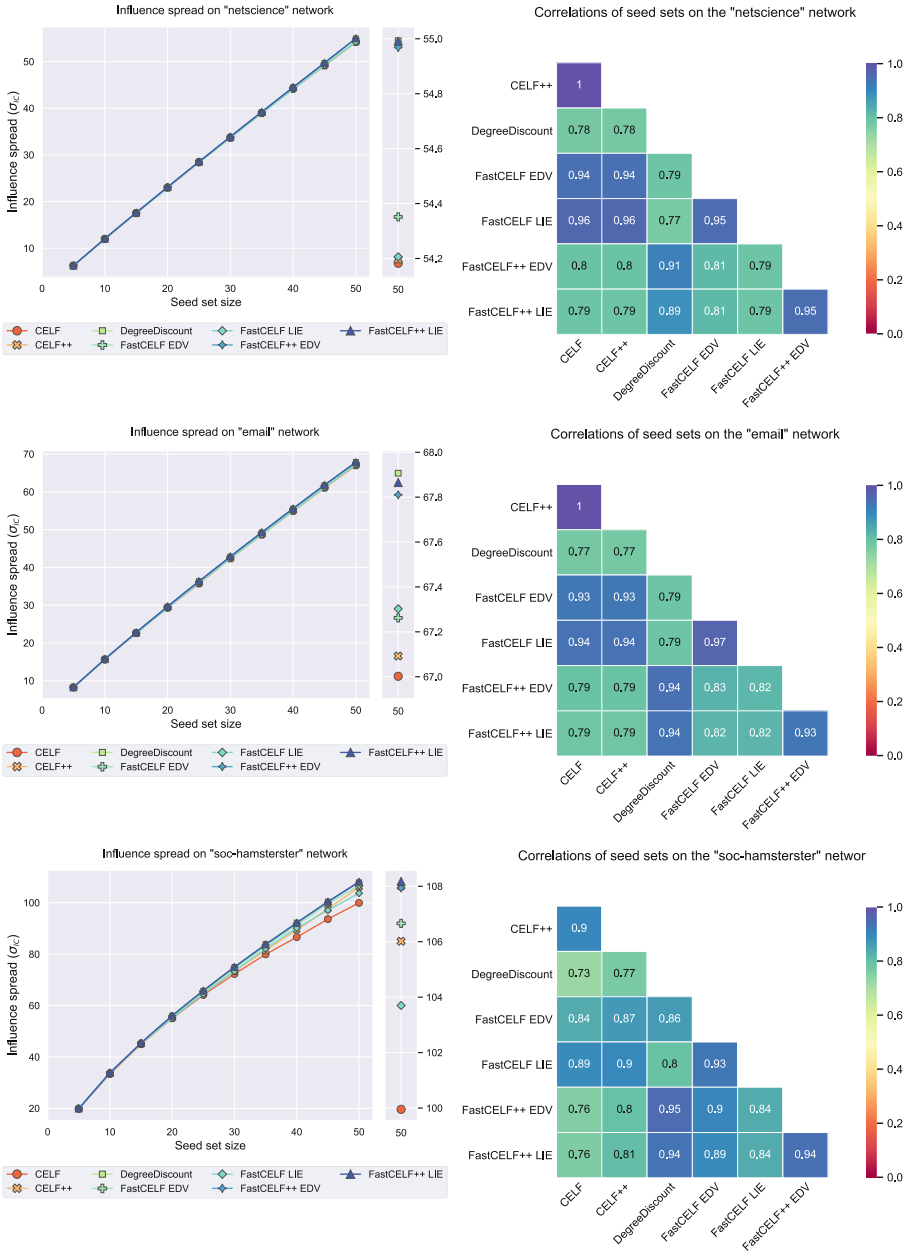

Fig. 2. Influence propagation and correlations of seed sets selected by heuristics

the correlation between their selected seed sets. A small margin can be observed for the Algorithms *DegreeDiscount*, FastCELF++EDV and FastCELF++LIE.

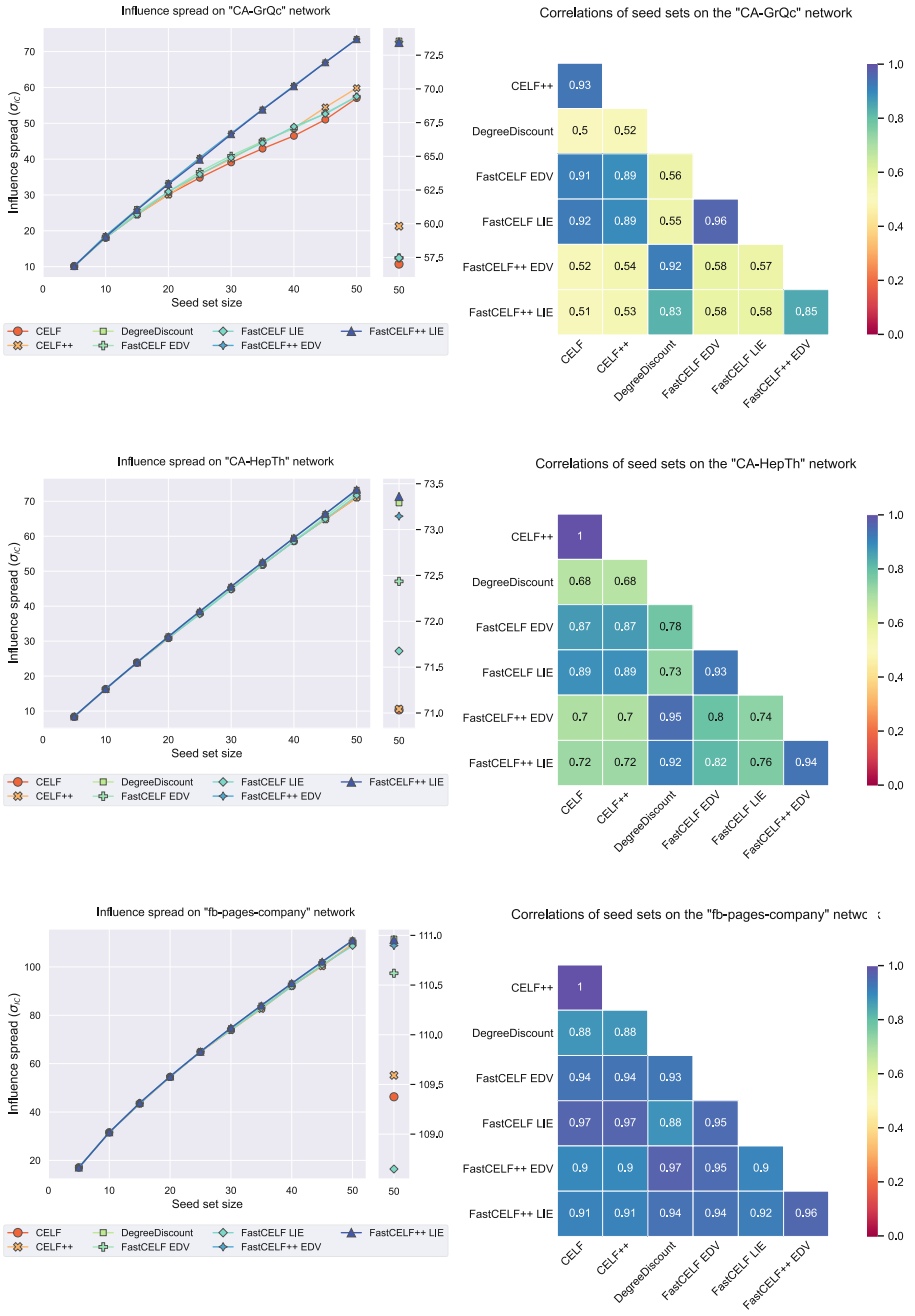

Fig. 3. Influence propagation and correlations of seed sets selected by heuristics.

For the network "fb-pages-sports", "CA-AstroPh" and "CA-condMat" the algorithms CELF and CELF++ can not be executed. The reason for that is the

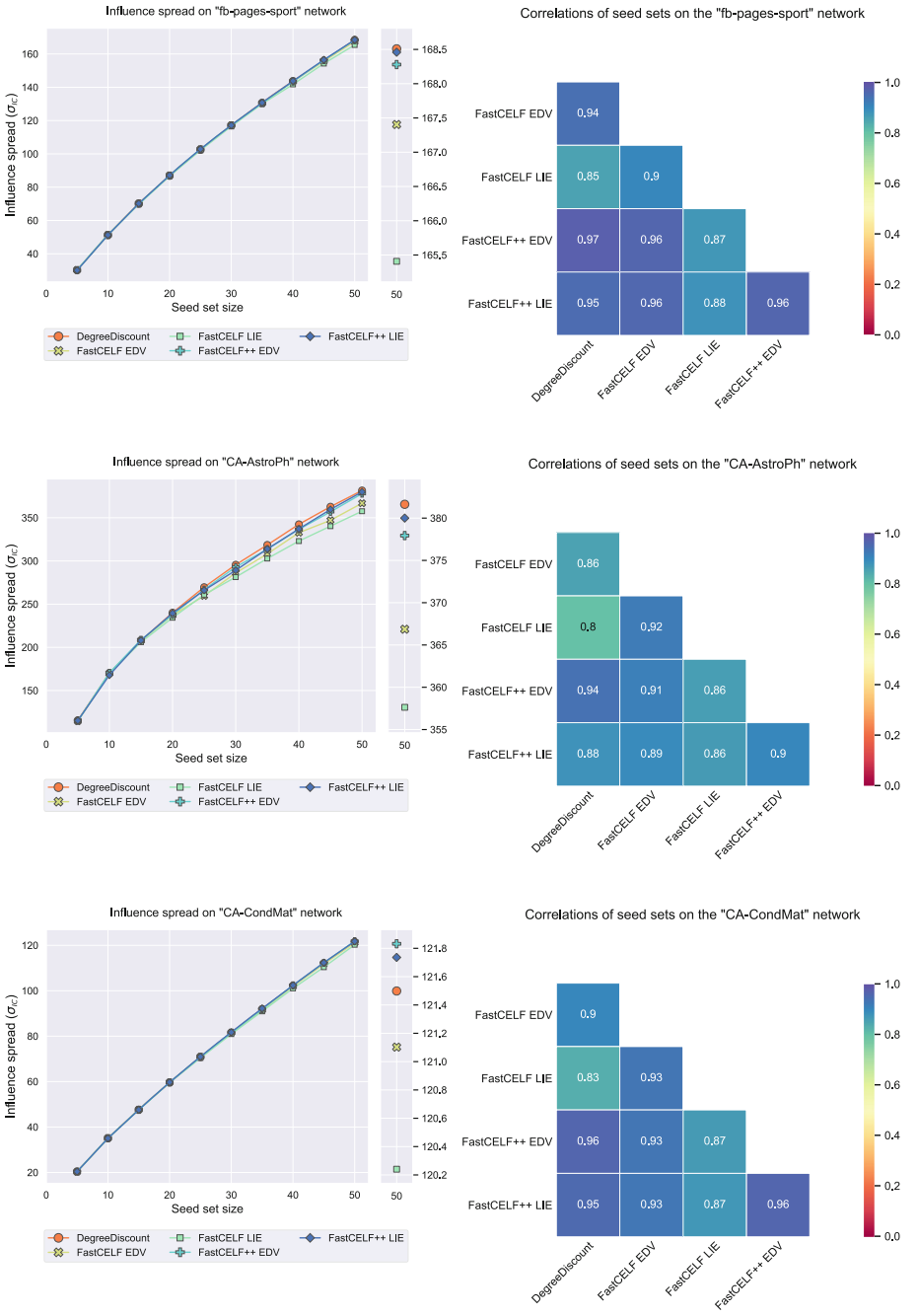

Fig. 4. Influence propagation and correlations of seed sets selected by heuristics.

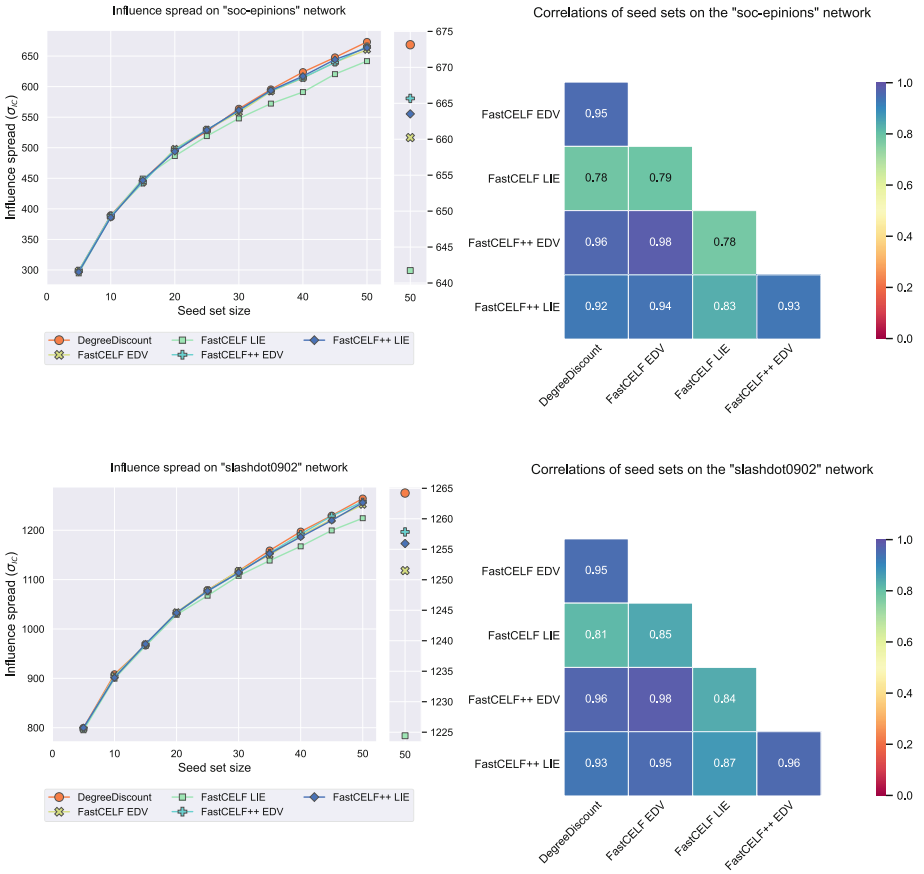

Fig. 5. Influence propagation and correlations of seed sets selected by heuristics

high computational time cost for these networks, which have more than 86000, 196000 and 91000 edges.

In Fig. 4, in first line, one can see that for the "fb-pages-sports" network, all algorithms work in a very similar way, which can be confirmed by the correlation between their selected seed sets. A small margin can be observed for the Algorithms *DegreeDiscount*, FastCELF++EDV and FastCELF++LIE.

In Fig. 4, in second line, one can see that for the "CA-AstroPh" there are a margin between Algorithms *DegreeDiscount*, FastCELF++EDV and FastCELF++LIE and the variations of FastCELF. The correlation between the selected seed sets of these algorithms is high.

In Fig. 4, in the last line, one can see that for the "CA-CondMat" network, all algorithms are almost identical, varying from 120.2 to 121.8, a small margin can be observed for the Algorithms FastCELF++EDV and FastCELF++LIE.

Furthermore, the heatmap presented the correlation between these seed sets, all from high to very high.

In Fig. 5 the diffusion results show that the *DegreeDiscount* was a little superior to the FastCELF++ networks for "soc-epinions" and "slashdot0902". The behavior of the Algorithms is very similar in these networks, showing the same rank for algorithms in two situations.

In Table 4 the summary of results for each network is presented. It is necessary to notice the superiority of the proposed algorithms based on CELF++, FASTCELF++EDV and FASTCELF++LIE and DegreeDiscount, used as a baseline. The best choice between the proposed algorithms for the tested networks was the FastCELF++LIE.

Table 4. Summary Results for each Networks and the three best algorithms for each one.

Network	1st Algorithm	2nd Algorithm	3rd Algorithm
netscience	FastCELF++LIE	DegreeDiscount	FastCELF++EDV
email	DegreeDiscount	FastCELF++LIE	FastCELF++EDV
soc-hamsterster	FastCELF++LIE	DegreeDiscount	FastCELF++EDV
CA-GrQc	FastCELF++LIE	DegreeDiscount	FastCELF++EDV
CA-HepTh	FastCELF++LIE	DegreeDiscount	FastCELF++EDV
fb-pages-company	FastCELF++LIE	DegreeDiscount	FastCELF++EDV
fb-pages-sport	DegreeDiscount	FastCELF++LIE	FastCELF++EDV
CA-AstroPh	DegreeDiscount	FastCELF++LIE	FastCELF++EDV
CA-CondMat	FastCELF++EDV	FastCELF++LIE	DegreeDiscount
soc-epinions	DegreeDiscount	FastCELF++EDV	FastCELF++LIE
soc-slashdot0902	DegreeDiscount	FastCELF++EDV	FastCELF++LIE

6 Conclusions and Future Works

The Influence Maximization Problem (IMP) in Complex Networks is an NP-hard problem. Therefore, it is necessary to use approximate strategies to guarantee a satisfactory solution with a possible execution time.

Many studies try to overcome the results achieved by the best-known greedy strategies in the literature. However, seed selection is the most critical task for the IMP problem, as it is directly related to the performance of diffusion models; therefore, creating new intelligent methods or improving existing strategies becomes essential.

Overall, the implemented heuristics proved to be comparable to *DegreeDiscount*, the FastCELF++, in different networks overcome in influence propagation, as using LIE or EDV. Therefore, the feasibility of replacing Monte Carlo simulations in the CELF and CELF++ algorithms with metamodels was shown, thus achieving a gain in propagation quality. The methodology developed

was sufficient to achieve the objectives proposed in this work. The correlations between pairs of seed sets with the RBO helped to understand the diffusion results and showed the similarity between the seeds selected by the classical methods and the seeds selected by the presented heuristics. Metamodels can quickly estimate the influence propagation and still maintain qualitatively comparable results to Monte Carlo simulations; as one can see in the RBO heatmaps, the seeds sets are very close. The developed methods have good efficiency and proved to be computationally scalable.

The results of this work are expected to contribute to the area, pointing to better strategies that present good performance, efficiency and speedup for the IMP problem.

Some directions can still be explored from this work: (1)The use of other diffusion models, as threshold model; (2) A sequence of the research carried out in this work is the variation of the activation probability p, indicating whether the metamodels are sensitive to this parameter; and, (3) One can develop a new metamodel that is quantitatively comparable to the propagation function calculation.

Acknowledgments. The authors thank CNPq and FAPEMIG for funding their projects.

References

1. Bakshy, E., Hofman, J.M., Mason, W.A., Watts, D.J.: Everyone's an influencer: quantifying influence on twitter. In: Proceedings of the Fourth ACM International Conference On Web Search and Data Mining, pp. 65–74 (2011)
2. Banerjee, S., Jenamani, M., Pratihar, D.K.: A survey on influence maximization in a social network. Knowl. Inf. Syst. **62**(9), 3417–3455 (2020). https://doi.org/10.1007/s10115-020-01461-4
3. Chen, W., Wang, Y., Yang, S.: Efficient influence maximization in social networks. In: Proceedings of the 15th ACM SIGKDD International Conference On Knowledge Discovery And Data Mining, pp. 199–208. ACM (2009)
4. Chen, W., Yuan, Y., Zhang, L.: Scalable influence maximization in social networks under the linear threshold model. In: 2010 IEEE International Conference On Data Mining, pp. 88–97. IEEE (2010)
5. Gong, M., Yan, J., Shen, B., Ma, L., Cai, Q.: Influence maximization in social networks based on discrete particle swarm optimization. Inf. Sci. **367**, 600–614 (2016)
6. Goyal, A., Lu, W., Lakshmanan, L.V.: Celf++ optimizing the greedy algorithm for influence maximization in social networks. In: Proceedings of the 20th International Conference Companion On World Wide Web, pp. 47–48 (2011)
7. Ienco, D., Bonchi, F., Castillo, C.: The meme ranking problem: Maximizing microblogging virality. In: 2010 IEEE International Conference on Data Mining Workshops, pp. 328–335. IEEE (2010)
8. Jiang, Q., Song, G., Gao, C., Wang, Y., Si, W., Xie, K.: Simulated annealing based influence maximization in social networks. In: Twenty-fifth AAAI Conference On Artificial Intelligence (2011)

9. Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the spread of influence through a social network. In: Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery And Data Mining, pp. 137–146. ACM (2003)
10. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., Glance, N.: Cost-effective outbreak detection in networks. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 420–429. ACM (2007)
11. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data (Jun 2014)
12. Li, Y., Fan, J., Wang, Y., Tan, K.L.: Influence maximization on social graphs: a survey. IEEE Trans. Knowl. Data Eng. **30**(10), 1852–1872 (2018)
13. Li, Y., Zhang, D., Tan, K.L.: Real-time targeted influence maximization for online advertisements (2015)
14. Mitzenmacher, M., Upfal, E.: Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis. Cambridge University Press (2017)
15. Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization. In: AAAI (2015). http://networkrepository.com
16. Song, X., Tseng, B.L., Lin, C.Y., Sun, M.T.: Personalized recommendation driven by information flow. In: Proceedings of the 29th Annual International ACM SIGIR Conference On Research And Development in Information Retrieval, pp. 509–516 (2006)
17. Tang, J., et al.: Maximizing the spread of influence via the collective intelligence of discrete bat algorithm. Knowl.-Based Syst. **160**, 88–103 (2018)
18. Webber, W., Moffat, A., Zobel, J.: A similarity measure for indefinite rankings. ACM Trans. Inform. Syst. (TOIS) **28**(4), 1–38 (2010)