Time Optimal Profit Maximization in a Social Network

Yong Liu, Wei Zhang
Hei Long Jiang University
China
liuyong001@hlju.edu.cn
149275442@qq.com

ABSTRACT: *Influence maximization aims to seek k nodes from a social network such that the expected number of activated nodes by these k nodes is maximized. However, influence maximization is different from profit maximization for a real marketing campaign. Moreover, we observe that when promotion time increases, the number of activated nodes tends to be stable eventually. In this paper, we first use real action log to propose a novel influence power allocation model with time span called IPA-T, and then present time optimal profit maximization problem called TOPM based on IPA-T. To address this problem, we propose an effective approximation algorithm called Profit-Max. Experimental results on real datasets verify the effectiveness and efficiency of Profit-Max.*

Keywords: Influence Maximization, Social Network, Profit Maximization, Action Log, Time Optimal

Received: 8 September 2017, Revised 12 October 2017, Accepted 18 October 2017

1. Introduction

In recent years, many large-scale social networks, such as Facebook, Twitter, Friendster, Microblog are becoming more and more popular. This is because that isolated users can be linked together to form a large virtual network by the online social network. New ideas and information can be spread to a large number of users in a short time through social networks.

Based on the ideas above, P. Domings et al. [1] first proposed the concept of viral marketing. We can choose a small set of influential users and give them free samples or discount products. Through the effect of word-of-mouth, a large number of users will buy the product eventually. D. Kempe et al. [2] formulated the problem as an influence maximization problem as follows. Given a directed graph $G = (V, E)$, a positive integer k, we aim to select k influential users called a set of seeds, so that by activating them, the expected number of users that are activated is maximized under a given propagation model. The literatures [3, 4, 5, 6, 7] have studied the influence maximization problem extensively and proposed a variety of solutions. However, the existing works usually assume that the influence spread is equal to the product profit and do not consider the cost of product promotion. In fact, as the promotional time increases, the promotional cost will increase continuously but the influence spread of the seed set will increase more and more slowly. The following experiments verify the view above. We use the algorithm in literature [8] to select top-10 influential nodes on a real data set Digg77461 (described in Section 5.2) as a seed set, and then select 7 time span with different length and observe the actual influence spread of the seed set w.r.t different time span. The experimental result is shown in Table 1, where *T* represents the length of time span and the unit length of *T* is a day, and *spread* represents the number of nodes that are activated actually.

$T(days)$	10	20	30	40	50	60	70
$Spread(nodes)$	165	240	282	295	301	305	308

Table 1. The *spread* vs different time span *T* in Digg77461 dataset

As shown in Table 1, the number of activated users increases rapidly at the initial stage of propagation. But as the propagation time increases, the growth rate of activated users becomes slow, and the number of activated users will become stable eventually. Moreover, we also discovery an interesting phenomenon. When the length of propagation time span is different, the most influential seed set is also different, which will be discussed in the experiments in Section 5.4. Accordingly, in the area of vital marketing, it is important to determine the optimal length of propagation time span and the corresponding seed set. It can help decision makers to make a reasonable marketing plan and thus achieve the maximum profit.

Consider the following scenario as a motivating example. Suppose that a company wants to market a new product through the social network. Promotion has a fixed cost per day and the new product has a fixed unit price. Because the number of activated users will become stable as the promotional time increases, the company wants to select the best promotional length and corresponding influential users in order to gain maximum profit. The application example above can be formalized as the following time optimal profit maximization problem (TOPM). Given the promotion cost of unit time, the unit price of one product, the action log such as history purchase records of users, we aim to choose the optimal time span *T* and the corresponding seed set *S*, such that by targeting *S*, we can achieve the maximum profit during the time span *T* but the profit will reduce when the promotion time is larger than *T*. Determining the optimal promotional time span *T* has important theoretical significance and practical application value. The contributions of this paper are summarized as follows.

• We use real action log to propose a new propagation model with time span which is called IPA-T propagation model, and then propose a new research problem, called time optimal profit maximization problem TOPM. We prove that the problem is NP-hard.

• In order to solve TOPM problem, we design an effective approximation algorithm Profit-Max, and show that the approximation ratio of Profit-Max algorithm is $(1 - 1/e - \beta/e)$, where *e* is the base of natural logarithm, and β represents the ratio between the cost of optimal solution and the profit of optimal solution.

• Experimental results on several real data sets show that Profit-Max can solve TOPM problem effectively and efficiently. In addition, an important and interesting finding is that the most influential seed set changes when the length of the promotion time changes. We use the framework for training and testing to further verify the prediction accuracy of Profit-Max.

2. Related Work

D. Kempe et al. [2] first proposed the influence maximization problem and presented a greedy algorithm to solve the problem and achieved a $(1 - 1/e)$ approximation ratio. The algorithm chooses *k* nodes iteratively and adds a node with the maximal spread margin to the current seed set until the size of seed set is *k*. However, the algorithm needs to run many Monte Carlo simulations to estimate the spread of seed set when choosing a node. Experimental results [2] show that when the number of nodes in a network is more than 1000 and the number of Monte Carlo simulation is set 10000, the greedy algorithm can't finish within a few hours. Clearly, the greedy algorithm is not suitable for large scale social networks. Several studies present various optimization strategies to improve the efficiency of the greedy algorithm. J. Leskovec et al. [3] proposed a CELF optimization strategy to improve the efficiency of the greedy algorithm and can achieve 700 times speedup than the original greedy algorithm. W. Chen et al. [4] proposed a heuristic algorithm Degree discount, which is two orders of magnitude faster than the greedy algorithm. Later, W. Chen et al. proposed a heuristic algorithm called PMIA under IC propagation model [5] and a heuristic algorithm called

LDAG under the LT model [6] to estimate the influence spread. Y. Wang et al. [7] presented a greedy algorithm based on community to solve influence maximization problem, and the algorithm both guarantees the approximation ratio and is several orders of magnitude to the greedy algorithm. W. Chen et al. [9] and B. Liu et al. [10] extended IC propagation model and proposed time constraint influence maximization problem. Because the affected people may not buy the product, L. Lu et al. [11] proposed profit maximization problem under LT-V model. He extended users' values into LT model and proposed LT-V model, thus simulate the process of product adoption. The algorithm chooses an optimal price vector and a seed set to achieve the purpose of profit maximization. S. Bhagat et al. [12] proposed product adoption maximization problem based on LT-C model and divided the state of nodes into three categories to further simulate the propagation of information. G. Li et al. [13] proposed location-aware influence maximization problem, which seek a seed set that maximize influence spread in certain region. G. Li et al. [13] proposed location-aware influence maximization problem, which seek a seed set that maximize the expected influence in certain region. Y Li et al. [14] proposed Keyword-Based Targeted influence maximization problem, which seek a seed set that maximizes the expected influence over users who are relevant to a given advertisement.

However, all of these methods have the following limitations. The influence probability on edges and activation threshold are set in advance, which can't simulate the spread of influence in the real world accurately. To this end, M. Kimura et al. [15] and A. Goyal et al. [16] used the observed data to predict the probability on the edge. A. Goyal et al. [8] proposed a data-based algorithm to solve the influence maximization problem. This algorithm directly leveraged available action logs of users to learn the process of influence propagation, proposed a CD model and developed an effective approximation algorithm based on CD model. However, we obtain an important discovery through the experiment on real data. When the length of time span is different, the most influential seed set is also different. Furthermore, in the real promotional environment, with the increase of time span, promotion cost will increase accordingly. Therefore, the company needs to select the optimal time span and corresponding seed set for real marketing campaign. The CD model did not consider the time factor, and thus achieve the same seed set for different time span, which is obviously unsuitable for real marketing campaign.

3. Problem

In Section 3.1, we introduce the influence power allocation model with time span called IPA-T. In Section 3.2, we give the formulation of time optimal profit maximization problem called TOPM.

3.1 IPA-T Model

An social network is modeled as a directed graph $G = (V, E)$, where V represents the users and E represents the relationships between users. The promotional time is denoted as a time span T. The action log L contains each action that is performed by each user in the social network. Each record in action log L is a tuple (u, a, t), which indicates user u performs action a at time t. In this paper, A stands for the universe of actions. $\forall u \in V; \forall a \in A$, we define a function $t(u, a)$ to denote the time that user u performs action a. We assume that each user performs an action at most once. If user u performs action a before user v, then $t(u, a) < t(v, a)$. If user u doesn't perform action a, its execution time is infinite, i.e. $t(u, a) = +\infty$. If $0 \leq t(v, a) - t(u, a) \leq T$ and $(u, v) \in E$, then we believe that action a is propagated from node u to node v in the time span T. In this case, we call that v is influenced by u, node v needs to allocate an influence power to u and ancestors of u as well.

The node set that influence node v directly in time span T w.r.t action a is denoted as $N_{in}(v, a, T) = \{u | (u, v) \in E, 0 \leq t(v, a) - t(u, a) \leq T\}$. Thus, if node v performs action a, then v assigns direct influence power to $\forall u \in N_{in}(v, a, T)$, which is denoted as $\alpha_{v,u}(a, T)$, and satisfies that $\sum_{u \in N_{in}(v,a,T)} \alpha_{v,u}(a, T) \leq 1$. We have several methods to define the value of direct influence power. For the sake of convenience, we set $\alpha_{v,u}(a, T) = 1/N_{in}(v, a, T)$, i.e., we give the equal value to each neighbor of node v. At the same time, node v assigns indirect influence power to each neighbor w of node u iteratively. If the time interval between node v and ancestor w is no more than time span T, i.e.,$0 \leq t(v, a) - t(w, a) \leq T$, then v assigns indirect influence power to w. For action a, the total influence power that node v assigns to its ancestor w in time span T is $\Gamma_{v,w}(a, T) = \sum_{u \in N_{in}(v,a,T)} \alpha_{v,u}(a, T) * \Gamma_{u,w}(a, T)$, where the base of the recursion is $\Gamma_{u,u}(a, T) = 1$. Similarly, for action a, the total influence power that node v assigns to a node set S in time span T is $\Gamma_{v,S}(a, T) = \sum_{u \in N_{in}(v,a,T)} \alpha_{v,u}(a, T) * \Gamma_{u,S}(a, T)$. If $u \in S$, then $\Gamma_{u,S}(a, T) = 1$. Finally, we aggregate all actions and compute the total influence.

In time span T, the total influence power that node v assigns to node w is defined as follows. $\Gamma_{v,w}(T) = \sum_{a \in P_v} \Gamma_{v,w}(a, T)/|P_v|$, where $|P_v|$ represents the number of actions that node *v* has performed. Similarly, for any set $S \in V$, in time span *T*, the total influence power that node *v* assigns to set *S* is defined as follows. $\Gamma_{v,S}(T) = \sum_{a \in P_v} \Gamma_{v,S}(a,T)/|P_v|$. The propagation model with the time constraints above is denoted as IPAT model. Thus, in time span *T*, the influence spread of set *S* is $\delta_{IPA-T}(S,T) = \sum_{v \in V} \Gamma_{v,S}(T)$ under IPA-T model. We give an example to further illustrate IPA-T model.

Example 1. Given a directed graph $G = (V, E)$, $V = \{1, 2, 3, 4, 5, 6\}$, $E = (1, 2), (1, 3), (1, 6), (3, 2), (3, 4), (3, 5), (4, 3), (5, 4)$ an action log $L = \{(1, a, 0), (2, a, 1), (6, a, 2), (3, a, 2), (4, a, 3), (5, a, 3)$. According to IPA-T model, we first compute the influence power of nodes in time span with length 1 and discover the most influential node. For tuple $(1, a, 0)$, node 1 first performs action *a* so that node 1 doesn't assign influence power to other nodes; For tuple $(2, a, 1)$, because we have $0 < t(2, a) - t(1, a) < 1$ and $(1, 2) \in E$, node 2 assigns influence power to node 1, which is $1/N_{in}(2, a, 1) = 1$; For tuple $(3, a, 3)$, despite $(1, 3) \in E$, but we have $t(3, a) - t(1, a) > 1$, thus node 3 doesn't assign influence value to node 1; Because of $(2, 3) \notin E$, node 3 doesn't assign influence power to node 2 as well. The rest tuples are processed in the similar way. In time span with length 2, we calculate the influence power of nodes in the same way, and the results are shown in Table 2.

$spread$	node1	node2	node3	node4	node5	node6
$T = 1$	1	0	2	0	0	0
$T = 2$	3	0	2	0	0	0

Table 2. The influence power in different time span

From Table 2, we observe that under the condition of the time span with different length, the most influential node is different. For example, when *T* is 1, the most important node is node 3, while *T* is 2, the most important node is node 1. As a result, we need to select different seed set based on different time span. As the promotion cost increases over time, it is desirable to find the optimal time span of promotion and corresponding seed set.

3.2 Problem Definition

Time Optimal Profit Maximization Problem (TOPM): Given a directed graph $G = (V, E)$, an action log *L*, the unit *price* of promotion product, the unit time *cost* of promotion, and a positive integer *k*, we aim to discover an optimal time span *T*, and a seed set *S* with size *k*. By targeting *S*, the net profit of products $price * \delta_{IPA-T}(S,T) - cost * T$ is maximized under IPA-T model, where $price * \delta_{IPA-T}(S,T)$ stands for the total sales in time span *T*, *cost* * *T* represents the total cost of promotion in time span *T*.

Theorem 1. The time optimal profit maximization problem TOPM is NP-hard.

Proof: If the time span *T* is set to the maximum time that users perform an action, i.e., $T = max\{t(u, a) \mid \forall u \in , a \in A\}$, IPA-T model is equal to CD model [8]. If we set *price* = 1 and *cost* = 0, TOPM is be equivalent to influence maximization problem under CD model. Because influence maximization problem under CD model, which is the special case of TOPM, is NP-hard [8], TOPM is NP-hard as well.

4. Profit-Maximization Algorithm

We firstly divide the entire time into *M* time units, and each time unit may be a day, a week or a month according to the actual situation. For each increasing length of time, we use algorithm Initialization to initialize the influence allocation list *Inflink*, then call algorithm Greedy-CELF to find current seed set *S_ temp* w.r.t. current time span and compute the *profit* of *S_ temp*. If the current *profit* is more than *max_ profit*, then *max_ profit* and optimal time span *T* are replaced by *profit* and current time span respectively. The pseudo-code of Profit-Max algorithm is shown in algorithm 1.

4.1 Scan Action Log and Initialize Inflink

Algorithm Initialization scans the action log *L* sequentially. For each action $a \in A$, we initializes the influence allocation list

Inflink [*a*][*u*] for each node *u* ∈ *V*. *Inflink* [*a*][*u*] stores the influence power of node *u* assigned to other nodes.

We preprocess the action log *L*(*u, a, t*) first by action and then time. *Active* stores the activated nodes in the current action. *P*[*u*] denotes the number of actions that user *u* has performed, *father*[*u*] stores the parents of user *u*, and *time*[*u*] denotes the time that user *u* performs the current action. We create the in degree adjacency list *inedge*[*u*] for each *u* ∈ *V* according to *G*. Then we scans the tuples of log *L*(*u, a, t*) sequentially. For each node *w* ∈ *in_edge*[*u*], if *w* ∈ *active* and the execution time of *u* in the current action *a* minus the execution time of *w* is no more than length *T*, then we add *w* to the influence allocation list *Inflink* [*a*][*u*] and allocate influence power to node *w*. Then we allocates influence powers to the parents of *w* iteratively. The pseudo-code of Initialization algorithm is shown in algorithm 2.

Algorithm 1: Profit-Max

```
Input: social network $G = (V, E)$, an action log $L$, the $price$ of
promotion product, the unit time $cost$ of promotion, and a
positive integer $k$
Output: the optimal time span $T$ and corresponding seed set $S$

 1: $max\_profit = 0$;
 2: divide the entire time into $M$ time units and let $t$ be the length of
    unit time;
 3: for $i = 1$ to $M$ do
 4:     $Inflink =$ Initialization$(G, L, i * t)$;
 5:     $S\_temp =$ Greedy-CELF$(Inflink, k)$;
 6:     $profit = price * \delta_{IPA-T}(Stemp, i * t) - cost * i * t$;
 7:     if $profit > maxp\_profit$ then
 8:         $maxp\_profit = profit$;
 9:         $S = S\_temp$;
10:         $T = i * t$;
11: Return $S$ and $T$;
```

Algorithm 2: Initialization

```
Input: $G = (V, E)$, the action log $L$, time span $T$
Output:the influence power allocation list $Inflink[a][u]$ for
each $a \in A, u \in V$
 1: for each $u \in V$ do
 2:     create in-degree adjacency list $in\_edge[u]$;
 3: for each action $a \in A$ do
 4:     $active \leftarrow \emptyset$;
 5:     for each $(u, a, t)$ do
 6:         $father[u] \leftarrow \emptyset$; $P[u] + +$;
 7:         for each $w \in (in\_edge[u] \bigcap active)$ do
 8:             if $(0 < time(u) - time(w) < T)$ then
 9:                 $father[u] \leftarrow father[u] \bigcup \{w\}$;
10:         for each $x \in father[u]$ do
11:             compute the direct influence power $\alpha_{u,x}$,add $x$ to
                $Inflink[a][u]$;
12:             for each $v \in Inflink[a][x]$ do
13:                 if $(0 < time(u) - time(v) < T)$ then
14:                     add $v$ to $Inflink[a][u]$;
15:         $active \leftarrow active \bigcup \{u\}$;
16: Return $Inflink$;
```

4.2 CELF Optimization to Find Influential Nodes

When the time span *T* is fixed, we use a greedy algorithm with CELF optimization to find seed set *S*. J. Leskovec et al [3] utilized the submodular property and proposed a CELF optimization algorithm. A nonnegative function $\Phi(.)$ is submodular, if for $\forall S_1 \subseteq S_2$ satisfying $\Phi(S_1 \bigcup u) - \Phi(S_1) \geq \Phi(S_2 \bigcup u) - \Phi(S_2)$. When the time span *T* is fixed, IPA-T model has the property of submodular, thus we can use the greedy algorithm with CELF optimization to select seed set.

The algorithm Greedy-CELF uses algorithm Margin-Compute to obtain margin gain *u:margin* for each $u \in V$, then updates the number of iteration of node *u* and inserts *u* into the priority queue *Q*. *Q* is kept in descending order according to the value of *margin*. Greedy-CELF greedily choose a node *u* with the maximal margin gain each time and put *u* into seed set *S* according to the following procedure, until seed set *S* contains *k* nodes.

In each iteration, the top node *u* of *Q* is analyzed. If the iteration number of *u* is the current iteration number, then we add *u* to seed set *S*, call algorithm Influence- Update to update the influence power allocation list of node *u* and the influence power given to set *S* by node *u*; else use algorithm Margin-Compute to calculate *u:margin*, renew the iteration number of node *u* and insert node *u* into the priority queue *Q*. The pseudo-code of Greedy-CELF algorithm is shown in algorithm 3.

Algorithm 3: Greedy-CELF

```
Input: $Inflink[a][u]$ for each $a \in A, u \in V$, a positive
number $k$.
Output:seed set $S$
 1: $SA \leftarrow \emptyset, S \leftarrow \emptyset, Q \leftarrow \emptyset$;
 2: for each $u \in V$ do
 3:     $u.margin$ =Margin-Compute($Inflink, SA, u$);
 4:     update the iteration number of node $u$ and inserts $u$ into the
        priority queue $Q$
 5: while ($|S| < k$) do
 6:     $u = pop(Q)$;
 7:     if (the iteration number of node $u$ is the current iteration)
        then
 8:         $S \leftarrow S \cup \{u\}$;
 9:         Influence-Update($Inflink, SA, u$);
10:     else
11:         $u.margin$ =Margin-Compute($Inflink, SA, u$);
12:         update the iteration number of node $u$ and inserts $u$ into
            the priority queue $Q$
13: Return $S$;
```

4.3 Aggregating All the Actions

The algorithm Margin-Compute calculates the margin gain when node *x* is added to the current seed set. Firstly, we aggregate the influence power of node *x* on action *a* and store this influence power into *gain*[*a*]. *SA*[*a*][*x*] represents the influence power which is assigned to the current seed set *S* by node *x* on action *a*. Thus the margin gain of adding *x* into seed set *S* for action *a* is *gain*[*a*] * (1 – *SA*[*a*][*x*]). Therefore, the total margin is the sum of *gain*[*a*] * (1 – *SA*[*a*][*x*]) for all actions. The pseudo-code of Margin-Compute algorithm is shown in algorithm 4.

4.4 Updating the Influence Power Allocation List

After node *x* is added into seed set *S*, the algorithm Influence-Update updates *Inflink* and the influence power of assigned to seed set *S* by other nodes. The pseudo-code of Influence-Update algorithm is shown in algorithm 5.

Algorithm 4: Margin-Compute

```
Input: $Inflink[a][u]$ for each $a \in A$ and $u \in V$, $SA$, node $x$.
Output: $x.margin$
1: $margin = 0$;
2: for each action $a$ do
3:     $gain[a] = 0$;
4:     for each $u \in V$ do
5:         if ($x \in Inflink[a][u]$) then
6:             $gain[a] = gain[a] + \Gamma_{u,x}(a)/P[u]$;
7:     $margin = margin + gain[a] * (1 - SA[a][x])$;
8: Return $margin$;
```

Algorithm 5: Influence-Update

```
Input: $Inflink[a][u]$ for each $a \in A$ and $u \in V$, $SA$, node $x$.
Output: $Inflink$ and $SA$
1: for each action $a$ do
2:     for each $u \in V$ do
3:         if ($x \in Inflink[a][u]$) then
4:             for each $v \in Inflink[a][x]$) do
5:                 delete the influence power assigned to $v$ through
                   node $x$;
6:         update $SA[a][u]$;
```

Complexity Analysis: The time complexity of Initialization algorithm is $O(|V| + |L| * \varphi)$, where $|V|$ is number of nodes, $|L|$ is the number of tuples in action log L, and φ is the maximum number of ancestors of all nodes in the social network w.r.t. all actions, i.e., $\varphi = max\{|Inflink[a][u]|, \forall a \in A, u \in V\}$. The time complexity of Greedy-CELF algorithm is $O((|V| + k) * |A| * \varphi)$, where $|A|$ is number of actions. Therefore the time complexity of Profit-Max algorithm is $O(M * (|L| + (|V| + k) * |A|) * \varphi)$, where M is number of time units.

Theorem 2. The approximation ratio of Profit-Max is $(1-1/e - \beta/e)$, where e is the base of natural logarithm, and β is the ratio between the cost of optimal solution (T_0, S_0) and the profit of optimal solution, i.e., $\beta = cost * T_0 / (price * \delta_{IPA-T}(S_0, T_0) - cost * T_0)$.

Proof: We select the optimal seed set under the condition of a fixed time span T each time. When T is a constant, according to Theorem 2 in literature [8], it is easily proved that $\delta_{IPA-T}(S, T)$ is submodular and monotone. When S is $\emptyset$, we have $\delta_{IPA-T}(S, T) = 0$. Therefore, when T is a constant, we can use the greedy algorithm to find an approximate solution within a factor of $(1-1/e)$. For the sake of convenience, $1-1/e$ is denoted as α. Let (T_0, S_0) be the optimal solution to TOPM, the seed set returned by Profit-Max is S' under the condition of $T = T_0$. The solution to TOPM returned by Profit-Max is (T_1, S_1). Then, we have $\alpha * \delta_{IPA-T}(S_0, T_0) < \delta_{IPA-T}(S', T_0)$. Thus we have $price * \delta_{IPA-T}(S_0, T_0) - cost * T_0 \leq \frac{1}{\alpha} * price * \delta_{IPA-T}(S', T_0) - cost * T_0 = \frac{1}{\alpha} * (price * \delta_{IPA-T}(S', T_0) - cost * T_0) + (\frac{1}{\alpha} - 1) * cost * T_0 \leq \frac{1}{\alpha} * (price * \delta_{IPA-T}(S_1, T_1) - cost * T_1) + (\frac{1}{\alpha} - 1) * cost * T_0$. Because $price * \delta_{IPA-T}(S_0, T_0) - cost * T_0 \leq \frac{1}{\alpha} * (price * \delta_{IPA-T}(S_1, T_1) - cost * T_1) + (\frac{1}{\alpha} - 1) * cost * T_0$, $1 \leq \frac{1}{\alpha} * \frac{(price * \delta_{IPA-T}(S_1, T_1) - cost * T_1)}{(price * \delta_{IPA-T}(S_0, T_0) - cost * T_0)} + (\frac{1}{\alpha} - 1) * \beta$. Hence, $\frac{price * \delta_{IPA-T}(S_1, T_1) - cost * T_1}{price * \delta_{IPA-T}(S_0, T_0) - cost * T_0} \geq (1 - (\frac{1}{\alpha} - 1) * \beta) * \alpha = \alpha - (1 - \alpha) * \beta = 1 - 1/e - \beta/e$.

5. Experimental Results and Analysis

We conduct a series of experiments on two different data sets to verify the effectiveness and efficiency of our algorithm from various aspects. Firstly, we study the effect of different parameters in Profit-Max algorithm; Secondly, we compare the proposed algorithm with other five methods to demonstrate the accuracy of Profit-Max algorithm.

5.1 Experiment Setup

We use two network datasets to evaluate the performance of our proposed algorithm. The first dataset is the music social network site Last.fm [17]. We extract 2100 nodes, 25435 edges and the corresponding action log. This log contains 1000 action and 21,646 records. A record (*u, a, t*) means that user *u* comments singer *a* at time *t*. If user *v* comments singers *a*, *v's* friend *u* comments singer *a* later, then we assume that action *a* propagates from user *v* to user *u*. We denote the first dataset as Last21646. The second dataset is the social news review site Digg[18]. We extract 5000 nodes, 395,513 edges and the corresponding action log from the raw network. The action log contains 500 action and 77,461 records. Each record (*u, a, t*) means that user *u* comments news *a* at time *t*. If user *v* comments news *a*, *v's* friend *u* comments news *a* later, then we assume that action *a* propagates from user *v* to user *u*. The second dataset is denoted as Digg77461. We also extract 1000 nodes, 20,985 edges and the corresponding action log from Digg network. The action log contains 500 action, 28,806 records. We denote it as Digg28806.

All algorithms are implemented in C++ language, compiled under the environment of Microsoft Visual Studio 2010. All experiments are run on a server with Intel (R) Core (TM) 2 Duo CPU, 2GB of main memory.

5.2 Comparing the Effects of Seed Size *k* and Time Span *T*

In this subsection, we study the effects of different seed size *k* and different time span *T* on *spread* and *profit*. As shown in Figure 1 and 3, when the time span *T* is fixed, the influence *spread* increases with the growth of *k*. When *k* is fixed, the influence *spread* increases with the growth of time span *T* until it becomes stable. As shown in Figure 2 and 4, when *k* is fixed, with the increase of time span *T*, *profit* begins to increase, after reaching a certain point, *profit* begins to decline. For example, on Last21646 dataset, we can obtain the maximal profit at time span $T = 6$, on Digg77461 dataset, we can obtain the maximal profit at time span $T = 3$. Thus, in order to maximize profit, it is important to choose the best promotional time span *T*.

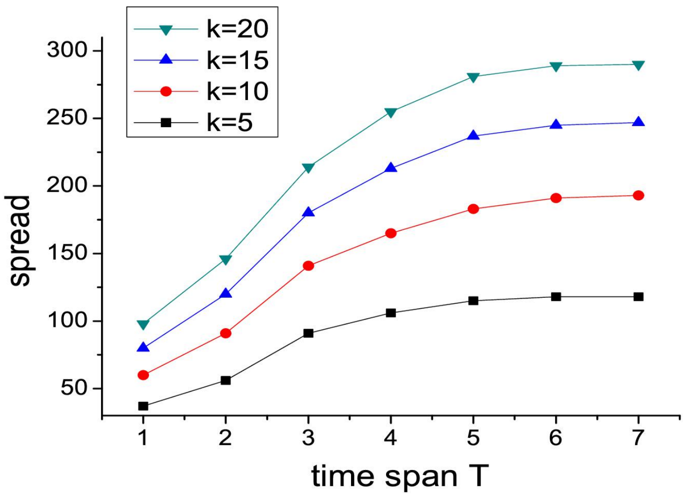

Figure 1. *Spread* vs *T* in Last21646 dataset

5.3 Comparing the Effects of Price and Cost

In this subsection, we study the effect of different *price* and *cost* on *profit* and optimal promotion time *T*. Figure 5 shows *profit* and optimal time span *T* with different *price* on Last21646. As shown in Figure 5, when the *price* of product increases, the *profit* and the best promotional time span *T* increases as well.

For example, when *price* is equal to 10, the best promotional time span *T* is 3, when *price* is equal to 40, the best promotional time span *T* is 6. Figure 6 shows *profit* and optimal time span *T* with different *cost* on Last21646. When the *cost* of product

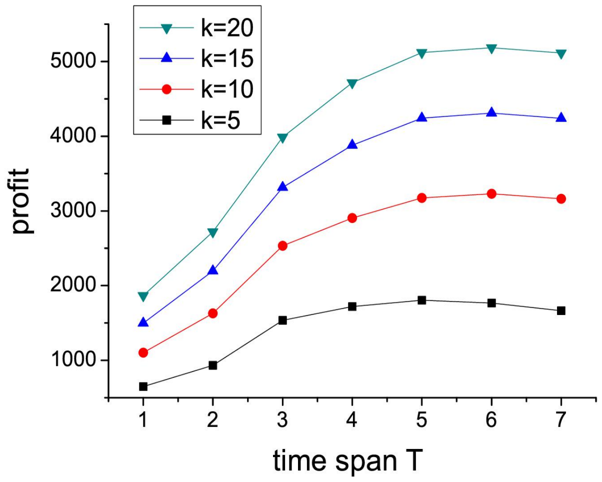

Figure 2. *Profit* vs *T* in Last21646 dataset, *price* = 20, *cost* = 100

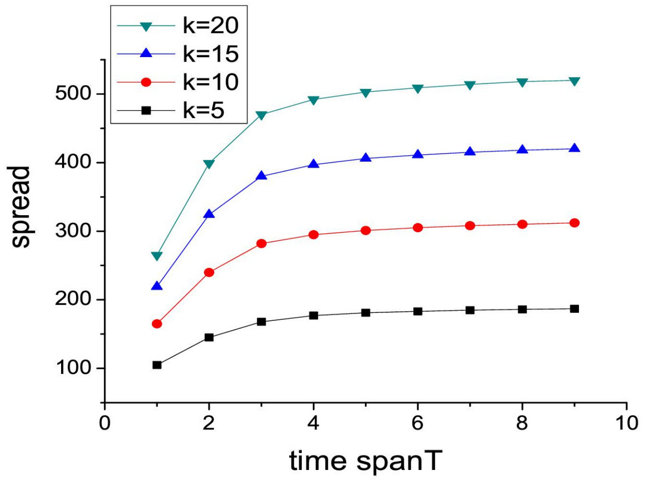

Figure 3. *Spread* vs *T* in Digg77461 dataset

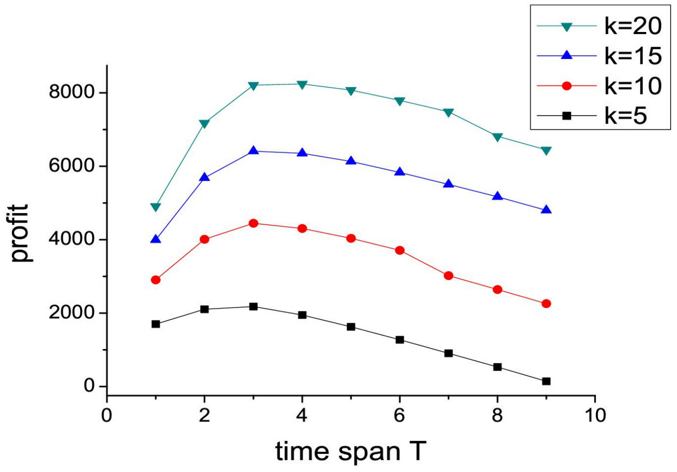

Figure 4. *Profit* vs *T* in Digg77461 dataset, *price* = 20, *cost* = 400

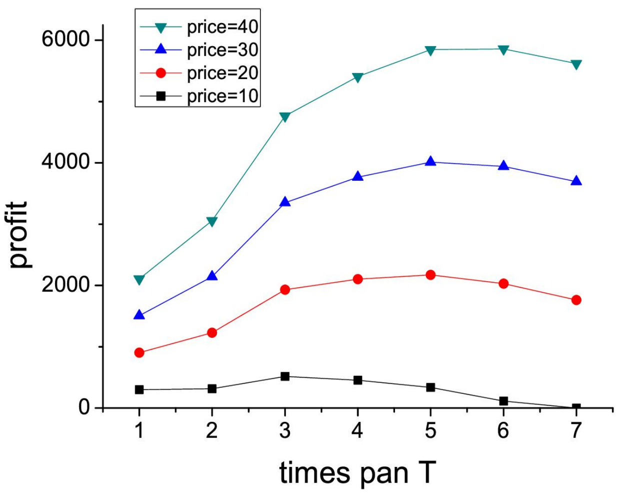

Figure 5. *Profit* vs *T* in Last21646 dataset, *cost* = 300, *k* = 10

increases, the best promotional time span *T* decreases. For example, when *cost* is 100, the best promotional time span *T* is 6; when *cost* is 400, the best promotional time span *T* is 4. From the experimental results, we can conclude that we need to select different promotional time span *T* for different *price* and *cost* in order to obtain maximum profit.

We also do similar experiments on Digg77461 and the results have the same tendency. Due to the limited space, we do not list here.

5.4 Comparing Profit-Max with Different Algorithms

In this subsection, we compare Profit-Max with the following algorithms in terms of the influence *spread* and running time. (1) Degree-Max: select seed set based on node's degree; (2) Greedy-Max: use Monte Carlo simulations to estimate the influence spread; (3) CD-Max: use greedy strategy to select seed set under CD model; (4) LAIC-Max: select seed set under LAIC model according to the length of time; (5) IC-MMax: select seed set under IC-M model according to the length of time.

We use the six algorithms described above to choose seed set with *k* = 10 and compute the influence *spread* under different length of time span on Last21646 dataset. Experimental results are shown in Figure 7. Degree-Max has the worst performance in the influence spread, which is due to that it only considers the topology of network to find the maximum degree of node, but the node with large degree is not the influential node. Greedy-Max uses Monte Carlo simulation to compute the influence spread, thus it is better than Degree-Max. LAIC-Max and IC-M-Max consider both the topology of network and time constraint, but they do not use real data, thus the influence spread is small, which demonstrates the importance of real data. CD-Max uses real data, but does not consider time factor, it selects the same seed set in different time span. Profit-Max uses real data and selects the seed set dynamically according to the length of time span. Therefore, Profit-Max has a better performance compared with CD-Max.

We also use the six methods to choose seed set with *k* = 10 on Digg28806 dataset and compare the influence spread. Experimental results are shown in Figure 8. In order to make the results obvious, we divided the whole time into 20 segments and only compare the former 10 segments. We can see that Profit-Max can achieve the maximal spread under different time span *T*, thus it is the most effective.

Figure 9 shows the running time of the six algorithms on Digg28806 dataset. The running time of LAIC-Max, IC-M-Max, Degree-Max, CD-Max, Greedy-Max include the time of selecting seed set and computing the influence spread on real data. Greedy-Max method requires Monte Carlo simulations to choose seed set, the running time is too long to finish in reasonable time. As shown in Figure 9, with the increase of time span *T*, the running time of LAIC-Max and IC-M-Max are linear growth. Profit-Max has the shortest running time, because it directly calculates the influence spread on real data. The other five algorithms need to choose seed set firstly, then calculate the influence spread under the condition of different time span *T*. Therefore Profit-Max is more efficient than the other five algorithms.

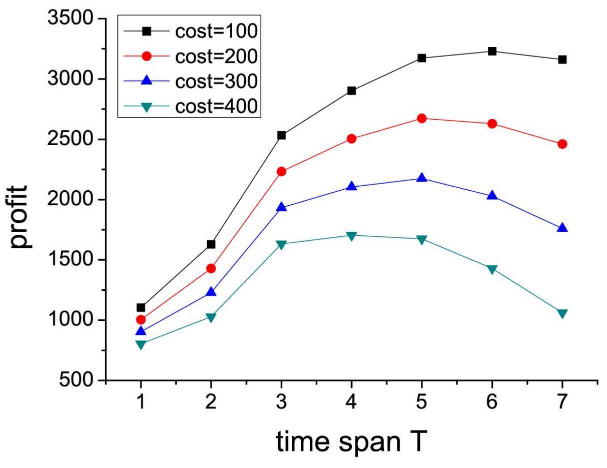

Figure 6. *Profit* vs *T* in Last21646 dataset, *price* = 20, *k* = 10

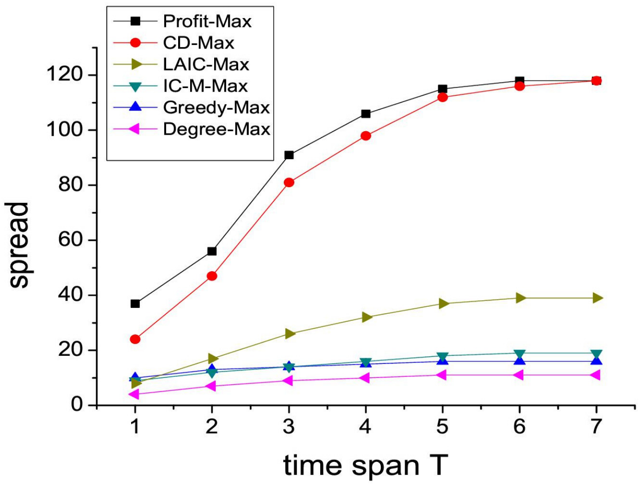

Figure 7. *Spread* vs *T* in Last21646 dataset with *k* = 10

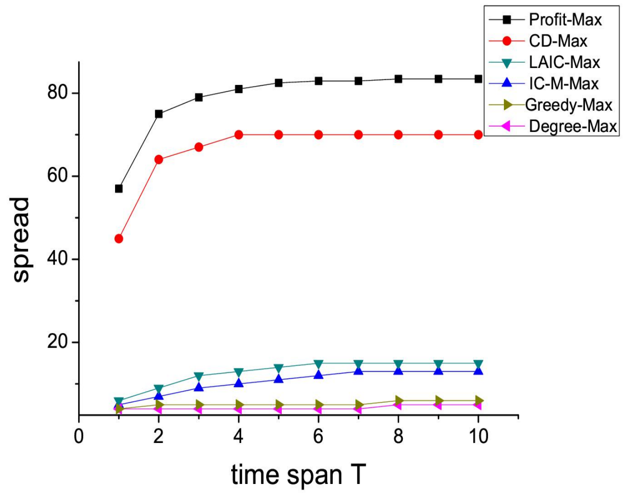

Figure 8. *Spread* vs *T* in Digg28806 dataset, *k* = 10

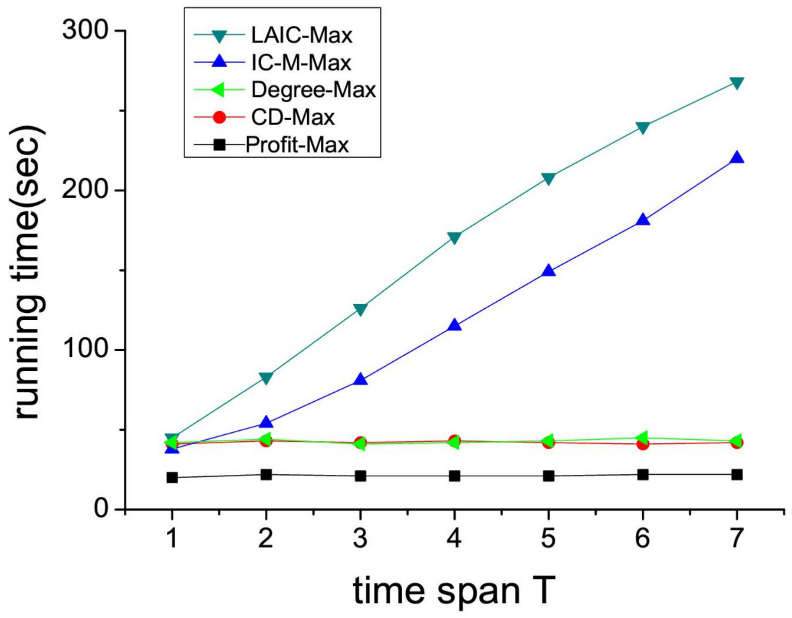

Figure 9. Running time vs T in Digg28806, $k = 10$

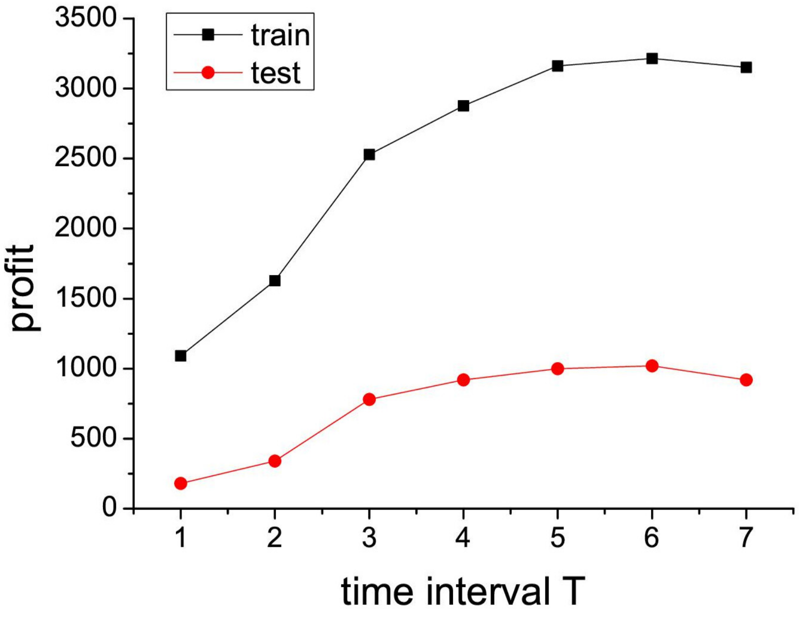

Figure 10. *Profit* vs T in Last21646 dataset, *price* = 20, *cost* = 100, $k = 10$

In the following experiments, we show that when the length of time span is different, we need to select different seed set. Table 3 shows the top-10 nodes obtained by Profit-Max w.r.t. different time span T. It is obvious that the seed set is different as well when time span T is different.

5.5 The Accuracy of Promotion Time and Seed Set

In this subsection, we firstly verify the accuracy of optimal time span T obtained by Profit-Max. We divide Last21646 dataset into a training dataset and a testing dataset in accordance with the ratio of 4:1. We use Profit-Max algorithm to find the optimal time span $T = 6$ and corresponding seed set $S = \{136, 1210, 545, 405, 147, 1283, 224, 1929, 1308, 1253\}$ on the training dataset and then calculate the profit achieved by seed set *S* w.r.t. different time span on the testing dataset. Experimental result is shown in Figure 10. Obviously, Profit-Max obtains the same optimal promotion time span *T* on the training and testing dataset, which validates the effectiveness of Profit-Max algorithm again.

Next, we evaluate the accuracy of seed set *S* selected by different algorithms. We compare Profit-Max with the other five algorithms. We select the different seed sets with $T = 6$ on the training dataset using different algorithms respectively, then calculate the profit achieved by different seed sets *S* with $T = 6$ on the testing dataset. Experimental result is shown in Figure 11. Because Greedy-Max and Degree-Max only consider network topology, they achieve the worst profit. LAIC-Max and IC-M-Max consider the topology of network and time factor, thus they obtain better profit than Greedy-Max and Degree-Max. CD-Max utilizes action log and thus achieves better profit than LAIC-Max and IC-M-Max. Besides action log, Profit- Max takes the time factor into consideration as well and thus achieves the best profit. From the experimental results, we conclude that Profit-Max achieves the best performance in terms of promotion time and profit.

$T = 1$	236,545,1253,1380,1832,58,1308,1283,753
$T = 2$	545,405,1210,236,1253,1283,2021,862,136,1023
$T = 3$	545,405,1210,136,1283,1253,1308,1423,1600,1606
$T = 4$	545,136,1210,405,1283,1308,1253,1423,224,147
$T = 5$	136,1210,545,405,1283,147,1308,224,1253,1929
$T = 6$	136,1210,545,405,147,1283,224,1308,1929,1253
$T = 7$	136,1210,545,405,147,1283,224,1308,1929,1253

Table 3. The *Spread* vs *T* in Digg77461 dataset

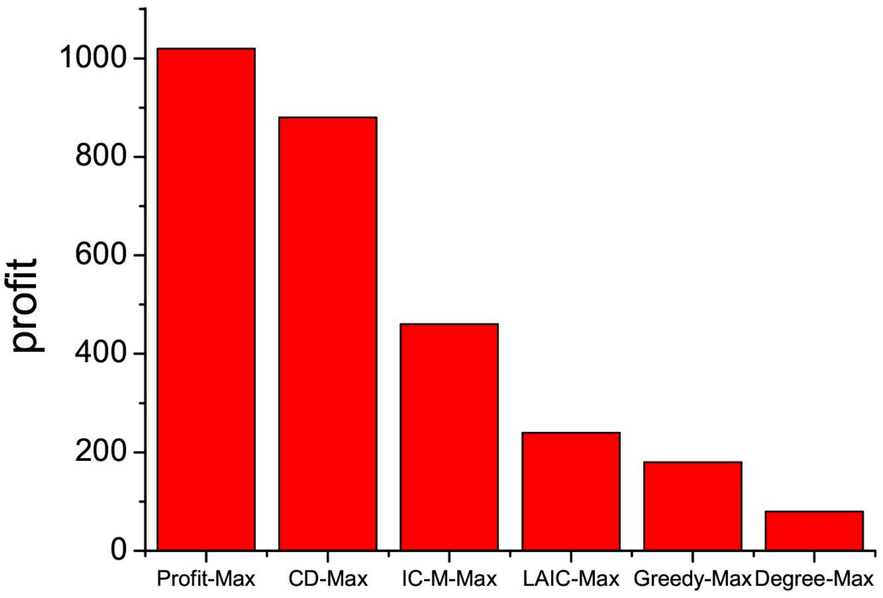

Figure 11. The *profit* of six algorithms in testing dataset, *price* = 20, *cost* = 100, *k* = 10

6. Conclusion

In this paper, we first propose an novel influence power allocation model with time span IPA-T. According to the model, we propose the time optimal profit maximization problem called TOPM. We prove that TOPM is NP-hard. We design an effective approximation algorithm Profit-Max for TOPM and show its approximation ratio. Extensive experiments demonstrate the effectiveness and efficiency of Profit-Max. Our future work intends to simulate the influence spread accurately according to action classification.

Acknowledgment

This work was supported by the Natural Science Foundation of China under grant of No. 61300225, and the Natural Science Foundation of Heilongjiang Province of China under grant of No. F201430.

References

[1] Domingos, P., Richardson, M. (2001). Mining the network value of customers, *In*: SIGKDD, p. 57–66.

[2] Kempe, J. M. K. D., Tardos., E. (2003). Maximizing the spread of influence through a social network, *In*: SIGKDD, p. 137–146.

[3] Leskovec, C. G. C. F. J. V. J., Krause, A., Glance., N. S. (2007). Costeffective outbreak detection in networks, *In*: SIGKDD, p. 420–429.

[4] Chen, Y. W. W., Yang., S. (2009). Efficient influence maximization in social networks, *In*: SIGKDD.

[5] Chen, C. W. W., Wang., Y. (2010). Scalable influence maximization for prevalent viral marketing in large scale social networks, *In*: SIGKDD.

[6] Chen, W. (2010). Scalable influence maximization in socialnetworks under the linear threshold model., *In*: ICDM, p. 88–97.

[7] Wang, G. Y., Cong, G. (2010). Community-based greedy algorithm for mining top-k influential nodes in mobile social networks, *In*: SIGKDD, p. 1039–1048.

[8] Goyal, F. B. A., Lakshmanan, L. V. S. (2012). A data-based approach to social inuence maximization, *In*: *Proceedings of the VLDB Endowment*, p. 73–84.

[9] Chen, Z. N. W., Lu, W. (2012). Time-critical influence maximization in social networks with time-delayed diffusion process, *In*: AAAI.

[10] Liu, D. X. B., Cong, G. (2012). Time constrained influence maximization in social networks, *In*: ICDM, p. 439–448.

[11] Lu, L. V. S. L. (2012). Profit maximization over social networks, *In:* International Conference on Digital Information Management (ICDIM) p. 479–488.

[12] Bhagat, A. G. S., Lakshmanan., L. V. S. (2012). Maximizing product adoption in social networks, *In*: WSDM, p. 603–612.

[13] Li, J. F. K. T. G., Chen, S., Li, W. (2014). Efficient location-aware inuence maximization, *In*: SIGMOD, p. 87–98.

[14] Tan, Y. L. Z. (2015). Realtime targeted influence maximization for online advertisements, *In*: VLDB, p. 1070–1081.

[15] Kimura, K. S. M., Nakano., R. (2008). Prediction of information diffusion probabilities for independent cascade model, *In*: Proceedings of the 12th International Conference on Knowledge- Based Intelligent Information and Engineering Systems, 2008.

[16] Goyal, F. B. A., Lakshmanan., L. V. S. (2010). Maximizing product adoption in social networks, *In*: WSDM, p. 241–250.

[17] Cantador, T. K. I., Brusilovsky, P. (2011). Second workshop on information heterogeneity and fusion in recommender systems, *In*: Rec- Sys, p. 387–388.

[18] Barbieri, G. M. N., Bonchi, F. (2013). Cascade-based community detection, *In*: WSDM, p. 33–42.