# Community-based Greedy Algorithm for Mining Top-K Influential Nodes in Mobile Social Networks

Yu Wang[†], Gao Cong[‡], Guojie Song[†*], Kunqing Xie[†]
[†]Key Laboratory of Machine Perception, Ministry of Education, Peking University, China
[‡]School of Computer Engineering, Nanyang Technological University, Singapore
wangyu@cis.pku.edu.cn, gaocong@ntu.edu.sg, gjsong@cis.pku.edu.cn,
kunqing@cis.pku.edu.cn

## ABSTRACT

With the proliferation of mobile devices and wireless technologies, mobile social network systems are increasingly available. A mobile social network plays an essential role as the spread of information and influence in the form of "word-of-mouth". It is a fundamental issue to find a subset of influential individuals in a mobile social network such that targeting them initially (e.g. to adopt a new product) will maximize the spread of the influence (further adoptions of the new product). The problem of finding the most influential nodes is unfortunately NP-hard. It has been shown that a Greedy algorithm with provable approximation guarantees can give good approximation; However, it is computationally expensive, if not prohibitive, to run the greedy algorithm on a large mobile network.

In this paper we propose a new algorithm called Community-based Greedy algorithm for mining top-$K$ influential nodes. The proposed algorithm encompasses two components: 1) an algorithm for detecting communities in a social network by taking into account information diffusion; and 2) a dynamic programming algorithm for selecting communities to find influential nodes. We also provide provable approximation guarantees for our algorithm. Empirical studies on a large real-world mobile social network show that our algorithm is more than an order of magnitudes faster than the state-of-the-art Greedy algorithm for finding top-$K$ influential nodes and the error of our approximate algorithm is small.

## Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Non-numerical Algorithms and Problems

## General Terms

Algorithms, Experimentation, Performance, Reliability

## Keywords

Social Networks, Community Detection, Influence Maximization

[*]Corresponding author. Email: gjsong@cis.pku.edu.cn

## 1. INTRODUCTION

A social network is a social structure connecting individuals or organizations. Examples of social networks include email networks, online FaceBook, and scientific collaboration networks[11]. A social network plays an important role as the spread of information and influence in the form of "word-of-mouth" [16]. It is a fundamental issue to find a small subset of influential individuals in a social network such that they can influence the largest number of people in the network [9].

Finding a subset of influential individuals has many applications. Let us recall the motivating example given by Kempel et al. [10]. Consider a social network together with the estimates for the extent to which individuals influence one another, and the network performs as the platform for marketing. A company would like to market a new product, hoping it will be adopted by a large fraction of the network. The company plans to initially target a small number of "influential" individuals of the network by giving them free samples of the product (the product is expensive or the company has limited budge so that they can only choose a small number of people). The company hopes that the initially selected users will recommend the product to their friends, their friends will influence their friends' friends and so on, thus many individuals will ultimately adopt the new product through the powerful word-of-mouth effect (or called viral marketing). The problem here is to choose a set of individuals to send the free samples such that they eventually influence the largest number of people in the network. Formally, the problem is called as *influence maximization* [10], which is, for a parameter $K$, to find a $K$-node set of maximum influence, where influence is propagated in the network according to a stochastic cascade model [10].

Kempel et al. establishes that the optimization problem of *influence maximization* is NP-hard [10]. They use the Greedy algorithm and prove that the optimal solution for influence maximization can be approximated to within a factor of $(1 - \frac{1}{e} - \epsilon)$. Their empirical study shows that the Greedy algorithm significantly outperforms the degree or centrality-based heuristics in influence maximization. The main problem of Greedy algorithm is efficiency, especially when the social network contains a large number of nodes. Recently, Chen et al.[4] propose an improved version of Greedy algorithm, called NewGreedy. To make the result better, it takes the first round with NewGreedy algorithm and the rest rounds using CELF Greedy algorithm[14], called MixedGreedy, which is shown to be more efficient than previously proposed Greedy algorithms[4, 10]. However, finding a small set of influential nodes in a large network with 723k nodes could still take days to complete on a modern server (as to be shown in our experiments).

In this paper, we tackle the problem of influence maximization in a Mobile Social Network (MSN) where individuals communicate with one another using mobile phones[7]. A MSN can be extracted from call logs and is modeled as a weighted directed graph: a phone user corresponds to a node; a directed edge from node $v$ to node $u$ is established, if there exits communication from $v$ to $u$, with the corresponding communication time as the weight of the edge.

We propose a new algorithm for mining top-$K$ influential nodes, called Community-based Greedy Algorithm (CGA). The basic idea is to exploit the community structure property of social networks. Intuitively, a community is a densely connected subset of nodes that are only sparsely linked to the remaining network [8]. Communities in a MSN represent real social groups [7], and thus individuals in a community will influence each other in the form of "word-of-mouth". The prohibitive cost of finding influential nodes over the whole network would be reduced greatly if we find influential nodes with regard to communities.

The main technical challenge of exploiting community structures to find top-$K$ influential nodes is to choose which communities to mine the top-$K$ influential nodes such that the results can be as close as the optimal solution. A straightforward solution is to mine top-$K$ influential nodes in each community and then aggregate them to get the best $K$ nodes. This is not desirable as it results in unnecessary computation. Instead we propose a dynamic programming method to incrementally choose which communities to process. Within a community we can adopt any existing algorithm to detect influential nodes. We also provide the provable approximation precision of our proposed solution.

Another technical issue is that we need an efficient algorithm to partition the network into communities based on information diffusion model[10]. There exist a number of algorithms [3,8,13,18,20, 21, 23, 24] for community detection. However, none of them takes into account information diffusion between nodes. To this end, we extend the algorithm[18] running in $O(E)$, where $E$ is the number of edges, one of the most efficient algorithm for community partitioning, in three aspects: 1) We accommodate information diffusion model into the algorithm; 2) The algorithm[18] is designed for undirected and unweighted graphs and needs to be extended for weighted directed graphs; 3) The algorithm generates too many small dispersed communities and finding influential nodes with regard to the small communities is prone to errors. To remedy the problem, we extend the algorithm with a combination step: we define combination entropy to measure the connection of two communities and combine them if the combination entropy between them is larger than a threshold.

The proposed algorithm has two salient features. First, it can greatly improve the efficiency of the existing algorithms for finding top-$K$ influential nodes while the loss in approximation precision is small. Second, it is orthogonal to existing algorithms for finding influential nodes in that it can combine with any of them by using them to detect influential nodes in a community.

In summary our contributions are twofold.

First, we propose a new algorithm for finding top-$K$ influential nodes; the algorithm exploits the community property of social networks. We offer the provable performance guarantee of the algorithm. Second, we conduct experiments on a large mobile social network with 723K nodes (the largest network used in previous work has 37K nodes) to evaluate the performance of our algorithm. Experimental results show that our algorithm can improve the state-of-the-art Greedy algorithm[4] by orders of magnitude while the approximation ratio is comparable to the Greedy algorithm.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 presents problem statement. Section 4 details the Community-based Top-$K$ Greedy algorithm and analyzes the precision of the algorithm. We report a performance evaluation in Section 5. Finally, we offer conclusions and research directions in Section 6.

## 2. RELATED WORK

Domingos and Richardson [6] are the first to study the influence maximization as an algorithmic problem and propose a probabilistic solution. Kempe et al. formulate the problem of finding a set of influential individuals as an optimization problem. They establish that the optimization problem is NP-hard [10], and present a Greedy Algorithm (GA), guaranteeing that the influence degree is within $(1 - \frac{1}{e})$ of the optimal influence degree. The basic idea of the GA algorithm is to calculate the influence set of each individual, and take turns to choose the node maximizing the marginal influence value until $K$ nodes are selected.

Leskovec et al.[14] present an optimized greedy algorithm, which is referred to as the "Cost-Effective Lazy Forward" (CELF) scheme. The CELF optimization uses the submodularity property of the influence maximization objective to reduce the number of evaluations on the influence spread of nodes. Recently, Chen et al.[4] propose two faster greedy algorithms called NewGreedy and MixedGreedy, respectively. Experiments show that MixedGreedy slightly outperforms NewGreedy. The main idea behind NewGreedy is to remove the edges that will not contribute to propagation from the original graph to get a smaller graph and do the influence diffusion on the smaller graph. The first round of MixedGreedy uses NewGreedy algorithm, and the rest rounds employ CELF algorithm. An earlier approach proposed by Kimura et al. [12] also removes edges that do not contribute to information diffusion, and does the propagation on the subnetwork. In addition, Chen et al.[4] also presents a degree discount heuristic algorithm called DegreeDiscount that runs faster than MixedGreedy. DegreeDiscount assumes that the influence spread increases with the degree of nodes. Unlike Greedy algorithm, DegreeDiscount algorithm has no provable performance guarantee.

The aforementioned approaches attack the efficiency issue by either improving GA algorithm or using new heuristics. However none of them takes into consideration the community property of social networks.

We also note that Scripps et al.[20, 21] present a metric to estimate the number of communities to which a node is attached and define community-based roles for a node (e.g. a node that links to many nodes from different communities takes a so-called ambassador role). They also briefly discuss the application of community-based roles, i.e. selecting nodes with an ambassador role to maximize the number of communities influenced by the selected nodes. The application problem is different from the influence maximization problem addressed in this paper and our algorithm is completely different from the approach [20, 21].

Our work is also related to community detection. There are a host of algorithms for community detection. However, it remains to be a challenging problem for a large social network. Some approaches assume that the number or size of the partitions into which the network is to be split is known in advance. They usually optimize a quality measure of network clustering, such as popular modularity measure [17], and need to solve an NP-complete problem. Some popular heuristic algorithms include min-max cut method [3], normalized cut [22], SCAN[3] and Spectral Clustering[13]. In a Mobile Social Network, the number of communities and the size of communities are unknown. Also, these methods are quite expensive when applied to a large network. A hierarchical clustering method [5] is proposed with complexity $O(Eh \log N)$, where $E$

is the number of edges, $N$ the number of nodes and $h$ the depth of hierarchical clusters. Raghavan et al. [18] propose a near linear algorithm running in $O(E)$ time, $E$ the number of edges, which is less expensive than earlier algorithms. In [18], it uses a simple label propagation algorithm that uses the network structure alone as its guide and requires neither optimizing an objective function nor prior information about the communities. However, none of existing approach takes into account information diffusion.

Although we focus on mobile social network in this paper, mobile social network reflects the properties of other scale-free networks, such as paper citation graph, in which degrees follow the the power-law distribution[19], and so does link-based similarity [2], and thus the proposed techniques would be applicable to other scale-free networks.

## 3. PRELIMINARIES AND PROBLEM STATEMENT

We introduce the Independent Cascade information diffusion model, and then the problem statement. Table 1 lists the notations to be used extensively in the rest of this paper.

**Table 1: Notation explanation**

| Notation | Description |
|---|---|
| $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ | A weighted directed MSN with vertex set $\mathcal{V}$, edge set $\mathcal{E}$ and weight $\mathcal{W}$ |
| $N$ | The number of nodes in $\mathcal{G}$ |
| $E$ | The number of edges in $\mathcal{G}$ |
| $d$ | The average degree of the network |
| $K$ | The number of influential nodes to be mined |
| $\mathcal{I}$ | The set of $K$ influential nodes |
| $\mathcal{C}_i$ | The $i$th community |
| $M$ | The number of communities |
| $\overline{\lambda}$ | The average diffusion speed |
| $R(A)$ | Influence degree in $\mathcal{G}$ of nodes in set A |
| $R_m(A)$ | Influence degree in community $\mathcal{C}_m$ of nodes in A |

**Mobile Social Networks.** We extract a Mobile Social Network from the call log and model it as a directed weighted graph: a phone user corresponds to a node; a directed edge from node $u$ to node $v$ is established, if there exits communication from $u$ to $v$, with the corresponding communication time as the weight of the edge. We denoted the graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, where $\mathcal{V}$, $\mathcal{E}$, and $\mathcal{W}$ represent nodes, edges, and weights, respectively.

**Independent Cascade Model.** The model is originally proposed by Lopez-Pintado[15], and is the most common dynamic model in information diffusion. It is widely used in maximizing the number of individuals who are influenced by the merchandize information in direct marketing. It is used in previous work on influence maximization [4, 10]. In the model we assign two states to nodes: ***active*** and ***inactive***. Active nodes are those that are influenced by other active nodes, and are able to influence their inactive neighbors; inactive nodes are those that are not influenced by their active neighbors. The state of a node can be switched from being inactive to being active, but not vice versa. The model has an important parameter called ***diffusion speed*** $\lambda$. When an active node $v_i$ contacts an inactive node $v_j$, the inactive node becomes active at a probability (rate) $\lambda$. In a viral marketing, ***diffusion speed*** models the tendency of individuals to accept a product. Thus the diffusion is affected by diffusion speed, node degree, and the number of initial active nodes.

The Independent Cascade model does not take edge weight into consideration. However, a mobile social network is a directed weighted graph, and the weight should play an important role in information diffusion. Intuitively, the more contacts between two nodes, the more likely the influence will happen. We extend the definition ***diffusion speed*** in the Independent Cascade model to accommodate the edge weight. The influence rate raises as the weight between nodes increases. We note that Weighted Independent Cascade model [10], a special case of Independent Cascade model, considers node weight but not edge weight, and thus is still not directly applicable to a MSN. We also note that Kempe et al. [10] also considers Linear Threshold model. As proved in [9], Linear Threshold model and Independent Cascade model can be unified with proper parameter initialization. We can extend the linear threshold model as we do for Independent Cascade model to accommodate edge weight, and our proposed algorithm is equally applicable.

**DEFINITION** 1. (**Diffusion Speed.**) *The diffusion speed from node $v_i$ to node $v_j$ $\lambda_{ij}$ is defined as:*

$$\lambda_{ij} = 2\overline{\lambda}\frac{w_{ij}}{w_{max} + w_{min}} \qquad (1)$$

*where $\overline{\lambda}$ is the average diffusion speed of the whole network, $w_{ij}$ is the weight of direct edge from $v_i$ to $v_j$, $w_{min}$ is the minimum weight of directed edge and $w_{max}$ the maximum weight in the network.*

Then the diffusion mechanism can be described as follows:

1. The diffusion process begins with an initial set of active nodes $A$ at step $t$=0. Let $S_0$ = $A$;
2. At each step $t$, an active node $v_i$ from last step $S_{t-1}$ will be given a single chance to influence each of its inactive neighbors $v_j$, with success probability $\lambda_{ij}$. If $v_j$ is successfully influenced, it is activated and is added set $S_t$.
3. The process terminates when $S_t$ is empty. The set of nodes influenced by $A$ is the union of all the $S_t$ generated at each step. We denote the number of nodes influenced by $A$ as $\mathcal{V}_A$.

**DEFINITION** 2. (**Influence Degree**) *Let A be the initial set of active nodes. The influence degree of set A is computed as:*

$$R(A) = \mathcal{V}_A/N \qquad (2)$$

*where $\mathcal{V}_A$ is the number of nodes influenced by A during information diffusion process.*

**ProblemStatement**: Given a mobile social network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, we aim to mine a set of top-$K$ influential nodes $S$ on the network such that $R(S)$ is maximized using the extended Independent Cascade information diffusion model.

It has been proved that the optimization problem is NP-hard [10]. A greedy algorithm can approximate the optimum to within a factor of $(1 - \frac{1}{e})$. However, the greedy algorithm is expensive for solving the influence maximization problem on a large-scale network. So we propose a community based greedy algorithm which mine the influential nodes in each community rather than the whole network.

## 4. MINING TOP-$K$ INFLUENTIAL NODES

We first present the proposed algorithm CGA for mining top-$K$ influential nodes in communities, and then present the partitioning algorithm that takes into account the diffusion model. We also provide precision analysis of the proposed algorithm.

## 4.1 CGA:Community-Based Greedy Algorithm

Although a greedy algorithm gives a good approximation to the problem of finding top-$K$ influential nodes, it is expensive for large networks.

We first present an important concept to be used in the rest of subsection. We say that a node $v$ **influences** node $u$ if node $v$ activates node $u$, i.e. $u$ becomes active from inactive, for at least $Q/2$ times out of $Q$ simulations of the diffusion process of Independent Cascade model. We also say that $u$ is *influenced by* $v$. Recall that whether $v$ can influence $u$ depends on the influence speed and also the weight from $v$ to $u$.

The main idea of our method is to divide a network into communities, and then choose communities to find top-$K$ influential nodes within communities. The community structure is a salient property of social network features: Individuals within a community have frequent contact and thus are more likely to influence each other; In contrast, individuals across communities have much less contact with each other and thus are less likely to influence each other. This property suggests that it might be a good approximation to choose influential nodes within communities instead of the whole network. Obviously it will be more efficient to choose influential nodes within communities.

Suppose that we already divide a network into $M$ communities (to be discussed in the next subsection). The remaining technical challenge is to choose which communities to find the top-$K$ influential nodes. A straightforward solution is to mine top-$K$ influential nodes in each community and then aggregate them to get the top-$K$ nodes. This is not desirable as it wastes some computation.

We propose a dynamic programming algorithm to choose which community to find the $k^{th}$ influential node, $k \in [1, K]$. Let $\mathcal{I}_{k-1}$ be the set of influential nodes obtained in the previous $k$-1 steps. We are then ready to compute the maximal increase of the influence degree with regard to community $\mathcal{C}_m$ if we mine the $k^{th}$ node in $\mathcal{C}_m$. The maximal increase is denoted as $\Delta R_m$ and we have:

$$\Delta R_m = \max\{R_m(\mathcal{I}_{k-1} \cup v_j) - R_m(\mathcal{I}_{k-1})|v_j \in \mathcal{C}_m\} \quad (3)$$

Note that in Equation 3 the influence degree $R_m(.)$ is computed with regard to the community $\mathcal{C}_m$ rather than the whole network.

To mine the $k^{th}$ influential node, we need to choose the community that will yield the largest increase of influence degree among all the communities. Let $R[m, k]$ ($m \in [1, M]$ and $k \in [1, K]$) be the influence degree of mining the $k^{th}$ influential node in the first $m$ communities. We have:

$$R[m, k] = max\{R[m-1, k], R[M, k-1] + \Delta R_m\} \quad (4)$$

$$R[m, 0] = 0, R[0, k] = 0 \quad (5)$$

The Equation 4 can be understood as follows: If the influence degree of mining the $k^{th}$ node in the first $m$-1 communities is smaller than that of mining the $k^{th}$ node in $\mathcal{C}_m$, we mine the $k^{th}$ node in $\mathcal{C}_m$; otherwise, we mine it in the former $m$-1 communities.

We select a community from the first $m$ communities to mine the $k^{th}$ influential node, and the selected community is represented by a sign function $s[m, k]$. It is defined as follows:

$$s[m,k] = \begin{cases} s[m-1,k], & R[m-1,k] \geq R[M,k-1] + \Delta R_m \\ m, & R[m-1,k] < R[M,k-1] + \Delta R_m \end{cases}$$

$$s[0, k] = 0 \quad (6)$$

To find the $k^{th}$, $k \in [1, K]$, influential node, we choose the community $s[M, k]$ (Note that these sign functions $s[m, k]$, $m < M$, are used to compute $s[M, k]$).

In principle, we can use any existing algorithm to find the $k^{th}$ influential node in community $s[M, k]$. In this paper, we adopt the MixedGreedy algorithm [4] since it has the same approximation precision as previously proposed greedy algorithms and is shown to be more efficient.

**Algorithm 1** CGA Algorithm

```
Input: network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, size of results $K$, influence speed $\overline{\lambda}$;
Output: $\mathcal{I}$: Top-$K$ influential nodes;
 1: $\mathcal{C} \leftarrow$ detect communities in $\mathcal{G}$;
 2: $M = |\mathcal{C}|$
 3: $\mathcal{I} = \mathcal{I}_1 = \mathcal{I}_2 = ... = \mathcal{I}_M = \emptyset$;
 4: for $k = 1$ to $K$ do
 5:     $R[0, k] = 0$; $s[0, k] = 0$;
 6: end for
 7: for $m = 1$ to $M$ do
 8:     $R[m, 0] = 0$;
 9: end for
10: for $k = 1$ to $K$ do
11:     for $m = 1$ to $M$ do
12:         $\Delta R_m = max\{R_m(\mathcal{I} \cup \{v_i\}) - R_m(\mathcal{I})|v_i \in \mathcal{C}_m\}$;
13:         $R[m, k] = max\{R[m-1, k], R[M, k-1] + \Delta R_m\}$;
14:         if $R[m-1, k] \geq R[M, k-1] + \Delta R_m$ then
15:             $s[m, k] = s[m-1, k]$;
16:         else
17:             $s[m, k] = m$;
18:         end if
19:     end for
20:     $j = s[M, k]$;
21:     $v_{max} = \underset{v_i \in \mathcal{C}_j}{\text{argmax}}(R_j(\mathcal{I}_j \cup \{v_i\}) - R(\mathcal{I}_j))$
22:     $\mathcal{I}_j = \mathcal{I}_j \cup \{v_{max}\}, \mathcal{I} = \mathcal{I} \cup \{v_{max}\}$;
23: end for
```

The algorithm is outlined in Algorithm 1. It first detects communities (line 1), and initializes the set of influential nodes discovered in the whole network ($\mathcal{I}$) and in each community ($\mathcal{I}_1$, ..., $\mathcal{I}_M$) as NULL (line 3). The algorithm initializes $R[m, k]$ and $s[m, k]$ (lines 4–9). In lines 10 – 19, the algorithm chooses which community to mine $k^{th}$ influential node using dynamic programming algorithm. If the sum of the influence degree $\Delta R_m$ and $R[M, k-1]$ is smaller than $R[m-1, k]$, i.e. the influence degree of mining $k^{th}$ node in the first $m$-1 communities (line 14), we should not find the $k^{th}$ node in community $\mathcal{C}_m$ and thus label $s[m, k]$ as $s[m-1, k]$ (line 15); otherwise we mine the $k^{th}$ node in community $\mathcal{C}_m$ and set $s[m, k]$ as $m$ (line 17). The algorithm will find the $k^{th}$ influential node in the community $\mathcal{C}_j$, $j$= $s[M, k]$ (line 20). We employ MixedGreedy algorithm[4] to find the node $v_{max}$ that maximizes $R_j(\mathcal{I}_j \cup \{v_i\}) - R(\mathcal{I}_j)$ in community $\mathcal{C}_j$ (line 21).

We proceed to illustrate the **CGA** Algorithm with an example.

1. We mine top-2 nodes from a network. Assume that the network is partitioned into three communities $\mathcal{C}_1$,$\mathcal{C}_2$, and $\mathcal{C}_3$, and based on Independent Cascade model we have: the maximal influence degree increment, denoted by $\Delta R_1$, $\Delta R_2$ and $\Delta R_3$ for each community, is $0.2, 0.3, 0.1$, respectively, for each community; and the second most maximal influence degree increment is 0.08, 0.06, 0.04, respectively, for each community. CGA works as follows:

2. We proceed to find which community to mine top=1 node.

   $R[1, 1] = max\{R[0, 1], R[3, 0] + \Delta R_1\} = 0.2$, $s[1, 1] = \mathcal{C}_1$;

$R[2,1] = max\{R[1,1], R[3,0] + \Delta R_2\} = max\{0.2, 0 + 0.3\} = 0.3$, $s[2,1] = \mathcal{C}_2$;

$R[3,1] = max\{R[2,1], R[3,0] + \Delta R_3\} = max\{0.3, 0.1\} = 0.3$, $s[3,1] = \mathcal{C}_2$;

Hence, we mine the top-1 node in community $\mathcal{C}_2$ because $s[3,1] = \mathcal{C}_2$ is the result.

3. Next, $R[1,2] = max\{R[0,2], R[3,1] + \Delta R_1\} = max\{0, 0.3 + 0.2\} = 0.5$, $s[1,2] = \mathcal{C}_1$;

   $R[2,2] = max\{R[1,2], R[3,1]+\Delta R_2\} = max\{0.5, 0.3+ 0.06\} = 0.5$, $s[2,2] = \mathcal{C}_1$;

   $R[3,2] = max\{R[2,2], R[3,1]+\Delta R_3\} = max\{0.5, 0.3+ 0.1\} = 0.5$, $s[3,2] = \mathcal{C}_1$.

   Note that $\Delta R_2$ in this step is 0.06, but not 0.3. We mine the second node in community $\mathcal{C}_1$ because $s[3,2] = \mathcal{C}_1$.

*Complexity Analysis:* We consider the complexity of lines 2–23 of Algorithm 1 (line 1 will be analyzed in next subsection). The algorithm needs $O(K)$ time in lines 4–6, and $O(M)$ time in lines 7–9. Suppose the largest community after community detection is $\mathcal{C}_p$, thus lines 10–19 take $O(MKT_p)$ time, and lines 20–23 of mining nodes using MixedGreedy algorithm [4, 14] (that uses NewGreedy [4] in the first round and uses CELF algorithm [14] for the rest rounds) take $O(K|C_p|T_p)$ time, where $T_p$ is the time to compute the influence degree of a node in community $\mathcal{C}_p$, and it takes $O(QE_p)$ time (where $Q$ is the number of simulations for the Independent Cascade model, and $E_p$ is the number of edges in community $\mathcal{C}_p$). Hence the worst-case complexity of lines 2–23 of Algorithm 1 is $O(MKT_p + K|C_p|T_p)$.

## 4.2 Influential Model Based Community Detection Algorithm

Community structure is a basic property of a MSN and communities represent real circles of social groups in which members are more likely to influence each other[1, 7]. We will make use of the detected communities to approximate the influence of nodes in the whole network. Hence, we want to detect communities based on the influences between nodes, rather than only the connection between nodes, such that the influence degree of nodes within a community can be as close as that in the whole network. There exist a number of algorithms for community detection, and they partition graphs based on the node connections. However, none of them takes into account information diffusion between nodes.

Our community detection algorithm consists of two steps, partition and combination. 1) **Partition.** we extend the algorithm[18] with the information influence mechanism based on Independent Cascade model. The algorithm[18], a nearly linear algorithm for community detection, is designed for undirected and unweighted graph, and thus is not directly applicable to a MSN. 2) **Combination.** The generated communities in a MSN by the partition step are very small and dispersed; we develop a method to combine communities such that the difference between influence degree of a node in its community and its influence degree in the whole network is restricted. We proceed to present the two steps.

### 4.2.1 Community Partition

We proceed to present an overview of our partitioning method. 1) Initially, each node is assigned a unique community label from 1 to $N$ (the number of nodes); 2) for each node we compute the set of its influenced neighbors using Independent Cascade diffusion model; 3) we iteratively propagate the labels through the network in finite iterations. The main principle of the label propagation is that a node $v$ should belong to the community that contains the maximum number of its influenced neighbors. In algorithm [18], label propagation is performed among a node and its neighbors while we perform the propagation among a node and its neighbors that can be influenced by the node (based on Independent Cascade model). Note that this principle also renders our community detection problem different from that considered in previous proposals, e.g. [18].

Steps 1 and 2 are straightforward and we proceed to explain the step 3, i.e. to perform the propagation of community label. This propagation is iterative in nature. At each iteration of the label propagation, we assign the community label for a node $v$ based on the labels of its neighbors that are influenced by $v$. Specifically, for each node $v$ we find out the label of the community that the majority of its influenced neighbors belong to, and the label will become the label of $v$. Formally, the label, denoted by $v.C^t$, of a node $v$ at iteration $t$ is represented as follows:

$$v.C^t = \mathrm{maxCMT}(u_1.C^{t-1}, u_2.C^{t-1}, ..., u_{s_v}.C^{t-1}) \quad (7)$$

where $t$ denotes the $t$th iteration, $s_v$ denotes the number of neighbors that are influenced by $v$, $u_i$ ($i \in [1, s_v]$) represents an influenced neighbor of node $v$, $u_i.C^{t-1}$ represents the community label of $u_i$ at iteration $t$-1, and $\mathrm{maxCMT}$ is to compute the majority label of $u_i.C^{t-1}$ ($i \in [1, s_v]$).

The partition step is detailed in lines 1–16 in Algorithm 2. The algorithm takes in three arguments, a network $\mathcal{G}$, the number of iterations $\tau$ and a threshold $\theta$ to be used by the combination step. In the algorithm, a node $v$ belongs to a community and its community label is denoted as $v.C$. Node $v$ is also associated with a bit vector $H_v$ of length $d_v$ that is the degree of node $v$. Each neighbor $u_j$ of $v$ corresponds to a bit $H_v(j)$, $j = 1, ..., d_v$, in $H_v$. If $v$ influences its neighbor $u_j$, the corresponding bit $H_v(j)$ is set to 1 and otherwise 0. We can easily compute $s_v$, the number of neighbors influenced by $v$, by counting the number of bit 1 in $H_v$. The algorithm initializes the community label of each node $v$ and finds the set of neighbors that are influenced by node $v$ in lines 1–10. The function IsInfluence is implemented according to the definition of *influence* (Section 4.1). The algorithm then iteratively propagates community labels in lines 11–16: At each iteration $t$, we update the label of node $v$ with the majority label of its neighbors that are influenced by $v$. It has been shown in[18] that the propagation will become relatively stable in a few iterations.

### 4.2.2 Community Combination

We expect that the difference between the node's influence degree in its community and its influence degree in the whole network is small. To achieve a good set of top-$K$ influential nodes with a good influence degree in our algorithm, we define combination entropy to measure the connections of two communities and combine two communities if the combination entropy between them is larger than a threshold.

**DEFINITION** 3. (**CombinationEntropy**) *If a node $v$ influences (activates) its neighbor $u$, we label the edge $e_{vu}$ as **live** [10]. If $e_{vu}$ is live and $v$ belongs to community $\mathcal{C}_m$, but $u$ belongs to a different community $\mathcal{C}_l$, we say that $u$ is a **live node** of $\mathcal{C}_m$. Let $L[\mathcal{C}_m]$ be the set of live nodes of $\mathcal{C}_m$. The combination entropy of community $\mathcal{C}_l$ to $\mathcal{C}_m$ is defined as:*

$$\mathrm{CoEntropy}(\mathcal{CE}^l_m) = \max_{v\in\mathcal{C}_m, u\in L[\mathcal{C}_m], u\in\mathcal{C}_l} \frac{\overline{R}_m(\{u\})}{R_m(\{v\})},$$

*where $\overline{R}_m(\{u\})$ denotes the influence degree of node $u$ outside the community $\mathcal{C}_m$, and $R_m(\{v\})$ denotes the influence degree of node*

**Algorithm 2** Community Detection

```
Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, threshold $\theta$, number of iterations $\tau$;
Output: the set of communities $\mathcal{C}$;
 1: for each $v \in V$ do
 2:     $v.C^0 \leftarrow$ a distinct community label;
 3:     for each neighbor $u_j$ of $v$ do
 4:         if IsInfluence($v, u_j$) then
 5:             $H_v(j) = 1$;
 6:         else
 7:             $H_v(j) = 0$;
 8:         end if
 9:     end for
10: end for
11: for $t$ =1 to $\tau$ do
12:     for each $v \in V$ do
13:         $v.C^t = \text{maxCMT}(u_1.C^{t-1}, u_2.C^{t-1}, ..., u_{s_v}.C^{t-1})$;
14:     end for
15:     $t = t + 1$;
16: end for
    {The following is the combination step}
17: $L[\mathcal{C}_i] = \emptyset, i \in [1, Z]$; {Z is the number of communities from Step 1}
18: for each $v \in V$ do
19:     for each neighbor $u_j, j \in [1, d_v]$, of $v$ do
20:         if (($v.C \neq u_j.C$) and ($H_v(j) = 1$)) then
21:             $L[\mathcal{C}_i] = L[\mathcal{C}_i] \cup \{u_j\}$;
22:         end if
23:     end for
24: end for
25: $isComb = 1$;
26: while $isComb = 1$ do
27:     for each community $\mathcal{C}_m$ do
28:         $\mathcal{S} = \{u.C | u \in L[\mathcal{C}_m]\}$;
29:         $isComb = 0$;
30:         for each $\mathcal{C}_l \in \mathcal{S}$ do
31:             Compute combination entropy CoEntropy($\mathcal{CE}^l_m$) for community $\mathcal{C}_l$
32:             if (CoEntropy($\mathcal{CE}^l_m$) > $\theta$) then
33:                 Integrate $\mathcal{C}_l$ into $\mathcal{C}_m$ and update $\mathcal{C}_m$;
34:                 $isComb$=1;
35:             end if
36:         end for
37:     end for
38: end while
```

*$v$ in its community $\mathcal{C}_m$. As we detect the communities based on diffusion model, we have $\overline{R}_m(\{u\}) \approx R_l(\{u\})$.*

For a node $v$ in $\mathcal{C}_m$, $L[\mathcal{C}_m]$ includes its influenced neighbors such that they will make diffusion degree of $v$ with regard to $\mathcal{C}_m$ different from diffusion degree of $v$ with regard to the whole network. Intuitively, combination entropy measures the difference of diffusion spread of a node $v$ ($v$ *in* $\mathcal{C}_m$) between in the community and in the whole network through a node in $L[\mathcal{C}_m]$. The difference reflects the precision loss due to diffusion within communities.

We set a threshold $\theta$. If the *combination entropy* CoEntropy ($\mathcal{CE}^l_m$) of community $\mathcal{C}_m$ to community $\mathcal{C}_l$ is bigger than $\theta$, then $\mathcal{C}_m$ and $\mathcal{C}_l$ will be combined.

The combination step is outlined in lines 17–38 in Algorithm 2. In lines 17–24, we compute the set $L[\mathcal{C}_m]$ for each community. The algorithm checks each node in $L[\mathcal{C}_m]$ set and decides whether $\mathcal{C}_l$ is combined into community $\mathcal{C}_m$ (lines 25–38). To make sure the *combination entropy* between any two communities is less than $\theta$, we perform the combination iteratively. We use a variable $isComb$ to record whether two communities are combined in each iteration; if no combination is done in an iteration ($isComb$ =0), we terminate the while-loop.

*Complexity Analysis*: It needs $O(EQ)$ time in lines 1–10, where $Q$, the rounds of simulations, is constant. Lines 11–16 need $O(E\tau)$ time, where $\tau$ is constant. Lines 17–24 needs $O(E)$ time. Let $Z$ and $M$ denote the number of communities before and after combination, respectively. Let $\mathcal{C}_p$ be the maximal community. Recall that computing the influence degree for one node in $\mathcal{C}_p$ take $T_p$ time. Lines 25–38 take $O((Z-M)NT_p)$ time in the worst case: the number of iterations is $Z$-$M$ in the worst case; each iteration takes $O(NT_p)$ time in the worst case, where we compute influence degree at $N$ nodes at most. Note that in practice it will be much less expensive than the worst case complexity since it is very likely that we do not need to compute influence degree for all nodes, and the number of iterations is smaller than $Z$-$M$, and many communities are smaller than the maximal community $\mathcal{C}_p$. Hence the worst-case complexity of Algorithm2 is $O(E+(Z-M)NT_p)$. Taken together with the complexity of Algorithm1, the total worst-case complexity is $O(E + (Z-M)NT_p + MKT_p + K|C_p|T_p)$.

## 4.3 Precision Analysis of CGA

We proceed to derive the performance guarantee of the CGA by analyzing the approximation ratio of CGA.

**LEMMA** 1. *Consider a node $v$ in community $\mathcal{C}_m$. The influence degree $R_m(\{v\})$ of node $v$ in its community is $\frac{1}{1+\Delta d * \theta}$ approximate to the influence degree in the whole network $\mathcal{G}$, where $\theta$ is the threshold used in the combination step in Algorithm 2 and $\Delta d$ is the maximal difference between the number of nodes affected by a node in network $\mathcal{G}$ and that in community $\mathcal{C}_m$. We have*

$$R_m(\{v\}) \geq \frac{1}{1 + \Delta d * \theta} R(\{v\}) \tag{8}$$

**PROOF.** Let $\Delta R(\{v\}) = R(\{v\}) - R_m(\{v\})$ be the difference of the diffusion degree in $\mathcal{C}_m$ and $\mathcal{G}$. As the combination is based on combination entropy $CoEntropy(\mathcal{CE}^l_m)$ and follows threshold $\theta$, we have the following:

$$\Delta R(\{v\}) = R(\{v\}) - R_m(\{v\})$$

$$= \sum_{u \in L[\mathcal{C}_m]} \overline{R}_m(\{u\}) \leq \sum_{u \in L[\mathcal{C}_m]} \theta * R_m(\{v\})$$

$$= (d[v] - d_m[v])\theta * R_m(\{v\}) = \Delta d[v] * \theta * R_m(\{v\})$$

where $d[v]$ denotes the number of active neighbors of node $v$ influenced by $v$ in the whole network, $d_m[v]$ denotes the number of active neighbors of node $v$ influenced by $v$ in the community $\mathcal{C}_m$ and $\Delta d[v] = d[v] - d_m[v]$.

Thus, we have

$$R_m(\{v\}) \geq \frac{1}{1 + \Delta d[v] * \theta} R(\{v\}) \geq \frac{1}{1 + \Delta d * \theta} R(\{v\}).$$

We get the proof. □

When the directed weighted mobile social network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ and diffusion speed $\overline{\lambda}$ are given, the number of neighbors that a node can activate in network $\mathcal{G}$ or a community can be estimated. Thus $\Delta d$ is a small number. $\theta$ is a constant.

**THEOREM** 1. *Let $\mathcal{I}^*$ be the set of $K$ nodes that maximize the influence degree, i.e. the optimal solution to our problem; let $\mathcal{I}$ be the set of $K$ nodes discovered by our algorithm; let $R(\mathcal{I}^*)$ and $R(\mathcal{I})$ be the influence degree of the two sets, respectively. The proposed CGA obtains a $(1 - e^{-\frac{1}{1+\Delta d \Theta}})$-approximation, that is*

$$R(\mathcal{I}) \geq (1 - e^{-\frac{1}{1+\Delta d \Theta}}) R(\mathcal{I}^*) \tag{9}$$

**PROOF.** Let $R_m(\{v_i\})$ be the influence degree of node $v_i$ in community $\mathcal{C}_m$ and $R(\{v_i\})$ be the influence degree of node in the whole network. Let $\mathcal{I}(k)$ be the first $k$ nodes in $\mathcal{I}$. Let the $k^{th}$ node in $\mathcal{I}$ be $v_k$.

At least $R(\mathcal{I}^*) - R(\mathcal{I}(k-1))$ individuals that are not covered by $R(\mathcal{I}(k-1))$ are covered by the $k$ subsets of $R(\mathcal{I}^*)$[16]. Hence, according to the *pigeonhole principle*, one of the $k$ subsets in the optimal solution must cover at least $\frac{R(\mathcal{I}^*(k))-R(\mathcal{I}(k-1))}{k}$ individuals[16]. As we spread the diffusion in community $\mathcal{C}_m$, the influence degree of $v_k$, $R(\{v_k\}) \geq \frac{R(\mathcal{I}^*(k))-R(\mathcal{I}(k-1))}{k(1+\Delta d\theta)}$. From the above analysis, we have $R(\mathcal{I}(1)) \geq \frac{R(\mathcal{I}^*(K))}{K(1+\Delta d\theta)}$. Thus, we have the following deduction process:

$$\begin{array}{rcl} R(\mathcal{I}(K)) & = & R(\mathcal{I}(K-1)) + R\{v_K\} \\ & \geq & R(\mathcal{I}(K-1)) + \frac{R(\mathcal{I}^*(K))-R(\mathcal{I}(K-1))}{K(1+\Delta d\theta)} \\ & = & (1-\frac{1}{K(1+\Delta d\theta)})R(\mathcal{I}(K-1)) + \frac{R(\mathcal{I}^*(K))}{K(1+\Delta d\theta)} \\ & \geq & (1-\frac{1}{K(1+\Delta d\theta)})^2 R(\mathcal{I}(K-2)) + \\ & & (1+(1-\frac{1}{K(1+\Delta d\theta)}))\frac{R(\mathcal{I}^*(K))}{k(1+\Delta d\theta)} \\ & & \ldots \\ & \geq & (1-\frac{1}{K(1+\Delta d\theta)})^{K-1} R(\mathcal{I}(1)) + \\ & & \sum_{i=0}^{K-2}(1-\frac{1}{K(1+\Delta d\theta)})^i \frac{R(\mathcal{I}^*(K))}{K(1+\Delta d\theta)} \\ & \geq & (1-(1-\frac{1}{K(1+\Delta d\theta)})^K)R(\mathcal{I}^*(K)) \end{array}$$

$$\begin{array}{rcl} \lim\limits_{k\to\infty}(1-\frac{1}{K(1+\Delta d\theta)})^K & = & \lim\limits_{k\to\infty}((1-\frac{1}{K(1+\Delta d\theta)})^{K(1+\Delta d\theta)})^{\frac{1}{1+\Delta d\theta}} \\ & = & (\lim\limits_{k\to\infty}(1-\frac{1}{K(1+\Delta d\theta)})^{K(1+\Delta d\theta)})^{\frac{1}{1+\Delta d\theta}} \\ & = & e^{-\frac{1}{1+\Delta d\theta}} \end{array}$$

As $f(K) = (1-\frac{1}{K(1+\Delta d\theta)})^K$ is a monotone increasing function, $e^{-\frac{1}{1+\Delta d\theta}}$ is the upper bound of $f(K)$. Hence $1 - e^{-\frac{1}{1+\Delta d\theta}}$ is the lower bound of $1-(1-\frac{1}{K(1+\Delta d\theta)})^K$. So $R(\mathcal{I}) \geq (1 - e^{-\frac{1}{1+\Delta d\Theta}})R(\mathcal{I}^*)$. We get the proof.

Based on Theorem 1, we can derive that CGA is a more general formulation compared with greedy algorithm for mining top-$K$ influential nodes in social networks. When $\theta$=0, as $K$ approaches $\infty$, the approximation factor $(1-(1-\frac{1}{K(1+\Delta d\theta)})^K)$ will become $(1-1/e)$, the approximation factor of Greedy Algorithm (GA). It shows the lower bound of our CGA is $(1-1/e)$ at $\theta = 0$, which is the same precision with that of GA. This is to say that GA is an instance of CGA.

*Remarks:* The parameter $\theta$ actually plays a role in balancing the approximation precision and efficacy. When $\theta = 0$, the number of generated communities will be 1, i.e. all communities will be combined into one, our algorithm is the same as the original Greedy algorithm with $(1-1/e)$-approximation. If $\theta > 0$, we have $(1-e^{-\frac{1}{1+\Delta d\theta}})$- approximation. As $\theta$ becomes smaller, the precision becomes higher; the number of generated communities $M$ will become smaller and thus the complexity becomes larger.

## 5. EXPERIMENTS

We evaluate the effectiveness and efficiency of the proposed CGA algorithm.

The data sets we used and experimental setup beforehand are described at first, and then the results with different parameters are shown.

All the experiments were conducted on a server with 2.0 GHz Intel Xeon 8 Core CPU and 8G memory running Debian/4.0 Operating system.

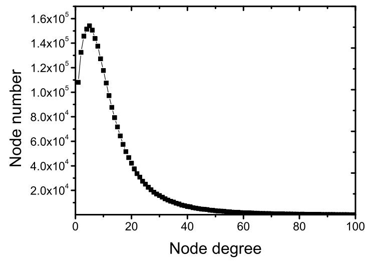


**Figure 1:** **Degree distribution of mobile social network**

## 5.1 Experimental setup

### *5.1.1 Data Sets*

We have a three-month CDR (call detailed record) data of a city from China Mobile, the largest mobile communication service provider in China. We extract a Mobile Social Network from the CDR data using the method presented in Section 3, and obtain a network with 723,201 nodes and an average degree of 13.4. Figure 1 shows the degree distribution of the mobile social network. We can see that it follows the power-law distribution, i.e. it is a scale-free network as many other social networks.

### *5.1.2 Algorithms and Parameters*

We take MixedGreedy [4] as the benchmark to evaluate the proposed algorithm CGA for two reasons. First, MixedGreedy is the state-of-the-art Greedy Algorithm for influence maximization, and it is shown that MixedGreedy outperforms previously proposed Greedy Algorithms, such as GA[10] and CELF[14]. Second, CGA adopts MixedGreedy to find influential nodes within communities, and thus a performance comparison between them will reveal their pros and cons.

Additionally, we compare with the other greedy algorithm proposed by Chen et al. NewGreedy[4]. We also compare with two heuristic algorithms DegreeDiscount[4] (See related work) and Random (simply select $K$ random nodes in the graph; it is evaluated in [4, 10]). To study the usefulness of the proposed community detection algorithm for CGA, we modify the algorithm CGA by replacing its community detection method (Section 3.2) with the community detection algorithm in [18], and the modified CGA is denoted as SPCGA.

We use two metrics to evaluate performance, namely the influence degree and runtime, by following the previous work on influence maximization.

We also evaluate the effect of parameters on performance, including the number of required nodes ($K$), the average influence speed ($\overline{\lambda}$), and the threshold for community combination ($\theta$). For parameters $K$ and $\overline{\lambda}$, we take both the previous work [4,12] and our actual network into consideration to choose the range of values.

## 5.2 Experimental Results

### *5.2.1 Experiment on community detection*

This experiment is to evaluate the performance of the proposed community detection algorithm on a Mobile Social Network. In the Independent Cascade diffusion model, we use the diffusion speed 0.05, and set the number of simulations as 100; a neighbor of a node is influenced by the node if the neighbor is activated by the node for at least 50 times in 100 simulations. To choose a value for the parameter $\theta$, the threshold for combination, we do experiment on a sub-graph with 1000 nodes (we randomly select a node from the whole MSN, and do a depth-first traversal starting from the se-

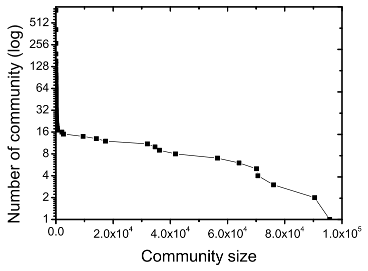


**Figure 2:** **Size distribution of extracted communities**

lected node to get 1000 nodes). We do experiment with different $\theta$ from 0.1 to 0.6 to combine the communities generated in the partition step. When $\theta$=0.1, all the communities are combined; when $\theta$=0.2, most of the communities are combined and the number of final communities is only 5; when $\theta$=0.5, the generated communities are dispersed and small (24 of them have less than 10 nodes); when $\theta$=0.3, we get 11 communities and this appear to be an appropriate number after we check the data manually. Hence we set $\theta$=0.3 as the threshold. Another parameter to be set is the number of iterations. It is suggested that the partitioning result would usually become relatively stable after 5 iterations in [18]. We observe that it is not completely stable after 5 iterations, i.e. the partitions of some nodes are still changed. To be safe, we set the number of iterations at 20 to make it more stable.

It takes 8,702 seconds for the proposed community detection algorithm to complete on the whole graph. In this paper, we take the runtime of community detection as part of runtime of CGA algorithm. However, in practice the community detection could be regarded as a pre-processing step since after the community detection is done we can find top-$K$ nodes with different $K$ values. The algorithm partitions the whole network into communities. It returns 36 communities(ignoring very small communities), among which the size of the largest community is 95,690 and the others are smaller. The size distribution of extracted communities is shown in Figure 2.

### 5.2.2 Varying $K$

This experiment is to evaluate the effect of the parameter $K$ on the influence degree and efficiency of different algorithms. We fix the average influence speed $\overline{\lambda}$ at 0.05, network size at 723k and combination threshold at 0.3. We vary $K$ from 1 to 30. The results are shown in Figure 3(a) and 4(a). Note that the reported runtimes of CGA and SPCGA include the time used to detect communities, which are 8,702 seconds and 1,714 seconds, respectively. Note that we use logarithmic scale for y-axis in Figure 4(a).

It can be seen in Figure 3(a) that 1) the influence degree of CGA is close to those of MixedGreedy and NewGreedy (the curve of MixedGreedy is almost overlapping with that of NewGreedy) at all values of $K$, and 2) the influence degree of CGA is much better than those of two heuristic methods DegreeDiscount and Random. For example, at $K$= 20, the influence degree of CGA is 89.96% of that of MixedGreedy while the influence degree of DegreeDiscount (resp. Random) is only 57.62% (resp. 55.72%) of that of MixedGreedy.

We also observe in Figure 4(a) that CGA is orders of magnitude faster than MixedGreedy. For example, at $K$= 20, CGA is 14.56 times faster than MixedGreedy. This is because CGA finds influential nodes in several communities, following the *divide-and-conquer* principle; however MixedGreedy is based on the whole network. We also find that the speedup factor increases as we increase the value of $K$.

As expected, the two heuristic methods run much faster than all greedy algorithms, especially Random needs only around 3 seconds. However, they perform poorly in terms of influence degree. This is consistent with the experimental results reported in previous work[4, 10]—"significantly better marketing results can be obtained by explicitly considering the dynamics of information in a network, rather than relying solely on structural properties of the graph[10]." This would also explain why the greedy algorithms are proposed and used for influence maximization in previous work although heuristic methods run faster. The heuristic methods do not take information influence model into consideration, and they cannot guarantee the precision.

We find that CGA outperforms SPCGA in terms of influence degree while their runtime is similar. The only difference between them is community detection method. Hence, this demonstrates the usefulness of the proposed community detection method of CGA in finding top-$K$ influential nodes.

### 5.2.3 Varying the diffusion speed $\overline{\lambda}$

This experiment is to compare with other algorithms in terms of influence degree and efficiency when we vary the average diffusion speed $\overline{\lambda}$. We vary $\overline{\lambda}$ from 0.01 to 0.10 (0.01, 0.1 are used in [4,12]). This experiment is conducted on the whole network with 723,201 nodes and we set $K$ = 20. Figures 3(b) and 4(b) show the influence degree and runtime of different algorithms, respectively. Note that we use logarithmic scale for y-axis in Figure 4(b).

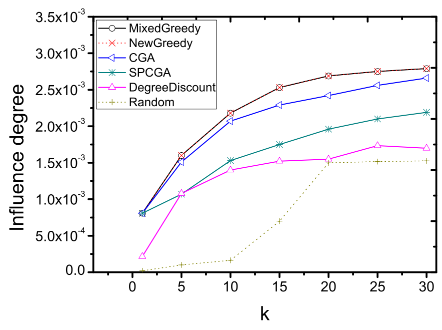
Figure 3(b) shows that the influence degree of CGA is very close to the influence degree of MixedGreedy and NewGreedy at all values of $\overline{\lambda}$. We observe that the influence degree of the two methods is nearly the same when $\overline{\lambda} < 0.07$ (although it cannot be seen clearly in the figure); the influence degree disparity between CGA and MixedGreedy becomes a bit larger as we increase the diffusion speed. This is because as the influence speed increases, a node in one community is more likely to influence the nodes in other communities, and thus the approximation of CGA would become less accurate.

Figure 4(b) shows that CGA runs orders of magnitude faster than MixedGreedy. We also find that when we increase $\overline{\lambda}$ the efficiency of MixedGreedy drops quickly (almost exponentially) while the runtime of CGA only increases slightly.

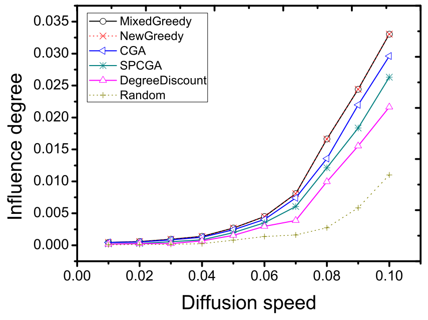
As the size of community is much smaller than the size of the whole network, increasing $\overline{\lambda}$ would incur much more computation on the whole network than each community. This would explain why the runtime of MixedGreedy increases nearly exponentially while the runtime of CGA only increases moderately. When $\overline{\lambda}$ is smaller than 0.02, our CGA costs more time than MixedGreedy, this is because our CGA includes community detection time, the community detection time is relative large when $\overline{\lambda}$ is very small. So CGA shows its advantages when $\overline{\lambda}$ is larger than 0.02. When $\overline{\lambda}$ is larger than 0.06, experiments on MixedGreedy and NewGreedy cannot complete in 10 days, and thus we do not give the runtime in Figure 4(b).

It can be seen that the influence degree (precision) of DegreeDiscount and Random is much worse than those of greedy algorithms. For example, at $\overline{\lambda} = 0.01$, the influence degree of DegreeDiscount (resp. Random) is 40.42% (resp. 29.87%) of that of MixedGreedy while CGA is 95.44% of MixedGreedy.

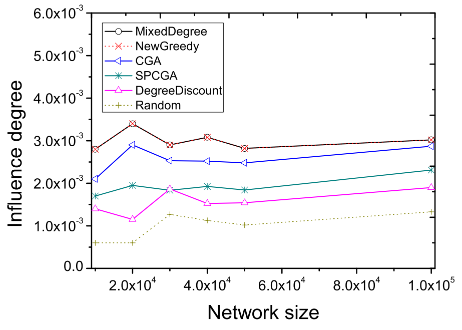
As expected, DegreeDiscount and Random are much faster than all greedy algorithms that take into account Independent Cascade model. However, their precision is much worse than that of greedy algorithms, rendering the heuristic algorithms less attractive than greedy algorithms [10].



(a) Varying the size of results $K$

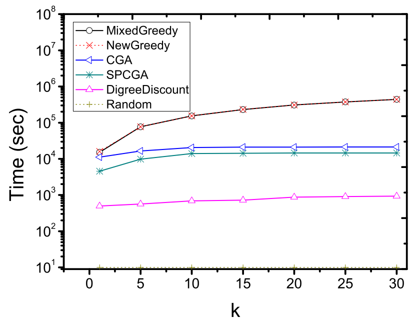


(b) Varying $\overline{\lambda}$

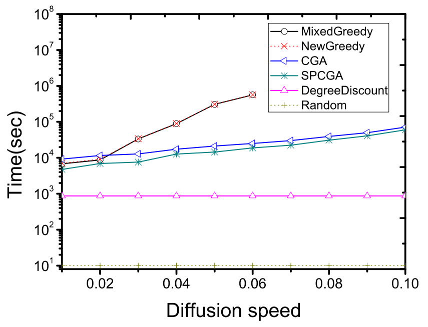


(c) Scalability

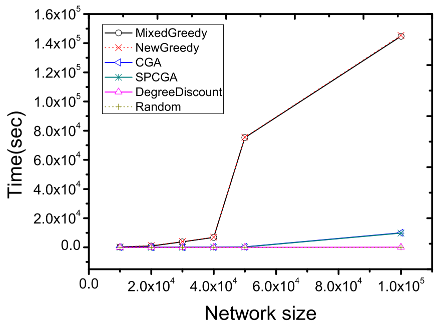
**Figure 3: Influence Degree**



(a) Varying the size of results $K$



(b) Varying $\overline{\lambda}$



(c) Scalability

**Figure 4: Runtime**

**Table 2: Community number $M$ *versus* $\theta$**

| $\theta$ | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 |
|---|---|---|---|---|---|---|---|
| $M$ | 1 | 9 | 25 | 36 | 55 | 86 | 127 |

### 5.2.4 *Scalability with the network size $N$*

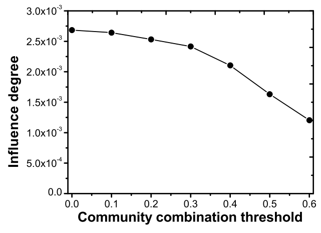
The objective of this experiment is to evaluate the scalability of CGA and other algorithms when we increase the network size. We set diffusion speed $\overline{\lambda}$ at 0.05 and $K$ at 20. Figure 3(c) and Figure 4(c) show the results of influence degree and runtime, respectively, when we vary the network size $N$ from 10K to 100K. Here we randomly select a node, and then starting from the node do a bread-first traversal to generate subnetworks of different sizes.

Figure 3(c) shows the influence degree is relatively stable across the networks of different sizes for all the methods. It is consistent with the previous experiments that the precision of the two heuristic methods is much worse than that of greedy algorithms. We can see from Figure 4(c) that the runtime of both CGA and MixedGreedy increases with the increase of network size; however CGA scales better than MixedGreedy.

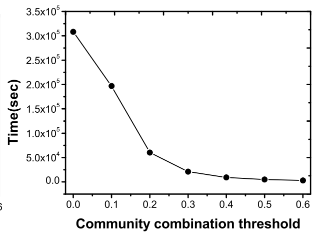
### 5.2.5 *Varying $\theta$*

This experiment is to evaluate the effect of community combination threshold $\theta$ (i.e., the number of network communities in MSN) on the performance of CGA. This experiment is conducted on the whole network with 723,201 nodes and we set $K = 20$ and $\lambda$=0.05. Since this experiment is to study the effect of $\theta$ on CGA, we do not compare with MixedGreedy.

Table 2 gives the number of communities detected when we increase $\theta$ from 0 to 0.6. The results of influence degree and runtime are shown in Figures 5(a) and 5(b), respectively.



(a) Influence degree



(b) Runtime

**Figure 5: Varying $\theta$**

As shown in Figure 5(a), the influence degree drops slowly as we increase $\theta$. The reason is that CGA finds influential nodes within a community and does not consider the influences on nodes in other communities; thus as the network is partitioned into more communities, the precision of CGA will drop. Figure 5(b) shows that CGA runs faster with the increase of $\theta$. The reason is that with the increase of $\theta$, more communities will be generated and the average size of community will be smaller, thus CGA will run faster.

We observe that $\theta$ plays a tradeoff between the runtime and influence degree: with the increase of $\theta$, CGA achieves better efficiency but at the price of slight loss of influence degree. For example, when we increase $\theta$ from 0.1 to 0.5, CGA runs about 39.63 times faster, while the influence spread drops about 38.25%. This is consistent with our theoretical analysis in Section 4. Note that when $\theta$ is 0, CGA works on the whole network and is actually the same as MixedGreedy.

## 6. CONCLUSIONS AND FUTURE WORK

In this paper we propose a new algorithm called Community-based Greedy algorithm for mining top-$K$ influential nodes in a MSN. We first extend the basic Independent Cascade model to take weight edge of MSN into consideration. CGA has two main components, an algorithm for detecting communities by taking into account information diffusion, and a dynamic programming algorithm for selecting communities to find influential nodes. We also provide provable approximation guarantees for CGA. Empirical studies on a large real-world mobile social network show that our algorithm is more than an order of magnitudes faster than the state-of-the-art Greedy algorithm for finding top-K influential nodes and the error of CGA is small compared with Greedy algorithm.

This work opens to several interesting directions for future work. Notably, it is relevant to take spatial information of mobile customers into consideration, and construct locations based social networks to find influential nodes; it is also interesting to study the evolution of influential nodes over time.

## 7. ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (60703066, 60874082), and Beijing municipal natural science foundation (4102026).

## 8. REFERENCES

[1] R. Albert and A. L. Barabasi. Statistical mechanics of complex networks. In Reviews of Modern Physics, 74, pages 47-97, 2002.

[2] Y. Cai, G. Cong, X. Jia, H. Liu, J. He, J. Lu, and X. Du. Efficient algorithm for computing link-based similarity in real world networks. In ICDM, pages 734-739, 2009.

[3] J. Chen, O. R. Zaiane, and R. Goebel. Detecting communities in social networks using max-min modularity. In SIAM, 2009.

[4] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In KDD, pages 199-208, 2009.

[5] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. Physical Review E, 70, 2004.

[6] P. Domingos and M. Richardson. Mining the network value of customers. In KDD, pages 57-66, 2001.

[7] Z. B. Dong, G. J. Song, K. Q. Xie, and J. Y. Wang. An experimental study of large-scale mobile social network. In WWW, pages 1175-1176, 2009.

[8] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. In Proc.Natl.Acad.Sci.USA 99, pages 7821-7826, 2002.

[9] D. Kempe, J. Kleinberg, and E. Tardos. Influential nodes in a diffusion model for social networks. In International colloquium on automata, languages and programming No32, pages 1127-1138, 2005.

[10] D. Kempel, J. Kleinberg, and E. Tardos. Maximizing the spread of inffluence through a social network. In ACM SIGKDD, pages 137-146, 2003.

[11] M. Kimura and K. Saito. Tractable models for information diffusion in social networks. In PKDD 2006, LNAI 4213, pages 259-271, 2006.

[12] M. Kimura, K. Saito, and R. Nakano. Extracting influential nodes for information diffusion on social network. In AAAI, pages 1371-1376, 2007.

[13] M. Kurucz, A. Benczur, K. Csalogany, and L. Lukacs. Spectral clustering in telephone call graphs. In SNAKDD, pages 82-91, 2007.

[14] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. S. Glance. Cost-effective outbreak detection in networks. In In Proc. SIGKDD, pages 420-429, 2007.

[15] D. Lopez-Pintado. Diffusion in complex social networks. In Journal of Economic Literature, pages 573-590, 2004.

[16] H. Ma, H. Yang, M. R. Lyu, and I. King. Mining social networks using heat diffusion processes for marketing candidates selection. In CIKM, pages 233-242, 2008.

[17] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. Physical Review E, 69, 2004.

[18] U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. In Phys.Rev.E76, 2007.

[19] S. Redner. How popular is your paper? An empirical study of the citation distribution. European Physical Journal B, 4:131-134, Aug. 1998.

[20] J. Scripps, P. N. Tan, and A. H. Esfahanian. Exploration of link structure and community-based node roles in network analysis. In Data Mining, IEEE International Conference, pages 649-654, 2007.

[21] J. Scripps, P. N. Tan, and A. H. Esfahanian. Node roles and community structure in networks. In Joint 9th WEBKDD, pages 26-35, 2007.

[22] J. Shi and J. Malik. Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell., 22(8):888-905, 2000.

[23] L. Wan, J. Liao, and X. Zhu. Cdpm: Finding and evaluating community structure in social networks. In ADMA, pages 620-627, 2008.

[24] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger. Scan: A structural clustering algorithm for networks. In KDD, pages 824-833, 2007.