Information Diffusion in Mobile Social Networks: The Speed Perspective

Zongqing Lu[†], Yonggang Wen[‡] and Guohong Cao[†]
[†]The Pennsylvania State University, [‡]Nanyang Technological University
[†]{zongqing, gcao}@cse.psu.edu, [‡]ygwen@ntu.edu.sg

***Abstract*—The emerging of mobile social networks opens opportunities for viral marketing. However, before fully utilizing mobile social networks as a platform for viral marketing, many challenges have to be addressed. In this paper, we address the problem of identifying a small number of individuals through whom the information can be diffused to the network as soon as possible, referred to as the *diffusion minimization* problem. Diffusion minimization under the probabilistic diffusion model can be formulated as an asymmetric k-center problem which is NP-hard, and the best known approximation algorithm for the asymmetric k-center problem has approximation ratio of $\log^* n$ and time complexity $O(n^5)$. Clearly, the performance and the time complexity of the approximation algorithm are not satisfiable in large-scale mobile social networks. To deal with this problem, we propose a community based algorithm and a distributed set-cover algorithm. The performance of the proposed algorithms is evaluated by extensive experiments on both synthetic networks and a real trace. The results show that the community based algorithm has the best performance in both synthetic networks and the real trace, and the distributed set-cover algorithm outperforms the approximation algorithm in the real trace in terms of diffusion time.**

I. Introduction

Social network plays an important role for spreading information, idea and influence among its members. Nowadays, social networks have been evolving to online social networks such as Facebook, Twitter, and Google+ that link humans, computers and the Internet, and information spreading in social networks has been changed from the way of "word-of-mouth" [1] to "word-of-text", "word-of-voice", "word-of-photo" and "word-of-video". In addition, with the proliferation of smart mobile devices, such as smartphone and tablet, people can easily go online with their mobile devices, meanwhile more and more native mobile social networks have been created like Foursquare, Instagram, and Path. Moreover, Bluetooth and Wi-Fi Direct extend communications between mobile devices from the restrictions of cellular infrastructure; user mobility and social connectivity bring numerous ad-hoc communication opportunities.

The emerging of mobile social networks opens opportunities for viral marketing [2]. Different from traditional televised or roadside-billboard advertising campaign, viral marketing takes advantage of the power of "word-of-mouth" to increase brand awareness or product sale through self-replicating viral processes, and it has attracted considerable attentions from mobile and social computing research society [3][4]. However, before fully utilizing mobile social network as a platform for viral marketing, many challenges have to be addressed.

As the essence of viral marketing applications is information diffusion from a small number of individuals to the entire network by "word-of-mouth", in this paper, we address the problem of identifying a small number of individuals through whom the information can be diffused to the entire network as soon as possible, referred to as the *diffusion minimization* problem. Diffusion minimization is naturally critical to viral marketing applications. For example, the "word-of-mouth" advertisement [4] should be disseminated to the network as soon as possible, and thus it would be of interest to many companies as well as individuals that want to increase brand awareness, or disseminate advertisements or innovative ideas through "word-of-mouth".

Diffusion minimization under the probabilistic diffusion model can be formulated as an asymmetric k-center problem which is NP-hard, and the best known approximation algorithm for the asymmetric k-center problem has approximation ratio of $\log^* n$ and time complexity $O(n^5)$ [5], where n is the number of nodes and $\log^* n$ is the iterated logarithm of n. Obviously, the performance and the time complexity of the approximation algorithm are not satisfiable in large-scale social networks. To deal with this problem, we design a community based algorithm with better performance and less time complexity. Different from existing approximation algorithms, the community based algorithm, from the social point of view, leverages the community structure to solve the diffusion minimization problem. Due to the lack of global information and the requirement to handle the dynamic evolving of mobile social networks, we further propose a distributed set-cover algorithm, where each node collects up-to-date information by probing messages in a distributed way. The performance of these algorithms is evaluated based on both synthetic networks generated by a well-known benchmark and a real trace. Simulation results show that the community based algorithm has the best performance in both synthetic networks and real trace, and the distributed set-cover algorithm outperforms the approximation algorithm in the real trace in terms of diffusion time.

The rest of this paper is organized as follows. Section II reviews related work. Section III gives the problem statement.

This work was supported in part by Network Science CTA under grant W911NF-09-2-0053.

978-1-4799-3360-0/14/$31.00

The community based algorithm is presented in Section IV, followed by the distributed set-cover algorithm in Section V. Section VI evaluates the performance of the proposed algorithms and Section VII concludes the paper.

II. Related Work

A. Information Diffusion

With the emerging of online social media, information diffusion has been extensively studied based on emails [6], Facebook [7], and Twitter [8]. One salient feature of information diffusion is the correlation between the number of friends engaging in spreading information and the probability of adopting the information [7].

Recently, a lot of research efforts focus on whether and how individuals influence each other. Domingos and Richardson [9] are the first to study the influence maximization problem and give a probabilistic solution. Kempe *et al.* [10] designed a greedy algorithm with approximation ratio of $(1 - \frac{1}{e})$. Recently, Wang *et al.* [11] presented a community based greedy algorithm and Jiang *et al.* [12] proposed a simulated annealing algorithm to address the influence maximization problem. Different from the influence maximization problem which studies how individuals influence each other and how to maximize the influence in social networks, the diffusion minimization problem investigates how information spreads and how to minimize the diffusion time.

B. Mobile Social Networks

Through mobile social networks, individuals with similar interests interact, communicate and connect with others through their mobile devices such as smartphones, tablets, etc. With the proliferation of smartphones, mobile social network has emerged as a new frontier in mobile computing research, and lots of research has focused on mobile social networks [13][14][15]. Moreover, many mobile social applications have been developed such as Micro-blog [16], SociableSense [17], etc.

Mobile social network is a fertile ground for the rapid spreading of information including text, photo, voice and video. Thus, information dissemination is an important problem in mobile social networks. Han *et al.* [15] designed a distributed random walk protocol for immunization of infectious diseases and information dissemination. Hu *et al.* [18] proposed an energy-aware user-contact detection algorithm through Bluetooth on smartphones. Peng *et al.* [4] addressed users' selfishness and privacy concerns for viral marketing. Ning *et al.* [3] proposed an incentive scheme to stimulate cooperation among selfish nodes for data dissemination. However, none of them considers the diffusion minimization problem.

III. Preliminaries, Problem Statement and Naïve Algorithm

A. Mobile Social Network

Let $G = (V, E)$ represent a weighted and undirected mobile social network, where V denotes the set of nodes with cardinality n and E denotes the set of edges. For two neighboring nodes $u, v \in V$, w_{uv} denotes the weight of the edge and $w_{uv} = w_{vu}$. The edge weight indicates the frequency of contacts between two nodes. For a node $u \in V$, d_u is the degree of node u and N_u is the neighbor set of u, and we have $d_u = \sum_{v \in N_u} w_{uv}$.

B. Probabilistic Diffusion Model

In the operational model of information diffusion, each node can be either *active* or *inactive*. Active nodes are the adopters of the information and are ready to diffuse the information to their inactive neighbors. The state of a node can be switched from inactive to active, but not the other way around. More specifically, when an active node u contacts an inactive node v, v becomes active with some probability $\lambda_{uv} = \frac{w_{uv}}{d_u}$. This is because the probability of information spreading from node u to the neighboring node v should be proportional to the connection fraction of node v over the degree of u. In other words, the more frequently node u contacts with node v, the more likely node v gets informed and becomes active. From the social relation point of view, a person most likely shares the information with his best friends rather than others. Different from the *linear threshold model* [10] and the *independent cascade model* [19] that describe how individuals influence each other in social networks, the *probabilistic diffusion model* describes how the information diffuses in social networks.

C. Problem Statement

The information diffusion process can be described as follows. First an initial set of active nodes is selected. When the contact happens between an active node and an inactive node, the inactive node becomes active with a probability. The process terminates when all the nodes are active.

Let S be the initial set of active nodes. The diffusion time of initially selected node set is defined as the time interval between the start and the end of the information diffusion process denoted by $\tau(S, V)$.

Given a weighted network $G = (V, E)$ and an integer k, we aim to identify a node set S, $|S| \leq k$ and $S \subseteq V$, such that $\tau(S, V)$ is minimum. This problem is referred to as the *diffusion minimization* problem and nodes in S are referred to as the *diffusion nodes*.

Under the probabilistic diffusion model, using the edge weight w_{uv} as the contact frequency in social network, the expected information diffusion time from node u (active) to neighboring node v (inactive) can be formulated as

$$t_{uv} = \frac{1}{\lambda_{uv}} \cdot \frac{1}{w_{uv}} = \frac{d_u}{w_{uv}^2}, \tag{1}$$

where $\frac{1}{w_{uv}}$ denotes the average time interval between contacts. Similarly, we have $t_{vu} = \frac{d_v}{w_{uv}^2}$ from node v to node u (the expected diffusion time from u to v and that from v to u are different, except $d_u = d_v$). For any pair of nodes, for example node u and v, the shortest expected diffusion time from u to v is denoted as $|(u, v)|$ and for simplicity we also call $|(u, v)|$ the expected diffusion time from u to v.

Since the diffusion time between any pair of nodes can be estimated by the expected diffusion time, the diffusion

minimization problem under the probabilistic diffusion model can be mathematically formulated as finding a subset $S \subseteq V$ with $|S| \leq k$ to minimize the expected diffusion time $\tau'(S, V)$:

$$\tau'(S, V) = \min \max_{v \in V} |(S, v)|, \tag{2}$$

where

$$|(S, v)| = \min_{u \in S} |(u, v)| \tag{3}$$

and $|(S, v)|$ is the expected diffusion time from set S to node v.

As $\exists u, v \in V, t_{uv} \neq t_{vu}$, the problem is the same as the *asymmetric k-center* problem, which is NP-hard. There is an approximation algorithm known for the asymmetric k-center problem with approximation ratio $\log^* n$ [5] and asymmetric k-center is $\log^* n$-hard to approximate [20]. Moreover, the time complexity of the approximation algorithm is $O(n^5)$. Therefore, the performance and the time complexity of the approximation algorithm are not satisfiable in large-scale social networks. Thus, we design better algorithms.

D. A Naïve Algorithm

The *closeness* (also known as closeness centrality) of a node is defined as the reciprocal of the sum of the shortest distances to all other nodes in the network. When applied to the probabilistic diffusion model, the closeness of node u can be denoted as $1/\sum_{v \in V} |(u, v)|$.

Closeness is a measure of how fast it will take to spread information from a node to all other nodes [21]. With regard to identifying S from V, a naïve solution for the diffusion minimization problem can be based on closeness; i.e., iteratively select the node with the highest closeness from the set of unselected nodes (i.e., $V \backslash S$) until $|S| = k$. More specifically, the closeness of node u at each iteration is calculated as

$$\frac{1}{\sum_{v \in V \backslash S} |(u, v)|}, \ u \notin S.$$

However, the naïve algorithm does not work well (as shown in the evaluation section), and hence we propose better algorithms.

IV. Community based Algorithm

Community represents a set of nodes in a network, where nodes inside the community have more internal connections than external connections [22][23][24]. Community structure is a prominent network property which provides a clear view of how nodes are organized and how nodes contact with each other, especially in social networks.

For information diffusion in mobile social network, community has the following properties. Within a community, nodes frequently contact each other and hence information can be quickly spread. Information diffusion from one community to another community is much slower compared to that within community.

The basic idea of the community based algorithm is to identify at least one diffusion node from each community. Let $\mathcal{C} = \{C_1, C_2, C_3, ...C_l\}$ denote the community structure, where $|\mathcal{C}| = l$ and $C_i \in \mathcal{C}$ denotes a community and $C_i \subseteq V$. For simplicity, we also denote C_i as C if there is no confusion. As k nodes need to be identified from $\mathcal{C}$, there are two cases: $k < l$ and $k \geq l$. For $k < l$, we cannot guarantee one diffusion node for each community, so some communities should be merged. For $k \geq l$, we need to consider how to identify more than one diffusion node in a community. Thus, in following we first show how to merge the detected communities to ensure $k \geq l$, and then study how to identify multiple diffusion nodes in a community.

A. Community Merge

Before getting into the details of merging communities, we first introduce two terms: *central node* and *diffusion radius*.

Definition 1. *The **Central Node** of a community is defined as the node from which the expected diffusion time to all other nodes in the community is minimum. The expected diffusion time of the central node is defined as the **Diffusion Radius** of the community.*

Let N_C and $R(C)$ denote the *central node* and the *diffusion radius* of community C, respectively, and we have

$$R(C) = \min_{u \in C} \left(\max_{v \in C} |(u, v)| \right).$$

By merging communities, the number of communities can be reduced from l to k. After that, the expected diffusion time of the network is determined by the community with the maximum diffusion radius since we will identify one diffusion node in each community. Thus, for community merge, we should minimize $\max\{R(C) : C \in \mathcal{C}\}$ after $(l - k)$ merging steps. Clearly, we have $R(C_i \cup C_j) > \max\{R(C_i), R(C_j)\}$. Thus, the merge of communities will increase the diffusion radius. Since, at each step, we merge two communities together, we have $\binom{|\mathcal{C}|}{2}$ choices to merge communities. In order to minimize $\max\{R(C) : C \in \mathcal{C}\}$ after $(l - k)$ merging steps, we have to search $\binom{l}{2}\binom{l-1}{2}\dots\binom{k+1}{2}$ times. However, the running time is $O(l^{l-k})$ and it is too expensive for large community structure. Thus, we propose an alternative approach for community merge.

For two rarely or indirectly connected communities, the merged community will have an unexpected large diffusion radius. In contrast, for two closely connected communities, the diffusion radius of the merged community may be more than the maximum of the two individual communities. Thus, the basic idea is to merge closely connected communities and make the diffusion radius of newly formed community as small as possible. C_j is a closely connected community to C_i if the sum of the edge weights between C_i and C_j normalized by $|C_j|$ is no less than that between C_i and $V \backslash C_i$. The set of closely connected communities of C_i ($C_i \in \mathcal{C}$) is denoted by $\mathcal{C}_{C_i}$, and we have

$$\mathcal{C}_{C_i} = \{C_j \in \mathcal{C} \backslash \{C_i\} : \frac{\sum_{u \in C_i, v \in C_j} w_{uv}}{|C_j|} \geq \frac{\sum_{u \in C_i, v \in V \backslash C_i} w_{uv}}{|V \backslash C_i|}\}$$

Algorithm 1: Community Merging

Input : $\mathcal{C}$
Output: $\mathcal{C}$
while $|\mathcal{C}| > k$ **do**
  $\mathcal{C}' = \mathcal{C}$
  $C_i = C_j = \emptyset$
  while $|\mathcal{C}'|$ **do**
    $C'_i = \arg\min_{C \in \mathcal{C}'} R(C)$
    $C'_j = \arg\min_{C \in \mathcal{C}'_{C_i}} R(C_i \cup C)$
    if $R(C'_i \cup C'_j) \leq \max\{R(C) : C \in \mathcal{C}\}$ || $C_i = C_j = \emptyset$ **then**
      $C_i = C'_i$ and $C_j = C'_j$
      break
    else
      if $R(C'_i \cup C'_j) < R(C_i \cup C_j)$ **then**
        $C_i = C'_i$ and $C_j = C'_j$
      end
      $\mathcal{C}' = \mathcal{C}' \backslash \{C'_i\}$
    end
  end
  $\mathcal{C} = \mathcal{C} \backslash \{C_i, C_j\} \cup \{C_i \cup C_j\}$
end

The community merging process works as follows. First, we choose the community with the lowest diffusion radius denoted by C_i among $\mathcal{C}$, then merge C_i with C_j, one from the set of closely connected communities $\mathcal{C}_{C_i}$, to obtain the lowest $R(C_i \cup C_j)$. If the diffusion radius of the newly formed community is less than the maximum value in $\mathcal{C}$, we merge them together. Otherwise, all other communities are iterated to find $R(C_i \cup C_j) \leq \max\{R(C) : C \in \mathcal{C}\}$. If we cannot find $R(C_i \cup C_j) \leq \max\{R(C) : C \in \mathcal{C}\}$, C_i and C_j are merged with the lowest $R(C_i \cup C_j)$ in $\mathcal{C}$. Then, $\mathcal{C}$ is updated and the process is iterated until $l - k$ merging steps.

The details of the community merging algorithm is shown in Algorithm 1. With $l-k$ merging steps, there are at most $|\mathcal{C}|-1$ searches for the merging with the lowest diffusion radius at each iteration, and one community has at most $|\mathcal{C}| - 1$ closely connected communities. Thus, the worst time complexity of merging community is $O(l^2(l-k))$.

B. Identifying Diffusion Nodes within Community

After community merge, $|\mathcal{C}| = k$. Thus, the design effort focuses on $k \geq |\mathcal{C}|$. In the rest of this section, $\mathcal{C}$ is either the communities after merge or the detected communities with $k \geq l$. Since $k \geq l$, we need to identify more than one diffusion nodes in a community.

As the expected diffusion time for a node $u \in C_i$ to be informed is $|(S_{C_i}, u)|$, where S_{C_i} denotes the set of diffusion nodes selected within community C_i, to identify multiple diffusion nodes, we iteratively choose the node, which minimizes the sum of the expected diffusion time from the set of selected diffusion nodes to every other node in community C_i, precisely,

$$\arg\min_{u \in C_i \backslash S_{C_i}} \sum_{v \in C_i \backslash S_{C_i}} |(S_{C_i} \cup \{u\}, v)|.$$

With the selection of S_{C_i}, the expected diffusion time of community C_i is denoted as $\tau'(S_{C_i}, C_i)$.

Algorithm 2: Community based Algorithm

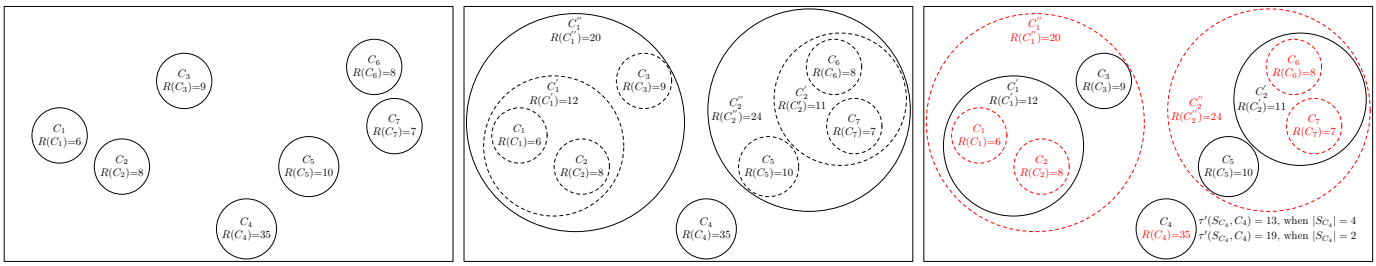
Input : $\mathcal{C}$
Output: S
$\mathcal{C}' = \mathcal{C}$
while $\exists C_i, C_j : R(C_i \cup C_j) < \max_{C \in \mathcal{C}'} R(C)$ **do**
  $\mathcal{C}' = \mathcal{C}' \backslash \{C_i, C_j\} \cup \{C_i \cup C_j\}$
end
$S = \{N_C : C \in \mathcal{C}'\}$
while $|S| < k$ **do**
  $C = \arg\max_{C \in \mathcal{C}'} R(C)$
  if $C \in \mathcal{C}$ **then**
    if $|S_C| = 1$ **then**
      $S = S \backslash \{N_C\}$
      $S = S \cup \{\arg\min_{u \in C \backslash S_C} \sum_{v \in C \backslash S_C} |(S_C \cup \{u\}, v)|\}$
    end
    $S = S \cup \{\arg\min_{u \in C \backslash S_C} \sum_{v \in C \backslash S_C} |(S_C \cup \{u\}, v)|\}$
  else
    $S = S \backslash \{N_C\} \cup \{N_{C_i}, N_{C_j}\}$; // C_i and C_j merged as C
    $\mathcal{C}' = \mathcal{C}' \backslash \{C\} \cup \{C_i, C_j\}$
  end
end

C. Algorithm Design

To select diffusion nodes which can effectively reduce the expected diffusion time, a straightforward solution is to select the central node from each community, and then iteratively choose nodes from the community with $\max\{R(C) : C \in \mathcal{C}\}$ until $|S| = k$. However, this approach is not efficient. Let us give an example. In $\mathcal{C}$, there is a community C_i with diffusion radius larger than that of the merged community of C_j and C_k. For this case, it is better to select more than one diffusion node from C_i and only choose one diffusion node from the merged community of C_j and C_k. Therefore, the designed algorithm should address this problem.

The community based algorithm works as follows. First we use similar technique as in Section IV-A to merge closely connected communities until $\forall C_i, C_j : R(C_i \cup C_j) > \max\{R(C) : C \in \mathcal{C}\}$, and thus we have the updated $\mathcal{C}$. We call it the *merging process*. Then, we choose the central nodes for individual communities as the candidates of diffusion nodes and thus we still have $k - |\mathcal{C}|$ candidates remaining. Next, we identify other candidate within community C_i with the maximum expected diffusion time in the current $\mathcal{C}$. If C_i is a merged community, it is split into two original communities C_j and C_k, and the central nodes of C_j and C_k are chosen as two candidates to replace N_{C_i}. Recall that we have $R(C_i) > \max\{R(C_j), R(C_k)\}$, so the replacement will decrease the expected diffusion time of network. If C_i is an originally detected community, we first choose a candidate, according to the approach described in Section IV-B, to replace N_{C_i}, then identify one more candidate and add it into S. The process is executed iteratively until no candidate remains and we call it the *restoring process*.

Fig. 1 gives an example of the community based algorithm, where $l = 7$ and $k = 8$. Figures 1a to 1b show the merging process. After that, three communities remain: $C_1^{''}$, $C_2^{''}$ and C_4. Figures 1b to 1c show the restoring process. After choosing the central node from each community, as $R(C_1^{''}) = 20$,



(a) detected communities (b) merged communities (c) restored communities

Fig. 1: Illustration of the community based algorithm, where $l = 7$ and $k = 8$. (a) to (b) show the merging process. (b) to (c) show the restoring process. Finally, $S = \{N_{C_1'}, N_{C_2'}, N_{C_3}, N_{C_5}\} \cup S_{C_4}$.

$R(C_2^{''}) = 24$ and $R(C_4) = 35$, we need to identify other diffusion nodes from C_4. After selecting two diffusion nodes from C_4, $\tau'(S_{C_4}, C_4) = 19$. Since $R(C_2^{''}) > \tau'(S_{C_4}, C_4)$, we then switch to $C_2^{''}$. As $C_2^{''}$ is a merged community, we replace $N_{C_2^{''}}$ with N_{C_5} and $N_{C_2'}$. The restoring process continues until $|S| = k$. Finally, we choose $N_{C_1'}$, $N_{C_2'}$, N_{C_3}, N_{C_5} and S_{C_4} ($|S_{C_4}| = 4$) as diffusion nodes, and the expected diffusion time of the network is the maximum of individual communities; that is, $\tau'(S_{C_4}, C_4) = 13$ as shown in Fig. 1c. If we choose the central node for each community, and then identify the rest of diffusion nodes from the community with the maximum diffusion radius, for this example, S will include $N_{C_1}, N_{C_2}, N_{C_3}, N_{C_5}, N_{C_6}, N_{C_7}$ and S_{C_4} ($|S_{C_4}| = 2$), and the expected diffusion time of the network is also determined by C_4, i.e, $\tau'(S_{C_4}, C_4) = 19$ where $|S_{C_4}| = 2$. Thus, our algorithm performs better than the straightforward solution; i.e., the expected diffusion time of our algorithm is 13 which is better than 19. More generally, our algorithm identifies more diffusion nodes in the community which determines the expected diffusion time of the network than the straightforward solution, and $\tau'(S_{C_i}, C_i)$ decreases with the increase of $|S_{C_i}|$ according to the approach in Section IV-B; thus, our algorithm has better performance.

Algorithm 2 details the community based algorithm. For community merge, the worst case is when all the communities are merged into one community and the running time is $O(l^3)$. For the selection of S, the worst case is when all the diffusion nodes are selected by the approach in Section IV-B and the running time is $O(kn^2)$. Thus, the worst time complexity of the community based algorithm is $O(l^3+kn^2)$. However, since l is much less than n, the running time is equivalent to $O(kn^2)$, which is much less than the approximation algorithm ($O(n^5)$).

V. Distributed Set-cover Algorithm

The approximation algorithm and the community based algorithm are centralized and require global information of the network; i.e., pairwise expected diffusion time is required for the approximation algorithm and community structure is required for the community based algorithm. However, such information might not be available or cost too much in some scenarios. Furthermore, mobile social networks might dynamically evolve over time and then the contact frequency between nodes (the edge weight) varies over time, which will affect the accuracy for calculating the pairwise expected diffusion time and detecting the communities. Thus, we propose a distributed set-cover algorithm to address these problems.

For a certain time period γ and a node u, there is a set of nodes to which u can diffuse information within γ, referred to as the *diffusion set* of u. Suppose γ is equal to the minimum diffusion time of the set of diffusion nodes, precisely,

$$\gamma = \min_{\substack{S \subseteq V \\ |S| \leq k}} \max_{v \in V} |(S, v)|,$$

the set of diffusion nodes S can be easily identified by selecting the nodes, where the union of the diffusion sets for the selected nodes is the set of nodes V. Although it is impossible to have the minimized diffusion time, this inspires the design of the distributed set-cover algorithm.

The distributed set-cover algorithm includes two phases: *discovering the diffusion set* and *identifying the k-node set*. For a given γ, which is a system parameter, the first phase leverages probing messages to find the diffusion set for each node in a distributed way; the second phase iteratively selects the node to maximize the union of the diffusion sets for the selected nodes.

A. Discovering the Diffusion Set

The diffusion set is identified as follows. For every period of time Δt, which is a system parameter, each node generates a probing message that includes node *id*, last relay node and time-to-live (TTL), where TTL is set to γ, and stores it in the local message queue. When a node u contacts with node v, u will choose one probing message whose id is not v and whose last relay node is not v from its message queue and forward the message with probability λ_{uv} ($\lambda_{uv} = \frac{w_{uv}}{d_u}$). If node v receives the probing message from u, it will decrease TTL by t_{vu}. After that, if TTL≥ 0, v will add the node *id* contained in the probing message into $\Gamma(v)$, where $\Gamma(v)$ denotes the set of nodes from which node v has received the probing message. Finally, if TTL≤ 0, the probing message is discarded, otherwise, the last relay node of the message is set to v and it is stored into v's local message queue so that it can be forwarded to other nodes. Since TTL of probing messages is initially set to γ and reduced by the expected diffusion time from receiver to sender at each message transfer, for each node u, u can diffuse information to $\Gamma(u)$ within γ and $\Gamma(u)$ is called the *up-to-date*

Algorithm 3: Identifying the k-node Set

Input : V, k
Output: S
$V^{'} = V$
while $|S| < k$ && $V^{'} \neq \emptyset$ **do**
  $u = \arg\max_{v \in V} |V^{'} \cap \Gamma(v)|$
  $V^{'} = V^{'} \backslash \Gamma(v) \backslash \{u\}$
  $S = S \cup \{u\}$
end

diffusion set of node u. Note that w_{uv} and d_u are needed for node u to compute λ_{uv}, and w_{uv} and d_v are needed for node v to calculate t_{vu}. However, it is easy for individual nodes to maintain these information, thus it is omitted here.

B. Identifying the k-node Set

We identify the k-node set based on the collected up-to-date diffusion set for each node. The k-node set is selected as follows. First, we mark $V^{'}$ as a copy of V, and then choose node u from V, which can maximize the intersection of $\Gamma(u)$ and $V^{'}$, i.e., $u = \arg\max\{|\Gamma(v) \cap V^{'}| : v \in V\}$. After that, u and nodes in the intersection of $\Gamma(u)$ and $V^{'}$ are excluded from $V^{'}$. The process is executed iteratively until $|S| = k$ or $V^{'} = \emptyset$. The algorithm of identifying the k-node set is detailed in Algorithm 3 and the worst case of time complexity is $O(kn^2)$.

C. Discussions

When node v (receiver) receives a probing message from node u (sender), node v will reduce TTL of the probing message by the expected diffusion time from v to u. *Why not reduce TTL by the diffusion time of the probing message from u (sender) to v (receiver)?* This is because the diffusion time from sender to receiver is different from that from receiver to sender. As we aim to collect the up-to-date diffusion set at the receiver side, reducing TTL by the diffusion time from sender to receiver is not feasible. Then, one may argue that if the probing message is forwarded from node u to v with probability λ_{vu}, the diffusion time of the probing message from u to v will be equivalent to that from v to u. However, for this case, node u will need d_v to calculate λ_{vu}, which will incur additional message overhead. More importantly, v cannot rely on the diffusion time of the probing message to determine whether u can be reached within γ. For example, if the diffusion time of the probing message from u to v is less than γ, meanwhile the expected diffusion time from v to u is more than γ, u should not be included in the up-to-date diffusion set of v since in most cases v cannot diffuse information to u within γ (i.e., $t_{vu} > \gamma$).

After discovering the diffusion set, each node can collect the up-to-date diffusion set. As the path along which a probing message travels is probably not the shortest path between two nodes in terms of expected diffusion time, *is the up-to-date diffusion set the same as the diffusion set for each node?* Although probing messages are likely to stay within or gather at certain region according to the probability of information diffusion between neighboring nodes, the diffusion set of a node is expectedly fully discovered within time γ. However, that requires generating and forwarding the probing message more frequently and hence results in high message overhead. Thus, in our solution, each node generates a message every Δt and forwards only one message opportunistically upon node contact with a probability. Although the diffusion set may not be disclosed completely, the up-to-date diffusion set approaches to the diffusion set over time.

There are two system parameters for the distributed set-cover algorithm: the time period (γ) and the frequency of generating probing message (Δt). γ determines the region that the probing message can spread and hence affects both performance and cost. Δt determines the number of probing messages spreading over the network and also affects performance and cost. Generally speaking, less generated messages will result in smaller up-to-date diffusion set and more generated messages will have more chances to block the relayed probing message (since at most one probing message is forwarded upon node contact), leading to smaller up-to-date diffusion set as well. Thus, there is a tradeoff for selecting the values of γ and Δt, which will result in tradeoffs between improving the performance and reducing the message overhead. In the next section, we will show how γ and Δt affect the performance of the distributed set-cover algorithm and how to achieve balance between performance and cost.

VI. Performance Evaluations

In this section, we evaluate the performance of the proposed algorithms based on synthetic networks and the Facebook trace.

A. Comparisons based on Synthetic Networks

In this section, we compare the community based algorithm (*Community*), the approximation algorithm (*Approximation*) proposed in [5] and the naïve algorithm (*Naïve*), in terms of expected diffusion time. For *Community*, we use the community detection algorithm proposed in [23]. Note that *Community* does not limit the selection of the detection algorithm and it is compatible with other detection algorithms.

In order to evaluate the performance for different network settings, we use the synthetic networks generated by the well-known benchmark proposed in [25]. It provides power-law distribution of node degree and edge weight, and various topology control. There are several parameters to control the generated network: the number of nodes n; the average neighbors α; the maximum neighbors α_{max}; the mixing parameter for the weights μ_w; the mixing parameter for the topology μ_t; the exponent for the weight distribution β; the minus exponent for the degree sequence ξ_1; the minus exponent for the community size distribution ξ_2. Without loss of generality, the settings of these parameters are close to [23], which are shown in Table I.

The expected diffusion time is formulated as the time unit of the contact frequency. For example, if the edge weight is formulated as contacts of one month, the time unit of the expected diffusion time is one month.

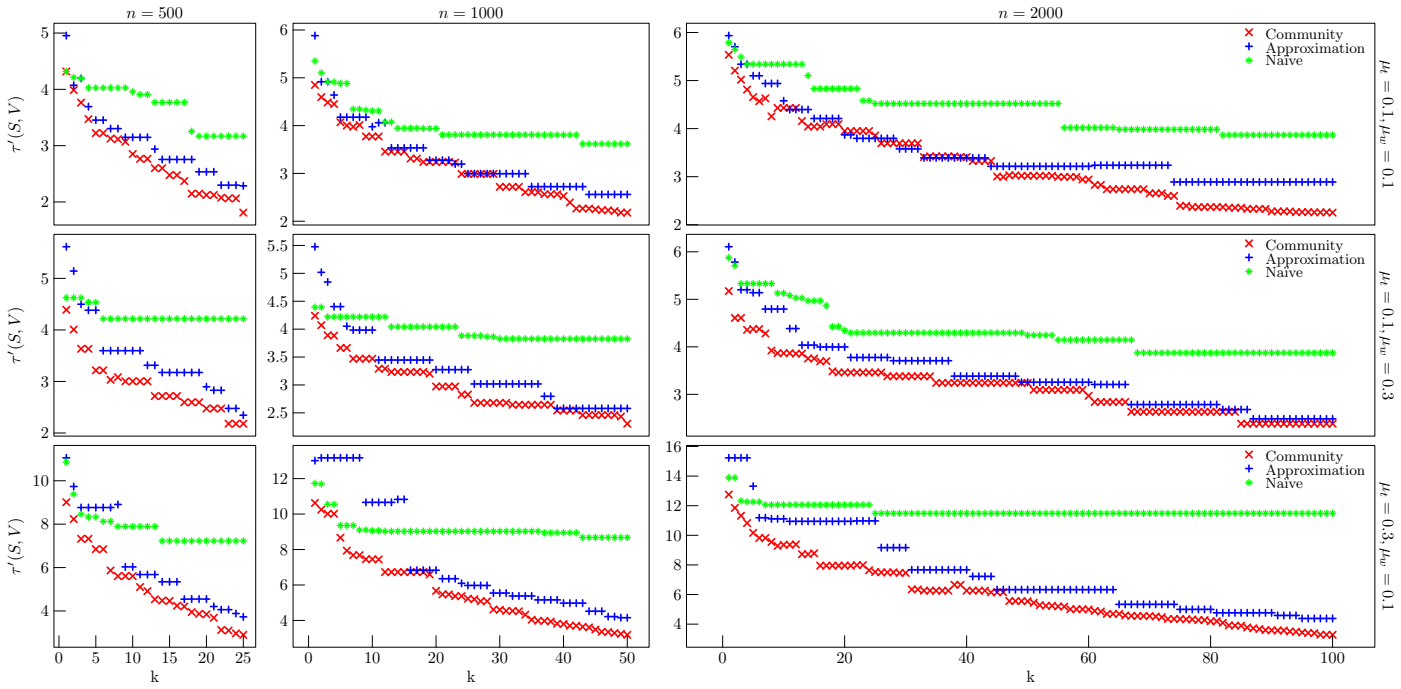


Fig. 2: Expected diffusion time of ***Community***, ***Approximation*** and ***Naïve*** with different network settings.

TABLE I: Parameter settings for the benchmark

Parameter	Value	Parameter	Value
n	500, 1000, 2000	ξ_1	2
μ_w	0.1, 0.3	ξ_2	1
μ_t	0.1, 0.3	α	15
β	2	α_{max}	20

Fig. 2 shows the expected diffusion time of *Community*, *Approximation* and *Naïve* with varying k for different network settings. Since in real applications k should be a small value, we choose k no more than 5% of the network size for each setting. As shown in Fig. 2, for network $\mu_t = 0.1$ and $\mu_w = 0.1$, *Community* and *Approximation* are comparable when k is small. However, with the increase of k, *Community* increasingly outperforms *Approximation*. In addition, both *Community* and *Approximation* are much better than *Naïve*. For networks with heterogeneity in edge weight, where $\mu_t = 0.1$ and $\mu_w = 0.3$, *Community* outperforms the other two algorithms for all k values and *Naïve* still has the worst performance. *Community* always has better performance, up to 40%, than *Approximation* in different network sizes for more heterogeneous network in topology, where $\mu_t = 0.3$ and $\mu_w = 0.1$. In addition, the expected diffusion time of *Naïve* does not change too much when k increases, compared to *Approximation* and *Community*, which means that selecting more diffusion nodes does not help too much in *Naïve*, especially in networks with more heterogeneity in topology.

In summary, *Community* performs better than *Approximation* and *Naïve* in terms of expected diffusion time, because *Community* relies on the community structure and identifies diffusion nodes from individual communities rather than the entire network as in *Approximation* and *Naïve*.

B. Estimations of γ and Δt

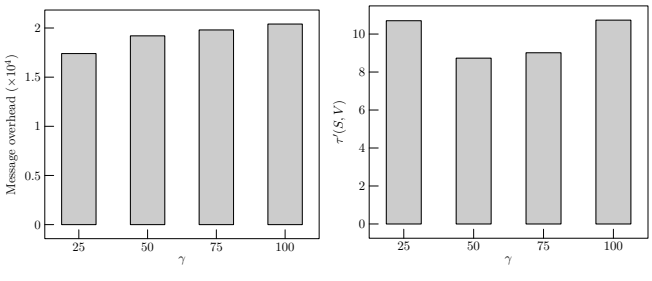
In this section, we estimate γ and Δt for the distributed set-cover algorithm (*Set-cover*) based on the Facebook trace [26], which contains friendship information and wall posts among Facebook users in the New Orleans regional network for more than two years. We choose a partial trace, which spans from Jan 2007 to Jan 2009 and contains 2320 nodes. We formulate the contact between two nodes as the wall post and the edge weight as the contact frequency.

As the neighboring information is needed for estimating γ, Δt, and the diffusion set, we use 3-month trace from Jan 2007 to April 2007 to construct the neighboring information for each node including the set of neighboring nodes, the node degree and the edge weight for each neighbor. The diffusion set discovery algorithm runs on other 3-month trace from April 2007 to July 2007 and the neighboring information is kept updated during this period. γ and Δt are the system parameters of *Set-cover*, which impact the performance and the cost.

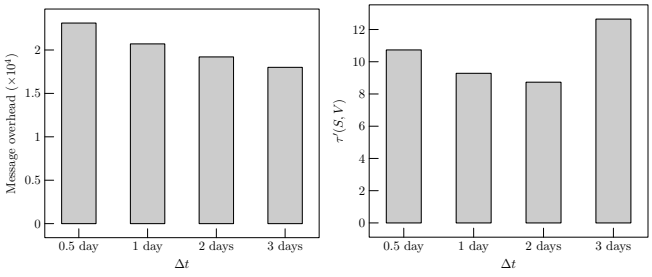
As γ determines the range that a probing messages can spread, it corresponds to k and certain network properties. We use the average expected diffusion time between neighboring nodes t_a, the average number of neighboring nodes $|N_a|$ and k to estimate γ. Assuming that each selected diffusion node can spread information to the same number of nodes (that is $\frac{n}{k}$ for each diffusion node), we set γ to the expected diffusion time to spread information to $\frac{n}{k}$ nodes. That is

$$\gamma = t_a \lceil \log_{|N_a|} \frac{n}{k} \rceil, \tag{4}$$

where $\lceil \log_{|N_{avg}|} \frac{n}{k} \rceil$ is the minimum hop to reach $\frac{n}{k}$ nodes from the diffusion node. Figs. 3a and 3b show the message overhead (the number of message transfers) and the expected diffusion



(a) message overhead vs γ (b) expected diffusion time vs γ



(c) message overhead vs Δt (d) expected diffusion time vs Δt

Fig. 3: Performance of ***Set-cover*** with varying γ and Δt in terms of message overhead and expected diffusion time, where $k = 50$, Δt =2 days for (a) and (b), and $k = 50$, $\gamma = 50$ for (c) and (d).

time of *Set-cover* for $k = 50$, $\Delta t = 2$ days with varying γ, where the estimated γ is 50 by Eq. 4. As can be seen, although the message overhead of $\gamma = 25$ is slightly less than $\gamma = 50$, $\gamma = 50$ has better diffusion time. Moreover, $\gamma = 50$ is a better choice than $\gamma = 75$, since the message overhead of $\gamma = 50$ is less than $\gamma = 75$, though they have similar diffusion time. Thus, our estimation of γ achieves a good balance between performance and cost.

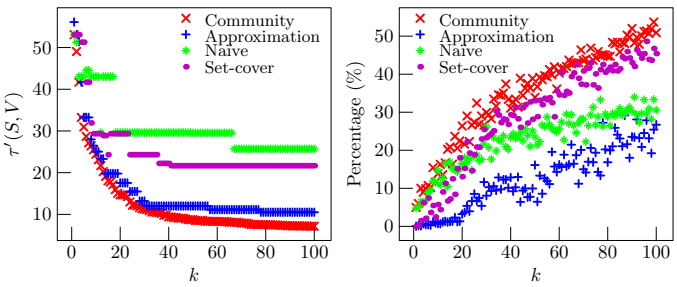
For Δt, we use the average node degree d_a and the time unit of contact frequency T to estimate Δt. We set Δt equal to the average time interval between contacts. That is

$$\Delta t = \lceil \frac{T}{d_a} \rceil. \quad (5)$$

Figs. 3c and 3d show the message overhead and the expected diffusion time of *Set-cover* for $k = 50$, $\gamma = 50$ with varying Δt, where the estimated Δt is 2 days by Eq. 5. Fig. 3c shows that the message overhead decreases with the increase of Δt. Although $\Delta t = 2$ days has little more message overhead than $\Delta t = 3$ days, it has much better performance as shown in Fig. 3d. Thus, the estimation of Δt achieves a good tradeoff between performance and cost.

C. Comparisons based on the Facebook Trace

We also evaluate the performance of the proposed algorithms based on the Facebook trace. *Community*, *Approximation* and *Naïve* are centralized algorithms which require the server to continuously collect and maintain global information. In contrast, *Set-cover* only needs to collect a few neighboring information for estimating γ and Δt, and the up-to-date diffusion



(a) expected diffusion time (b) percentage of active nodes

Fig. 4: Performance on the Facebook trace of ***Community***, ***Approximation***, ***Naïve*** and ***Set-cover***.

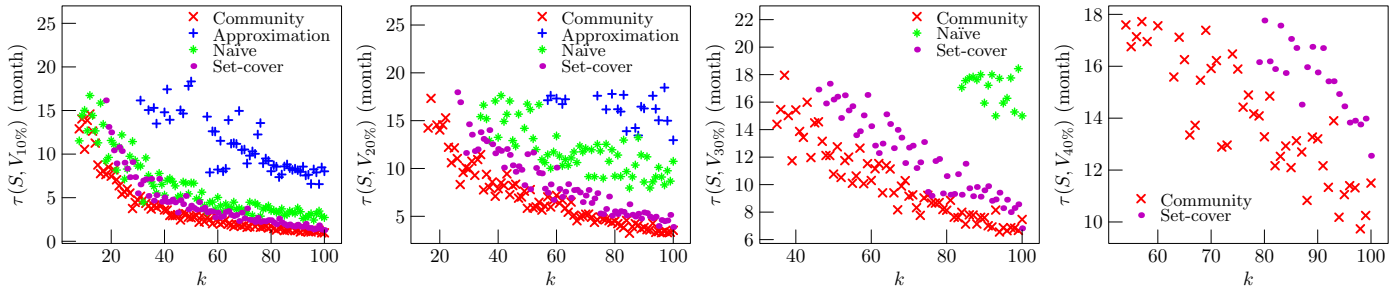
set. Although *Set-cover* requires probing messages, each probing message only transfers between neighboring nodes with a probability when contact happens. Thus, the message overhead of *Set-cover* is much less than the centralized algorithms.

Fig. 4a shows the expected diffusion time, where γ and Δt are chosen according to Eq. 4 and Eq. 5, and the Facebook trace from Jan 2007 to July 2007 is used. As shown in Fig. 4a, *Community* outperforms all other algorithms, *Approximation* is better than *Set-cover* and *Naïve*, and *Naïve* is the worst. Although without global information, the performance of *Set-cover* is comparable with *Community* and *Approximation* when k is small, and it is also much better than *Naïve*.

Fig. 4b shows the real information diffusion from the selected k-node set, where the information diffusion runs on the rest of the Facebook trace for all the algorithms. Due to sparse node contacts in the rest of the trace, the information cannot be spread all over the network at the end of the trace. Thus, we compare the percentage of active nodes over all nodes for each algorithm at the end of the trace, which represents the performance of information diffusion within certain time period. As shown in Fig. 4b, *Community* still performs the best, *Set-cover* outperforms both *Approximation* and *Naïve*, and *Approximation* is the worst. When $k = 100$, *Community* can diffusion the information to more than 50% of nodes, whereas *Approximation* can diffuse to less than 30%.

At mentioned above, all algorithms cannot spread information to the entire network at the end of the trace. Thus, we compare the time for diffusing information to 10%, 20%, 30% and 40% of nodes with varying k from 1 to 100 as shown in Fig. 5. As *Approximation* cannot spread the information to 30%, 40% of nodes at the end of the trace, it is not included in Fig. 5c and Fig. 5d. For the same reason, *Naïve* is not included in Fig. 5d. Similarly, some k values are not shown in some figures (e.g., 40 in Fig. 5d) because at the end of the trace the specified percentage of nodes cannot be informed with selected S ($|S| = k$).

As shown in Fig. 5a and Fig. 5b, *Community*, *Set-cover* and *Naïve* can diffuse information to 10%, 20% of nodes with much smaller k and shorter diffusion time for the same k values than *Approximation*, and *Community* and *Set-cover* outperform *Naïve*. For example, as shown in Fig. 5b, *Community* can spread the information to 20% of nodes with k less than 20,



(a) 10% nodes (b) 20% nodes (c) 30% nodes (d) 40% nodes

Fig. 5: Diffusion time of ***Community***, ***Approximation***, ***Naïve*** and ***Set-cover*** to 10%, 20%, 30% and 40% of nodes.

Set-cover can achieve it with k less than 30, *Naïve* can achieve it with k less than 40, and *Approximation* can achieve it with k around 60. For $k = 60$, the diffusion time for *Community*, *Set-cover*, *Naïve* and *Approximation* is 7, 10, 13 and 17 months, respectively. As shown in Fig. 5c, *Community* and *Set-cover* can spread information to 30% of nodes with shorter diffusion time and smaller k than *Naïve*. Moreover, *Community* is always the best as shown in Fig. 5. Although *Set-cover* is worse than *Approximation* in terms of expected diffusion time, it can spread information to 10%, 20%, 30% and 40% of nodes with smaller k and shorter diffusion time than *Approximation* and *Naïve*.

In summary, *Community* has the best performance on the Facebook trace, and *Set-cover* outperforms *Approximation* and *Naïve* in terms of diffusion time.

VII. Conclusions

In this paper, we addressed the problem of identifying a small number of nodes through which the information can be diffused to the network as soon as possible. We proposed two algorithms: community based algorithm and distributed set-cover algorithm, to solve the diffusion minimization problem in mobile social networks from different aspects. More specifically, community-based algorithm leverages the community structure, while distributed set-cover algorithm collects information by probing messages in a distributed way. Simulation results show that the community based algorithm has the best performance for both synthetic networks and the Facebook trace. Despite the lack of global information, the distributed set-cover algorithm outperforms the approximation algorithm in the Facebook trace in terms of diffusion time.

References

[1] H. Ma, H. Yang, M. R. Lyu, and I. King, "Mining social networks using heat diffusion processes for marketing candidates selection," in *Proc. of ACM CIKM*, 2008.

[2] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *Proc. of ACM SIGKDD*, 2010.

[3] T. Ning, Z. Yang, H. Wu, and Z. Han, "Self-interest-drive incentives for ad dissemination in autonomous mobile social networks," in *Proc. of IEEE INFOCOM*, 2013.

[4] W. Peng, F. Li, X. Zou, and J. Wu, "A privacy-preserving social-aware incentive system for word-of-mouth advertisement dissemination on smart mobile devices," in *Proc. of IEEE SECON*, 2012.

[5] S. Vishwanathan, "An o(log*n) approximation algorithm for the asymmetric p-center problem," in *Proc. of ACM SODA*, 1996.

[6] D. Liben-Nowell and J. Kleinberg, "Tracing information flow on a global scale using internet chain-letter data," *Proceedings of the National Academy of Sciences*, vol. 105, no. 12, pp. 4633–4638, 2008.

[7] E. Bakshy, I. Rosenn, C. Marlow, and L. Adamic, "The role of social networks in information diffusion," in *Proc. of ACM WWW*, 2012.

[8] D. M. Romero, B. Meeder, and J. Kleinberg, "Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on twitter," in *Proc. of ACM WWW*, 2011.

[9] P. Domingos and M. Richardson, "Mining the network value of customers," in *Proc. of ACM SIGKDD*, 2001.

[10] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proc. of ACM SIGKDD*, 2003.

[11] Y. Wang, G. Cong, G. Song, and K. Xie, "Community-based greedy algorithm for mining top-k influential nodes in mobile social networks," in *Proc. of ACM SIGKDD*, 2010.

[12] Q. Jiang, G. Song, G. Cong, Y. Wang, W. Si, and K. Xie, "Simulated annealing based influence maximization in social networks," in *Proc. of AAAI*, 2011.

[13] W. Dong, V. Dave, L. Qiu, and Y. Zhang, "Secure friend discovery in mobile social networks," in *Proc. of IEEE INFOCOM*, 2011.

[14] M. Li, N. Cao, S. Yu, and W. Lou, "Findu: Privacy-preserving personal profile matching in mobile social networks," in *Proc. of IEEE INFOCOM*, 2011.

[15] B. Han and A. Srinivasan, "Your friends have more friends than you do: identifying influential mobile users through random walks," in *Proc. of ACM MobiHoc*, 2012.

[16] S. Gaonkar, J. Li, R. R. Choudhury, L. Cox, and A. Schmidt, "Micro-blog: sharing and querying content through mobile phones and social participation," in *Proc. of ACM MobiSys*, 2008.

[17] K. K. Rachuri, C. Mascolo, M. Musolesi, and P. J. Rentfrow, "Sociablesense: exploring the trade-offs of adaptive sampling and computation offloading for social sensing," in *Proc. of ACM MobiCom*, 2011.

[18] W. Hu, G. Cao, S. V. Krishanamurthy, and P. Mohapatra, "Mobility-assisted energy-aware user contact detection in mobile social networks," in *Proc. of IEEE ICDCS*, 2013.

[19] D. López-Pintado, "Diffusion in complex social networks," *Games and Economic Behavior*, vol. 62, no. 2, pp. 573–590, 2008.

[20] J. Chuzhoy, S. Guha, E. Halperin, S. Khanna, G. Kortsarz, R. Krauthgamer, and J. S. Naor, "Asymmetric k-center is log* n-hard to approximate," *Journal of the ACM*, vol. 52, no. 4, pp. 538–551, 2005.

[21] M. E. Newman, "A measure of betweenness centrality based on random walks," *Social networks*, vol. 27, no. 1, pp. 39–54, 2005.

[22] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[23] Z. Lu, Y. Wen, and G. Cao, "Community detection in weighted networks: Algorithms and applications," in *Proc. of IEEE PerCom*, 2013.

[24] X. Zhang and G. Cao, "Transient community detection and its application to data forwarding in delay tolerant networks," in *Proc. of IEEE ICNP*, 2013.

[25] A. Lancichinetti and S. Fortunato, "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities," *Physical Review E*, vol. 80, no. 1, p. 016118, 2009.

[26] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, "On the evolution of user interaction in facebook," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Social Networks*, 2009.