Towards Information Diffusion in Mobile Social Networks

Zongqing Lu, *Member, IEEE*, Yonggang Wen, *Member, IEEE*, Weizhan Zhang, *Member, IEEE*, Qinghua Zheng, *Member, IEEE*, and Guohong Cao, *Fellow, IEEE*

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 compared to existing algorithms, and the distributed set-cover algorithm outperforms the approximation algorithm in the real trace in terms of diffusion time.

Index Terms—Information diffusion, mobile social networks, community structure

1 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". For example, a company would like to quickly raise the awareness of a new product in a network. The company initially gives free samples of the product to a small number of individuals in the network (the product is expensive or the company has limited budge such that they can only choose a small number of people). The company hopes that the initially selected users will spread the information of the new product to their friends, and their friends will propagate the information to their friends' friends and so on.

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

- *Z. Lu and G. Cao are with the Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802. E-mail: {zongqing, gcao}@cse.psu.edu.*
- *Y. G. Wen is with the School of Computer Engineering, Nanyang Technological University, 639798, Singapore. E-mail: ygwen@ntu.edu.sg.*
- *W. Zhang and Q. Zheng are with the Department of Computer Science and Engineering, Xi'an Jiaotong University, Xi'an 710049, China. E-mail: {zhangwzh, qhzheng}@mail.xjtu.edu.cn.*

Manuscript received 19 Nov. 2014; revised 24 June 2015; accepted 25 June 2015. Date of publication 1 July 2015; date of current version 31 Mar. 2016. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TMC.2015.2451624

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, considering the properties of communities that information can be quickly spread within a community and information diffusion from one community to another is much slower. Due to lack of global information and the requirement to handle the dynamic evolving of targeted networks, we further propose a distributed set-cover algorithm, where each node collects social contact information by probing messages (e.g., using Wi-Fi/Bluetooth) 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 major contributions of this paper are summarized as follows.

- We propose the probabilistic information diffusion model and formulate the diffusion minimization problem in mobile social networks.
- We design a community based algorithm, which considers both non-overlapping and overlapping community structure, to solve the diffusion minimization problem.
- We further propose a distributed set-cover algorithm, which includes two phases: discovering the diffusion set and identifying the k-node set, to solve the problem.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 gives the problem statement. The community based algorithm is presented in Section 4, followed by the distributed set-cover algorithm in Section 5. Section 6 evaluates the performance of the proposed algorithms and Section 7 concludes the paper.

2 Related Work

2.1 Information Diffusion

With the emerging of online social media, information diffusion has been extensively studied based on emails [6], blogs [7], Flickr [8], Facebook [9], and Twitter [10]. 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 [9].

Recently, a lot of research efforts focus on whether and how individuals influence each other. Domingos and Richardson [11] were the first to study the influence maximization problem and gave a probabilistic solution. Kempe et al. [12] formally formulated the problem of identifying k-node set to maximize the influence as an optimization problem. They investigated the influence maximization under two diffusion models: independent cascade model and linear threshold model and designed a greedy algorithm with approximation ratio of $(1-\frac{1}{e})$. After Kempe et al. established the influence maximization problem, it has attracted a lot of attentions. Leskovec et al. [13] proposed an optimized greedy algorithm, Chen et al. [14] proposed two faster greedy algorithms, and Jiang et al. [15] proposed a simulated annealing algorithm. Time-constrained influence maximization problem were investigated in [16], [17], both of which proposed a greedy algorithm to achieve the approximation ratio $(1-\frac{1}{e})$, and positive influence maximization was investigated in [18].

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.

2.2 Mobile Social Networks

Through mobile social networks, individuals with similar interests interact, communicate and connect with others by 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 [19], [20], [21]. Moreover, many mobile social applications have been developed such as CenceMe [22], Micro-blog [23], SociableSense [24], METIS [25], 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. McNamara et al. [26] investigated the content sharing among co-located mobile users in urban transportation and proposed a user-centric prediction scheme that collected the historical co-location information to determine the best content sources. Han et al. [19] designed a distributed random walk protocol for immunization of infectious diseases and information dissemination. Hu et al. [27] proposed an energy-aware user-contact detection algorithm through Bluetooth and accelerometer 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 the collaboration among selfish nodes for data dissemination. Lu et al. [28] proposed skeleton as the network structure of mobile social network based on best friendships and exploited it for data dissemination and worm containment. However, none of them considers the diffusion minimization problem.

This paper substantially extends the preliminary version of our result appeared in [29]. In [29], we mainly focused on how to efficiently solve the diffusion minimization based on non-overlapping community structure. In this paper, we design a more general algorithm which considers both non-overlapping and overlapping community structure and we perform additional extensive simulations in synthetic networks with overlapping community structure. Moreover, we redesign the distributed set-cover algorithm to avoid the deviousness of traveling paths of probing messages and thus enrich the up-to-date information collected by each node.

3 Preliminaries, Problem Statement and Naïve Algorithm

3.1 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 $d_u = \sum_{v \in N_u} w_{uv}$.

3.2 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.

The evolutionary game theory based diffusion model is explored, in [30], [31], to consider the influence of users decisions, actions and socio-economic connections on information diffusion. However, this diffusion model requires users' payoff matrix on whether to forward the diffused information. Since such payoff information is not always available in mobile social networks, this diffusion model cannot be adopted in this work.

Different from the *linear threshold model* [12] and the *independent cascade model* [32] that describe how individuals influence each other in social networks, the *probabilistic diffusion model* describes how the information diffuses in social networks.

3.3 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

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

where $\lambda_{uv} = \frac{w_{uv}}{d_u}$ and $\frac{1}{w_{uv}}$ 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 [33]. 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.

3.4 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 [34]. 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.

4 Community Based Algorithm

Considering the design of information diffusion in mobile social networks, intuitively, the concept of social relations

should be exploited. In this section, we design the community based heuristic algorithm.

Community represents a set of nodes in a network, where nodes inside the community have more internal connections than external connections [35], [36], [37]. 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 networks, communities have 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. In addition, for two communities, for example C_i and C_j, they may overlap with each other, i.e., $C_i \cap C_j \neq \emptyset$.

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, after that give the design of the community based algorithm.

4.1 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 $\binom{l}{2}\binom{l-1}{2}\cdots\binom{k+1}{2}$ 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 $V \backslash C_i$. Moreover, since C_i and C_j may be overlapped, the set of overlapped nodes $C_i \cap C_j$ should be excluded from C_j when considering the sum of edge weights between C_i and C_j. Note that overlapped communities tend to be closely connected communities. The set of closely connected communities of C_i ($C_i \in \mathcal{C}$) is denoted by $\mathcal{C}_{C_i}$ and represented as

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

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. Note that after $l-k$ merge steps, the communities may still overlap.

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 $O(l^2(l-k))$.

4.2 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 merging 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,

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

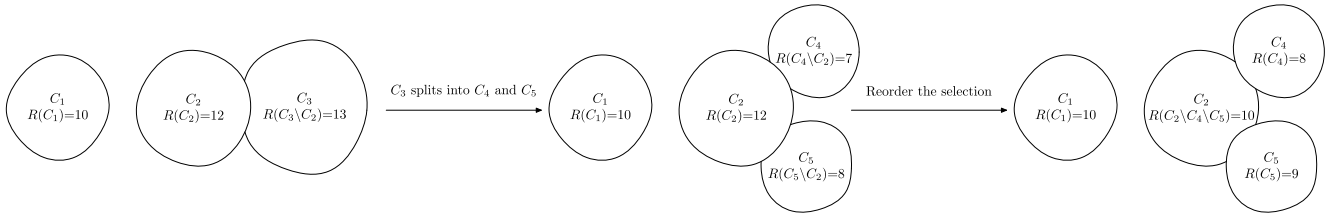


Fig. 1. Illustration of the selection of central nodes for overlapping community structure.

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

4.3 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. Furthermore, since there are two types of community structure: non-overlapping and overlapping (i.e., $\exists C_i, C_j : C_i \cap C_j \neq \emptyset$), the designed algorithm should consider both of them. The community based algorithm works as follows.

First we use similar technique as in Section 4.1 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, for non-overlapping community structure, we can easily choose the central nodes for individual communities as the candidates of diffusion nodes and thus we still have $k - |\mathcal{C}|$ candidates remaining. However, for overlapping community structure, the selection of central nodes is more complex. Since $\exists C_i, C_j : C_i \cap C_j \neq \emptyset$, the overlapped nodes between C_i and C_j are already covered by N_{C_i} if N_{C_i} is selected before N_{C_j}. Thus, the selection of central nodes in C_j should not consider the set of overlapped nodes. The selected central node should be $N_{C_j \backslash C_i}$ rather than N_{C_j}, since $R(C_j) \geq R(C_j \backslash C_i)$. Therefore, a proper order of selecting central nodes for overlapping community structure is needed to obtain a better expected diffusion time. Moreover, the overlapped nodes covered by the previously selected central nodes should not be considered by subsequent selection of central nodes. To minimize the expected diffusion time, the selection of central nodes of overlapping communities works as follows. First, we select the community with minimum diffusion radius from $\mathcal{C}$, denoted as C_1, and choose N_{C_1} as a candidate. Then, we choose the community that minimizes $R(C_i \backslash C_1), C_i \in \mathcal{C} \backslash \{C_1\}$, denoted as C_2, and $N_{C_2 \backslash C_1}$ will be selected. Repeatedly, for ith selection we have $N_{C_i \backslash C_1 \backslash \ldots \backslash C_{i-1}}$. We denote $C_i \backslash C_1 \backslash \ldots \backslash C_{i-1}$ as C_i^*, which represents the node set C_i excluding the nodes covered by $i - 1$ already selected central nodes.

Next, we identify other candidate within community C_i with the maximum expected diffusion time in the current $\mathcal{C}$. If C_i is an originally detected community, we first choose a candidate, according to the approach described in Section 4.2, to replace N_{C_i} ($N_{C_i^*}$ for overlapping community structure), then identify one more candidate and add it into S. If C_i is a merged community, it is split into two original communities C_j and C_k, and then we have the updated $\mathcal{C}$. For non-overlapping community structure, 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 this replacement will decrease the expected diffusion time of the network. For overlapping community structure, directly choosing $N_{C_j^*}$ and $N_{C_k^*}$ might not be a good choice to obtain the minimum expected diffusion time due to the change of the selection order of central nodes. That is, for example as shown in Fig. 1, $\mathcal{C} = \{C_1, C_2, C_3\}$, where $R(C_1) = 10$, $R(C_2) = 12$, $R(C_3) = 14$ and $R(C_3 \backslash C_2) = 13$. Since C_3 is a combined community of C_4 and C_5, in order to identify one more candidate, we split it into C_4 and C_5, where $R(C_4 \backslash C_2) = 7$ and $R(C_5 \backslash C_2) = 8$. So, if we keep current selection order, i.e., N_{C_1}, N_{C_2}, $N_{C_4 \backslash C_2}$ and $N_{C_5 \backslash C_2}$, the expected diffusion time of the network will be $R(C_2) = 12$. However, if we re-select all the candidates based on the updated $\mathcal{C}$, we will have $R(C_4)$, $R(C_5)$, $R(C_1)$, $R(C_2 \backslash C_4 \backslash C_5)$. A better expected diffusion time is obtained as $R(C_2 \backslash C_4 \backslash C_5)$, since $R(C_2 \backslash C_4 \backslash C_5) \leq R(C_2)$. Thus, we should re-select all the candidates for the updated $\mathcal{C}$ if the split of a community happens for overlapping community structure.

The aforementioned process is executed iteratively for both non-overlapping and overlapping community structure until no candidate remains and we call it *restoring process*.

Fig. 2 gives an example of the community based algorithm for non-overlapping community structure, where $l = 7$ and $k = 7$. Fig. 2a shows the detected communities. Figs. 2a to 2c show the merging process. Figs. 2c to 2f show the restoring process. After the merging process, three communities remain: C_1'', C_2'' and C_4 as in Fig. 2c. After choosing the central node from each of these communities, as $R(C_1'') = 20$, $R(C_2'') = 24$ and $R(C_4) = 35$, we need to identify another diffusion node from C_4. After selecting two diffusion nodes from C_4, $\tau'(S_{C_4}, C_4) = 19$ as shown in Fig. 2d. Since $R(C_2'') > \tau'(S_{C_4}, C_4)$, we then switch to C_2''. As C_2'' is a merged community, we split C_2'' into C_5 and C_2', and replace $N_{C_2''}$ with N_{C_5} and $N_{C_2'}$ as shown in Fig. 2e. 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} ($|S_{C_4}| = 3$) as diffusion nodes, and the expected diffusion time of the network is the maximum of

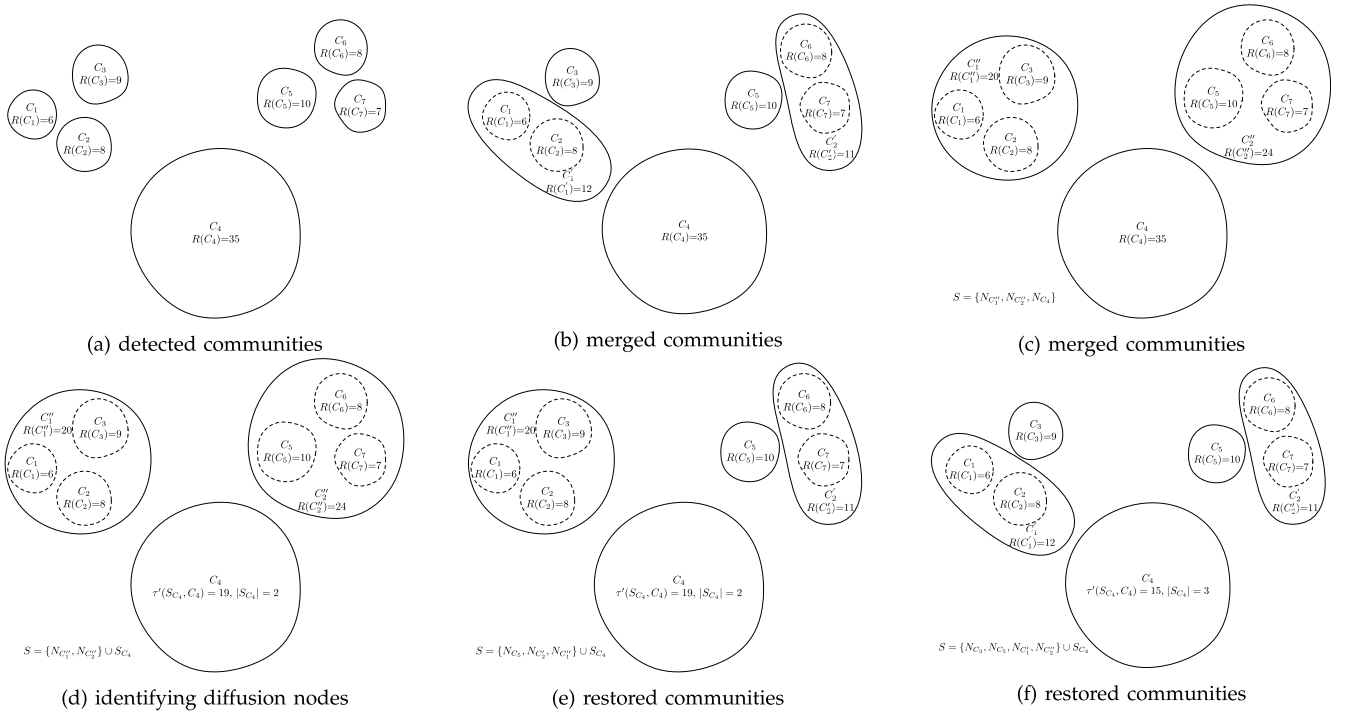


Fig. 2. Illustration of the community based algorithm for non-overlapping community structure, where $l = 7$ and $k = 7$. (a) to (c) show the merging process. (c) to (f) show the restoring process (identifying diffusion nodes). Finally, $S = \{N_{C_3}, N_{C_5}, N_{C'_1}, N_{C'_2}\} \cup S_{C_4}$.

individual communities; that is, $\tau'(S_{C_4}, C_4) = 15$ as shown in Fig. 2f. 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 $N_{C_1}, N_{C_2}, N_{C_3}, N_{C_4}, N_{C_5}, N_{C_6}$ and N_{C_7}, and the expected diffusion time of the network is also determined by C_4, i.e., $R(C_4) = 35$. Thus, our algorithm performs better than the straightforward solution; i.e., the expected diffusion time of our algorithm is 15 which is much better than 35. 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 4.2; thus, our algorithm has better performance.

Regarding the time complexity, 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 4.2 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 ($O(n^5)$).

4.4 Performance Analysis

Since the community based algorithm relies heavily on the community structure, which is a natural property of networks, it is hard to give a mathematically rigorous performance analysis. In the following, we provide insights into the performance of the algorithm based on the diffusion node selection process.

As mobile social networks usually consist of a large number of communities and k is usually small, we consider the case that there is only one diffusion node identified from a community. As described in Section 4.3, after the merging process, the number of communities is no more than k (i.e., $k \geq |\mathcal{C}|$), and the merging of any two communities will produce a community with larger diffusion radius than the maximum one in $\mathcal{C}$ (i.e., $\forall C_i, C_j : R(C_i \cup C_j) > \max\{R(C) : C \in \mathcal{C}\}$). Thus, according to the criteria of community merge, the communities have similar diffusion radius when the merging process stops. Suppose that S^* is the diffusion node set of the optimal solution and τ'_* is the optimal expected diffusion time. For a node $u \in S^*$, let $V_u = \{v \in V : |(u, v)| \leq \tau'_*\}$. If the node set V_u is treated as a community, the communities $\{V_u : u \in S^*\}$ are generally prone to have similar or even same diffusion radius. Based on these facts, we assume that the community based algorithm performs equally with the optimal solution at this phase, although there is a slight deviation between them. In the following, we present the performance analysis based on this assumption.

We use an example to illustrate the comparison between the optimal solution and the community based algorithm. Assume that a large number of nodes form as a straight line (the length is L) and the distance (the expected diffusion time) between neighboring nodes is identical. When $k = 1$, both of these two approaches will choose the node in the middle as the diffusion node. When $k = 2$, the optimal solution will choose the nodes at $\frac{L}{4}$ and $\frac{3L}{4}$, and the optimal expected diffusion time τ'_* is $\frac{L}{4}$; the community based algorithm will divide the nodes into two communities and find one diffusion node from each community, and thus they still perform equally, i.e., $\tau' = \frac{L}{4}$. When $k = 3$, the optimal solution will select the nodes at $\frac{L}{6}, \frac{L}{2}$ and $\frac{5L}{6}$, and $\tau'_* = \frac{L}{6}$; the

community based algorithm will divide one of two communities into two, but the expected diffusion time of the network is still determined by the remaining community, i.e., $\tau' = \frac{L}{4}$ and $\tau'/\tau'_* = \frac{3}{2}$. By deduction, we have the best case of performance that $\tau'/\tau'_* = 1$, when $k = 2^n$, $n = 0, 1, 2, \ldots$, and the worst case of performance that $\tau'/\tau'_* = \frac{2^n-1}{2^{n-1}} \approx 2$, when $k = 2^n - 1$, $n = 2, 3, 4, \ldots$.

Although such comparison is derived from the assumption that can be hardly approved for each case since community structure varies in different networks, the assumption is generally valid and thus the analysis gives insights into the performance of the community based algorithm, i.e., the expected diffusion time of the community based algorithm is at most two times of the optimal solution.

5 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, such as mobile social networks constructed from opportunistic node contacts. Furthermore, 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, in this section, we propose a distributed set-cover algorithm to address these problems, where each node collects up-to-date information and the collected information is exploited to solve the diffusion minimization problem.

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 network nodes V. Although it is impossible to have the minimized diffusion time beforehand, 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.

5.1 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 which includes the set of traversed nodes and time-to-live (TTL), where the set of traversed nodes initially includes the node *id* of the message generator and TTL is set to γ, and stores it in the local message queue. When a node u contacts with node v, u will randomly choose one probing message whose set of traversed nodes dose not include 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 deduct TTL by t_{vu}. After that, if TTL ≥ 0, v will merge the set of traversed nodes in the probing message into $\Gamma(v)$, where $\Gamma(v)$ denotes the set of nodes which have been traversed by the probing messages received at node v. Finally, if TTL ≤ 0, the probing message is discarded, otherwise, node v is added into the set of traversed nodes and the message 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 diffusion set* of node u. Note that, during this discovering process, 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. The formal description of the discovering of diffusion sets is shown in Protocol 1.

Protocol 1. Discovering the diffusion set

Inputs: Δt *and* γ
Event: *Every* Δt.
Object: *All nodes*:
I. Generate a probing message including TTL and a set of traversed nodes, where TTL is set to γ and the set of traversed nodes is initialized to include the message generator.
II. Add the message into local message queue.
Event: *When two nodes contact with each other*.
Object: *Each of them*
Outgoing:
I. Randomly select a probing message, whose set of traversed nodes does not include the receiver, and send it out with the probability of information diffusion from sender to receiver.
Incoming:
I. When received a probing message, deduct the expected diffusion time from receiver to sender from TTL of the message.
II. If TTL ≥ 0, add the set of traversed nodes into the *up-to-date* diffusion set.
III. If TTL > 0, include itself into the set of traversed nodes and store the message into its message queue, otherwise, discard the message.

5.2 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 1 and the worst case of time complexity is $O(kn^2)$.

Algorithm 1. Identifying the k-node Set

Input: V, k
Output: S
1 $V^{'} = V$
2 **while** $|S| < k \,\&\&\, V^{'} \neq \emptyset$ **do**
3 $u = \arg\max_{v \in V} |V^{'} \cap \Gamma(v)|$
4 $V^{'} = V^{'} \backslash \Gamma(v) \backslash \{u\}$
5 $S = S \cup \{u\}$
6 **end**

5.3 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., $t_{vu} > \gamma$).

The size of the probing message increases at each transmission, since one more node *id* is added into the set of traversed nodes each time. However, this only incurs very little additional transmission overhead, because the traveling region of probing messages is restricted by γ and thus the number of nodes that can be traversed by probing messages is limited. The set of traversed nodes of probing messages is designed to avoid the deviousness of traveling paths of probing messages (i.e., a probing message only can visit each traversed node once) and to potentially increase the traversed nodes within γ.

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.

6 Performance Evaluations

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

6.1 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 [36], which can detect both non-overlapping and overlapping community structure. 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 of the algorithms in different network settings, we use the synthetic networks generated by the well-known benchmark proposed in [38]. It provides power-law distribution of node degree and edge weight, and various topology control, and it allows overlaps between communities. 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; the number of communities of overlapping nodes o_m; the number of overlapping nodes o_n; the overlapping fraction θ (o_n/n). The settings of these parameters are close to [39], which are shown in Table 1. Compared to [39] that employed the same benchmark, same values are assigned to most parameters such as o_m, μ_t and μ_w, while more variations are allowed for other parameters than [39] to make it more general. For example, the overlapping fraction can be changed from 0 to 0.5 in the networks with 500, 1,000 and 2,000 nodes, respectively.

TABLE 1
Parameter Settings for Benchmark

Parameter	Value	Meaning	Parameter	Value	Meaning
n	500, 1,000, 2,000	number of nodes	μ_w	0.1, 0.3	mixing parameter for edge weights
μ_t	0.1, 0.3	mixing parameter for topology	β	2	exponent for weight distribution
α_{max}	20	maximum neighbors	ξ_1	2	minus exponent for degree sequence
α	15	average neighbors	ξ_2	1	minus exponent for community size distribution
θ	0 to 0.5	overlapping fraction	o_m	2	number of communities of overlapping nodes

The expected diffusion time is formulated based on 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.

6.1.1 Non-Overlapping Community Structure

Fig. 3 shows the expected diffusion time of *Community*, *Approximation* and *Naïve* with varying k for different network settings with non-overlapping community structure (i.e., $\theta = 0$). Since in real applications k should be a small value, we choose k no more than 5 percent of the network size for each setting and k varies from 1 to 5 percent of network size. As shown in Fig. 3, for the networks with $\mu_t = 0.1$ 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 the networks with more 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 percent, 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. Moreover, *Approximation* performs worse than other two algorithms when k is small and such performance deteriorates in the networks with more heterogeneity in topology.

6.1.2 Overlapping Community Structure

Fig. 4 shows the performance of these algorithms on different network settings with overlapping community structure, where we fix k to 5 percent of network sizes and vary θ for 0 to 0.5. As shown in Fig. 4, for the networks with $\mu_t = 0.1$ and $\mu_w = 0.1$, *Community* is better than *Approximation* and *Naïve*, especially for the network with large size. Note that due to the randomness of the benchmark and the change of θ, the optimal expected diffusion time of generated networks may also vary. As shown in the figure, the performance of *Community* is relatively stable for different network sizes with $\mu_t = 0.1$ and $\mu_w = 0.1$ when θ varies. *Approximation* is also stable in terms of θ, but the

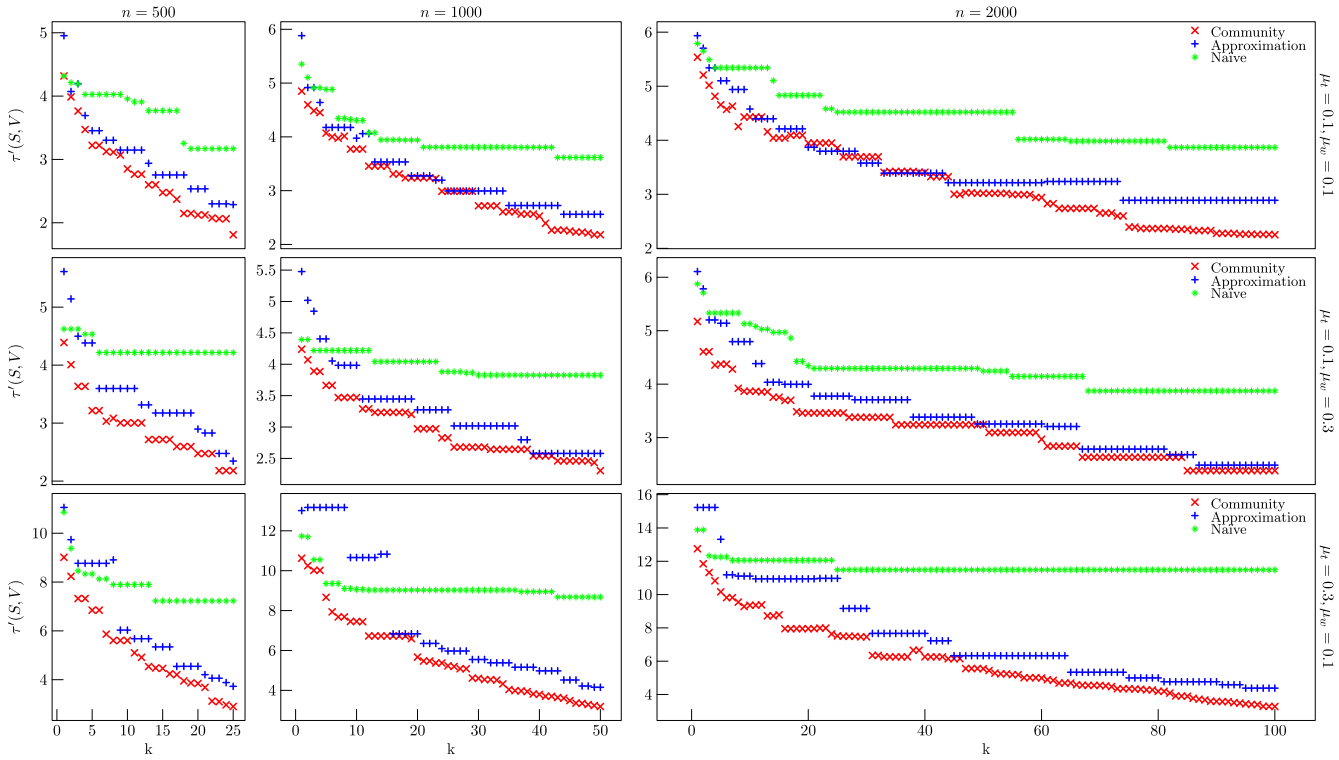


Fig. 3. Expected diffusion time of *Community*, *Approximation*, and *Naïve* in different network settings with varying k and fixed $\theta = 0$.

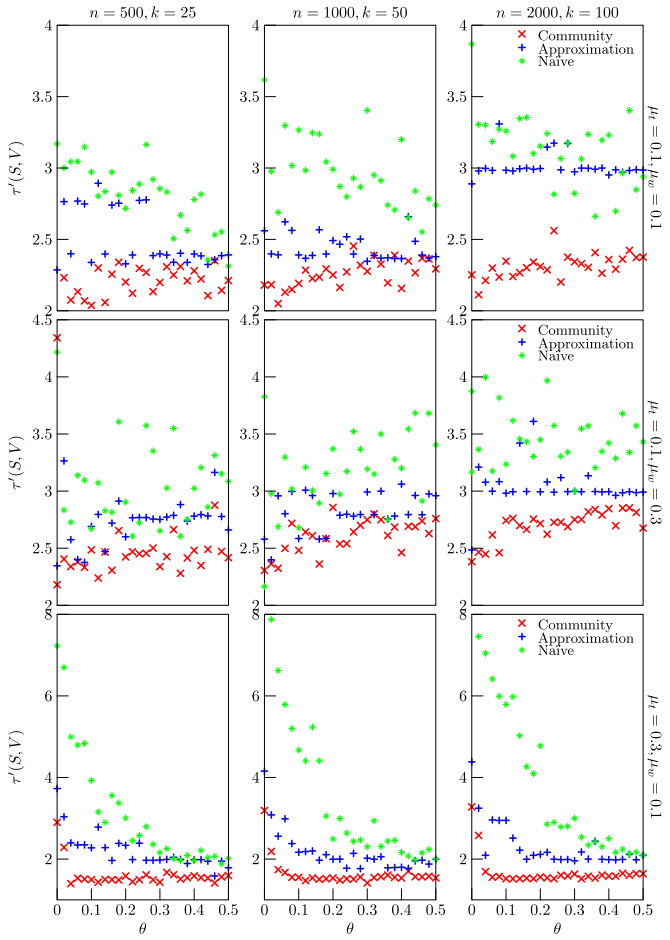


Fig. 4. Expected diffusion time of *Community*, *Approximation*, and *Naïve* in different network settings with varying θ and fixed k.

performance significantly decreases when the network size goes large. Meanwhile, the performance of *Naïve* appears more random in terms of overlapping fraction θ than other two algorithms. For the networks with more heterogeneity in edge weight (i.e., $\mu_t = 0.1$ and $\mu_w = 0.3$) *Community* still outperforms other two algorithms. The performance of *Community* and *Approximation* is more stable with varying θ and much better than *Naïve*, which still has the randomness of performance. For the networks with more heterogeneity in topology (i.e., $\mu_t = 0.3, \mu_w = 0.1$), the expected diffusion time of all three algorithms drops when θ increases from 0, and then becomes stabilized when θ further increases. As shown in Fig. 4, although the performance of *Approximation* and *Naïve* is improved when θ increases, the performance of *Community* is much better and more stable. The steady performance of *Community* in all network settings in Fig. 4 proves that *Community* works well for the overlapping community structure.

The computation time of these algorithms is related to the size of the network and k-node set and thus varies largely in the experiments. However, it exhibits the same as we analyzed for computational complexity in Section 4; i.e., *Community* has better computational complexity than *Approximation*. For example, in the experiments, when the network size is 1,000 and $k = 20$, the computation time is 12, 16 and 19 s for *Naïve*, *Community* and *Approximation*, respectively; when the network size is 2,000 and $k = 50$, the computation time is 70, 93 and 142 s for *Naïve*, *Community* and *Approximation*, respectively.

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*. Moreover, *Community* works well for both non-overlapping and overlapping community structure.

TABLE 2
Facebook Trace Summary

Trace	Facebook
No. of nodes	2,320
No. of edges	9,150
Average neighbors	7.9
No. of contacts	168,542
Duration (days)	752

6.2 Estimations of γ and Δt

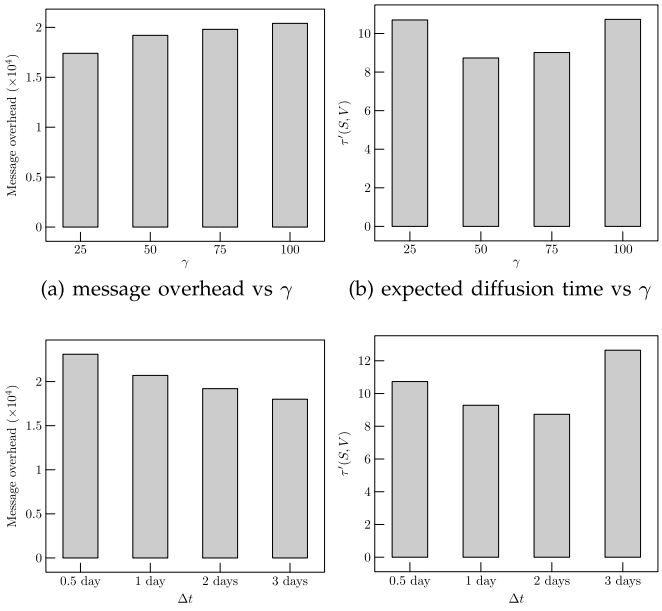
In this section, we estimate γ and Δt for the distributed set-cover algorithm (*Set-cover*) based on the Facebook trace [40], which contains friendship information and wall posts (with timestamp) among Facebook users in the New Orleans regional network for more than four years. We choose a partial trace which spans from January 2007 to January 2009 and contains 2,320 nodes. The chosen trace is summarized in Table 2, where we formulate the contact between 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 three-month trace from January 2007 to March 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 three-month trace from April 2007 to June 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

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

where $\lceil \log_{|N_{avg}|} \frac{n}{k} \rceil$ is the minimum hop to reach $\frac{n}{k}$ nodes from the diffusion node. Figs. 5a and 5b show the message overhead (the number of message transfers) and the expected diffusion 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



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

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

Fig. 5. Performance of *Set-cover* with varying γ and Δt in terms of message overhead and expected diffusion time, where $k = 50$, $\Delta t = 2$ days for (a) and (b), and $k = 50$, $\gamma = 50$ for (c) and (d).

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.

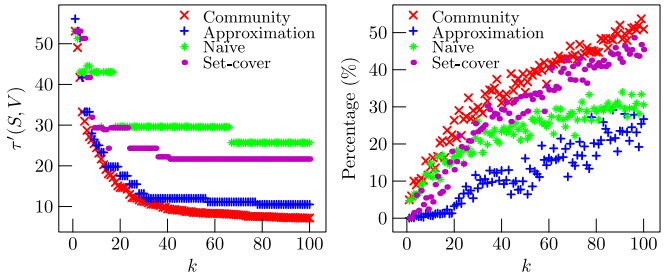
For Δt, we use the average node degree d_a and the time unit of contact frequency T to estimate Δt. We set Δt equal to the average time interval between contacts. That is

$$\Delta t = \left\lceil \frac{T}{d_a} \right\rceil. \tag{5}$$

Figs. 5c and 5d 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 two days by Eq. (5). Fig. 5c 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. 5d. Thus, the estimation of Δt achieves a good tradeoff between performance and cost.

6.3 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 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.

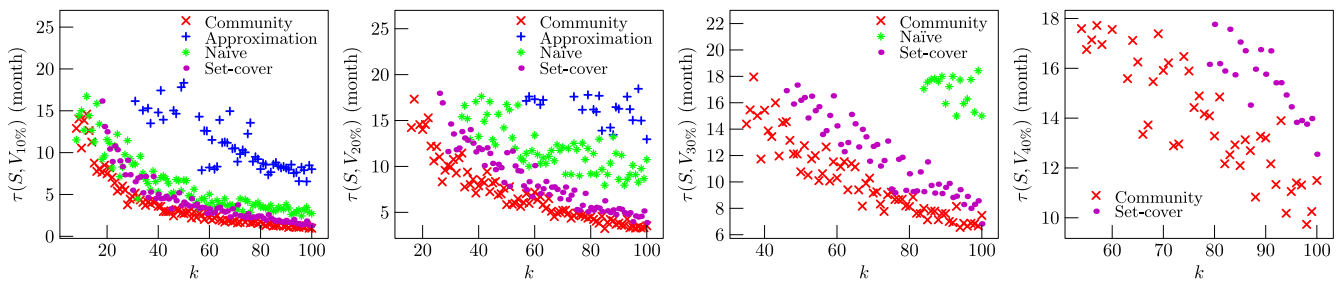


(a) expected diffusion time (b) percentage of active nodes

Fig. 6. Performance on the Facebook trace of *Community*, *Approximation*, *Naïve*, and *Set-cover*.

Fig. 6a shows the expected diffusion time, where γ and Δt are chosen according to Eq. (4) and Eq. (5), and the Facebook trace from January 2007 to June 2007 is also used for other three centralized algorithms to construct the network. Specifically, we construct a network based on node contacts recorded in the trace, identify diffusion nodes according to each algorithm, and then calculate the expected diffusion time for each algorithm. As shown in Fig. 6a, *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. 6b expected diffusion time for each algorithm k-node set, where the information diffusion runs on the rest of the Facebook trace for all the algorithms. Specifically, at the beginning of the rest of the Facebook trace, the selected k-node set is set as active nodes, and then the information is spread from the k-node set to other nodes in the network according to the probabilistic diffusion model, i.e., upon a node contact, the information is spread from an active node to an inactive with a probability. 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



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

Fig. 7. Diffusion time of *Community*, *Approximation*, *Naïve*, and *Set-cover* to 10, 20, 30 and 40 percent of nodes.

trace, which represents the performance of information diffusion within certain time period. As shown in Fig. 6b, *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 percent of nodes, whereas *Approximation* can diffuse to less than 30 percent.

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 percent of nodes with varying k from 1 to 100 as shown in Fig. 7. As *Approximation* cannot spread the information to 30, 40 percent of nodes at the end of the trace, it is not included in Fig. 7c and Fig. 7d. For the same reason, *Naïve* is not included in Fig. 7d. Similarly, some k values are not shown in some figures (e.g., 40 in Fig. 7d) because at the end of the trace the specified percentage of nodes cannot be informed with selected S ($|S| = k$).

As shown in Figs. 7a and 7b, *Community*, *Set-cover* and *Naïve* can diffuse information to 10, 20 percent 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. 7b, *Community* can spread the information to 20 percent of nodes with k less than 20, *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. 7c, *Community* and *Set-cover* can spread information to 30 percent of nodes with shorter diffusion time and smaller k than *Naïve*. Moreover, *Community* is always the best as shown in Fig. 7. Although *Set-cover* is worse than *Approximation* in terms of expected diffusion time, it can spread information to 10, 20, 30 and 40 percent 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.

7 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: the community based algorithm and the distributed set-cover algorithm, to solve the diffusion minimization problem in mobile social networks from different aspects. Specifically, the community-based algorithm, for the social point of view, leverages the community structure to select diffusion nodes, while the distributed set-cover algorithm identifies diffusion nodes based on the information collected 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.

Acknowledgments

This work was supported in part by the US National Science Foundation (NSF) under grant number CNS-1421578, by Network Science CTA under grant W911NF-09-2-0053, and by DTRA under grant HDTRA1-10-1-0085.

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. 17th ACM Conf. Inf. Knowl. Manage.*, 2008, pp. 233–242.

[2] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 1029–1038.

[3] T. Ning, Z. Yang, H. Wu, and Z. Han, "Self-interest-drive incentives for ad dissemination in autonomous mobile social networks," in *Proc. IEEE INFOCOM*, 2013, pp. 2310–2318.

[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. 9th Annu. IEEE Commun. Soc. Conf. Sensor, Mesh Ad Hoc Commun. Netw.*, 2012, pp. 596–604.

[5] S. Vishwanathan, "An o(log*n) approximation algorithm for the asymmetric P-center problem," in *Proc. 7th Annu. ACM-SIAM Symp. Discrete Algorithms*, 1996, pp. 1–5.

[6] D. Liben-Nowell and J. Kleinberg, "Tracing information flow on a global scale using internet chain-letter data," *Proc. Nat. Acad. Sci.*, vol. 105, no. 12, pp. 4633–4638, 2008.

[7] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins, "Information diffusion through blogspace," in *Proc. 13th Int. Conf. World Wide Web*, 2004, pp. 491–501.

[8] M. Cha, A. Mislove, and K. P. Gummadi, "A Measurement-driven analysis of information propagation in the flickr social network," in *Proc. 18th Int. Conf. World Wide Web*, 2009, pp. 721–730.

[9] E. Bakshy, I. Rosenn, C. Marlow, and L. Adamic, "The role of social networks in information diffusion," in *Proc. 21st Int. Conf. World Wide Web*, 2012, pp. 519–528.

[10] 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. 20th Int. Conf. World Wide Web*, 2011, pp. 695–704.

[11] P. Domingos and M. Richardson, "Mining the network value of customers," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2001, pp. 57–66.

[12] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2003, pp. 137–146.

[13] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2007, pp. 420–429.

[14] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 199–208.

[15] Q. Jiang, G. Song, G. Cong, Y. Wang, W. Si, and K. Xie, "Simulated annealing based influence maximization in social networks," in *Proc. 25th AAAI Conf. Artif. Intell.*, 2011, pp. 127–132.

[16] B. Liu, G. Cong, D. Xu, and Y. Zeng, "Time constrained influence maximization in social networks," in *Proc. IEEE 12th Int. Conf. Data Mining.*, 2012, pp. 439–448.

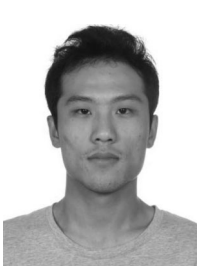
[17] W. Chen, W. Lu, and N. Zhang, "Time-critical influence maximization in social networks with Time-delayed diffusion process," in *Proc. AAAI Conf. Artif. Intell.*, 2012, pp. 592–598.

[18] H. Zhang, T. N. Dinh, and M. T. Thai, "Maximizing the spread of positive influence in online social networks," in *Proc. IEEE 33rd Int. Conf. Distrib. Comput. Syst.*, 2013, pp. 317–326.

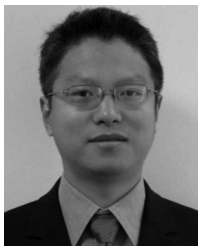
[19] B. Han, J. Li, and A. Srinivasan, "Your friends have more friends than you do: Identifying influential mobile users through Random-walk sampling," *IEEE/ACM Trans. Netw.*, vol. 22, no. 5, pp. 1389–1400, Oct. 2014.

[20] J. Wu, M. Xiao, and L. Huang, "Homing spread: Community home-based multi-copy routing in mobile social networks," in *Proc. IEEE INFOCOM*, 2013, pp. 2319–2327.

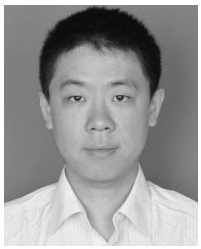
[21] W. Gao, G. Cao, T. La Porta, and J. Han, "On exploiting transient social contact patterns for data forwarding in delay-tolerant networks," *IEEE Trans. Mobile Comput.*, vol. 12, no. 1, pp. 151–165, Jan. 2013.
[22] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell, "Sensing meets mobile social networks: The design, implementation and evaluation of the cenceme application," in *Proc. 6th ACM Conf. Embedded Netw. Sens. Syst.*, 2008, pp. 337–350.
[23] 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. 6th Int. Conf. Mobile Syst., Appl. Serv.*, 2008, pp. 174–186.
[24] 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. 17th Annu. Int. Conf. Mobile Comput. Netw.*, 2011, pp. 73–84.
[25] K. K. Rachuri, C. Efstratiou, I. Leontiadis, C. Mascolo, and P. J. Rentfrow, "Metis: Exploring mobile phone sensing offloading for efficiently supporting social sensing applications," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, 2013, pp. 85–93.
[26] L. McNamara, C. Mascolo, and L. Capra, "Media sharing based on colocation prediction in urban transport," in *Proc. 14th ACM Int. Conf. Mobile Comput. Netw.*, 2008, pp. 58–69.
[27] W. Hu, G. Cao, S. V. Krishanamurthy, and P. Mohapatra, "Mobility-assisted Energy-aware user contact detection in mobile social networks," in *Proc. IEEE 33rd Int. Conf. Distrib. Comput. Syst.*, 2013, pp. 155–164.
[28] Z. Lu, X. Sun, Y. Wen, and G. Cao, "Skeleton construction in mobile social networks: Algorithm and application," in *Proc. 11th Annu. IEEE Int. Conf. Sens., Commun. Netw.*, 2014, pp. 477–485.
[29] Z. Lu, Y. Wen, and G. Cao, "Information diffusion in mobile social networks: The speed perspective," in *Proc. IEEE INFOCOM*, 2014, pp. 1932–1940.
[30] C. Jiang, Y. Chen, and K. Liu, "Evolutionary dynamics of information diffusion over social networks," *IEEE Trans. Signal Process.*, vol. 62, no. 17, pp. 4573–4586, Sep. 2014.
[31] C. Jiang, Y. Chen, and K. Liu, "Graphical evolutionary game for information diffusion over social networks," *IEEE J. Sel. Topics Signal Process.*, vol. 8, no. 4, pp. 524–536, Aug. 2014.
[32] D. López-Pintado, "Diffusion in complex social networks," *Games Econ. Behavior*, vol. 62, no. 2, pp. 573–590, 2008.
[33] 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," *J. ACM*, vol. 52, no. 4, pp. 538–551, 2005.
[34] M. E. Newman, "A measure of betweenness centrality based on random walks," *Soc. Netw.*, vol. 27, no. 1, pp. 39–54, 2005.
[35] S. Fortunato, "Community detection in graphs," *Phys. Rep.*, vol. 486, no. 3-5, pp. 75–174, 2010.
[36] Z. Lu, Y. Wen, and G. Cao, "Community detection in weighted networks: Algorithms and applications," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, 2013, pp. 179–184.
[37] X. Zhang and G. Cao, "Transient community detection and its application to data forwarding in delay tolerant networks," in *Proc. 21st IEEE Int. Conf. Netw. Protocols*, 2013, pp. 1–10.
[38] A. Lancichinetti and S. Fortunato, "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities," *Phys. Rev. E*, vol. 80, no. 1, p. 016118, 2009.
[39] S. Gregory, "Finding overlapping communities in networks by label propagation," *New J. Phys.*, vol. 12, p. 103018, 2010.
[40] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, "On the evolution of user interaction in facebook," in *Proc. 2nd ACM SIGCOMM Workshop Soc. Netw.*, 2009, pp. 37–42.



Zongqing Lu received the BE and ME degrees from Southeast University, China, and the PhD degree in computer science from Nanyang Technological University, Singapore. He is currently working as a postdoctoral fellow in the Department of Computer Science and Engineering at the Pennsylvania State University. His research interests include mobile computing, mobile cloud computing, mobile networking, mobile sensing, social networks, and opportunistic networks. He is a member of the IEEE.



Yonggang Wen received the PhD degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT) in 2008. He is currently an assistant professor with the School of Computer Engineering at Nanyang Technological University, Singapore. Previously, he has worked at Cisco as a senior software engineer and a system architect for content networking products. He has also worked as a research intern at Bell Laboratories, Sycamore Networks, and served as a technical advisor to the chairman at Linear A Networks, Inc. His research interests include cloud computing, mobile computing, multimedia network, cyber security, and green ICT. He is a member of the IEEE.



Weizhan Zhang received the BS degree in electronics engineering in 1999 from Zhejiang University, China, and the PhD degree in computer science in 2010 from Xi'an Jiaotong University, China. He served as a software engineer in Datang Telecom Corporation from 1999 to 2002. Now, he is an associate professor in the Department of Computer Science and Technology in Xi'an Jiaotong University, China. He is currently a visiting scholar at Penn State. His research interests include multimedia networking, cloud computing, and mobile computing. He is a member of the IEEE.

Qinghua Zheng received the BS degree in computer software in 1990, the MS degree in computer organization and architecture in 1993, and the PhD degree in system engineering in 1997 from Xian Jiaotong University, China. He did postdoctoral research at Harvard University from February 2002 to October 2002 and visiting professor research at HongKong University from November 2004 to January 2005. Since 1995, he has been with the Department of Computer Science and Technology at Xi'an Jiaotong University. Now, he is a professor in the Department of Computer Science and Technology at Xi'an Jiaotong University, and serves as the vice president of Xi'an Jiaotong University. His research interests include intelligent e-learning theory and algorithm, network security, and trusted software. He is a member of the IEEE.

Guohong Cao received the BS degree from Xian Jiaotong University, China, the MS and PhD degrees in computer science from the Ohio State University in 1997 and 1999, respectively. Since then, he has been with the Department of Computer Science and Engineering at the Pennsylvania State University, where he is currently a professor. His research interests are wireless networks and mobile computing. He has published more than 150 papers in the areas of wireless sensor networks, wireless network security, vehicular ad hoc networks, cache management, data access and dissemination, and distributed fault tolerant computing. He has served on the editorial boards of the *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Wireless Communications*, and has served on the organizing and technical program committees of many conferences. He received the US National Science Foundation (NSF) CAREER award in 2001. He is a fellow of the IEEE.