Algorithms and Applications for Community Detection in Weighted Networks

Zongqing Lu, Xiao Sun, Yonggang Wen, Guohong Cao, and Thomas La Porta

Abstract—Community detection is an important issue due to its wide use in designing network protocols such as data forwarding in Delay Tolerant Networks (DTN) and worm containment in Online Social Networks (OSN). However, most of the existing community detection algorithms focus on binary networks. Since most networks are naturally weighted such as DTN or OSN, in this article, we address the problems of community detection in weighted networks, exploit community for data forwarding in DTN and worm containment in OSN, and demonstrate how community can facilitate these network designs. Specifically, we propose a novel community detection algorithm, and introduce two metrics: intra-centrality and inter-centrality, to characterize nodes in communities, based on which we propose an efficient data forwarding algorithm for DTN and a worm containment strategy for OSN. Extensive trace-driven simulation results show that the proposed community detection algorithm, the data forwarding algorithm, and the worm containment strategy significantly outperform existing works.

Index Terms—Community detection, data forwarding, worm containment, delay tolerant networks, online social networks.

1 Introduction

Community is used to represent a group of nodes in a network where nodes inside the community have more internal connections than external connections [2][3]. Community has been well studied in biology, sociology, psychology, business, etc, and has been exploited for designing network protocols such as data forwarding in Delay Tolerant Networks (DTN) and worm containment in Online Social Networks (OSN) [4].

In DTN, mobile nodes contact each other opportunistically. Due to low node density and unpredictable node mobility, end-to-end connections are hard to maintain. Alternatively, node mobility is exploited to let mobile nodes physically carry data as relays, and forward data opportunistically upon contacts. Then, the key problem is how to select the appropriate relays. Since social relations among mobile users are more likely to be long-term characteristics and less volatile than node mobility, forwarding schemes based on social concepts [5][6][7][4][8][9] outperform traditional approaches [10][11], where data forwarding is facilitated with social concepts such as betweenness and social similarity [5], node centrality [7], social contact pattern [8][9], and community [6][4].

In OSN, especially when the communication is through wireless networks, mobile devices can be easily infected by malicious software such as worms or virus. Thus, many researchers design solutions to slow down and contain the worm propagation. Among them, one important problem is how to schedule who to get the software patches first when network bandwidth is a bottleneck. To address this problem, community concepts are exploited in [12][4]. In [12], the bridge nodes between communities are first patched so that they can be isolated to contain the worm propagation. In [4], the overlapped nodes between communities are identified and their neighboring nodes are patched first.

From these two examples, we can see that the community structure can be exploited for protocol design. Then, how to detect communities becomes an important issue. Community detection has attracted lots of attentions, as detailed in [3]. However, most of the existing community detection algorithms focus on binary networks, since some networks are naturally binary, such as biological networks where the edge between two nodes either exists or not. Among these algorithms, *CFinder* [13] and *RAK* [14] are the most popular and efficient ones. *CFinder* defines a k-clique community as a union of all k-cliques that can be reached from each other through a series of adjacent k-cliques, where two k-cliques are said to be adjacent if they share $k-1$ nodes. *RAK* attaches a unique label with each node and uses label propagation to detect communities.

However, most networks are weighted such as social networks, DTN or OSN. To simplify the analysis or design, these networks can be formulated as binary networks. For example, best friends are treated the same as normal friends in OSN. Multiple contacts or single contact are both counted as having contacts in DTN. Although binary network can simplify the analysis, some important information of weighted network may be lost, and hence affect the network performance. For example, with binary network in DTN, when choosing a relay, a node will not be able to differentiate nodes that it has contacted once or multiple times in the past.

- *Z. Lu, X. Sun, G. Cao and T. La Porta are with the Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802. E-mail: zongqing@cse.psu.edu, xxs118@cse.psu.edu, gcao@cse.psu.edu and tlp@cse.psu.edu.*
- *Y.G. Wen is with the School of Computer Engineering, Nanyang Technological University, 639798 Singapore. E-mail: ygwen@ntu.edu.sg.*
- *A preliminary version [1] of this paper appeared in IEEE PerCom 2013.*

Recently, there is some research on community detection in weighted networks, e.g., *COPRA* [15] and *Strength* [16]. *COPRA* is based on *RAK* [14], and hence, similar to *RAK*, it hardly converges to a constant state during label propagation and the detected community is not deterministic. *Strength* [16] exploits node strength and belonging degree to detect the overlapping community structure. However, the performance of *Strength* degrades dramatically as the overlapping increases. Moreover, none of them has been applied to DTN or OSN. Although there are some distributed community detection algorithms [17], it is hard to apply them to DTN routing. This is because when communities are detected in a distributed way, for two nodes in the same community, their detected communities can be totally different.

In this article, we first address the problems of community detection in weighted network, and then exploit community for data forwarding in DTN and worm containment in OSN to demonstrate how community can facilitate these network designs. Our detailed contributions are as follows:

- We design a novel community detection algorithm for weighted networks.
- We introduce two metrics: intra-centrality and inter-centrality, to characterize nodes in communities, based on which we propose an efficient data forwarding algorithm for DTN and a worm containment strategy for OSN.
- Based on real traces, we study the performance of the proposed community detection algorithm, the data forwarding algorithm and the worm containment strategy, and compare them to existing work.

The rest of this paper is organized as follows. Section 2 presents our community detection algorithm for weighted networks. Then, we introduce intra-centrality and inter-centrality in Section 3. Section 4 presents our forwarding algorithm for DTN, and worm containment strategy for OSN. Section 5 evaluates the performance of our proposed algorithms, and Section 6 concludes the paper.

2 Community Detection in Weighted Networks

2.1 Preliminary

Let $G = (V, E)$ represent a weighted and undirected network, where V denotes the set of nodes and E denotes the set of edges. For two nodes $u, v \in V$, the edge between them is denoted as $(u, v) \in E$, and w_{uv} denotes the weight of the edge. The network community structure is denoted by $\mathcal{C} = \{C_1, C_2, C_3, ...\}$, where $C_i \in \mathcal{C}$ denotes a community. We do not require $C_i \cap C_j = \emptyset$ which means the communities may be overlapped. For simplicity, we denote C_i as C if there is no confusion. For a node $u \in V$, k_u, N_u are the node degree and the neighbor set of node u, respectively, where $k_u = \sum_{v \in N_u} w_{uv}$. For a community C and a node u, the *belonging degree* $B(u, C)$ between node u and community C is defined as

$$B(u, C) = \frac{\sum_{v \in C} w_{uv}}{k_u}. \tag{1}$$

Thus, when all neighbors of a node u are included in community C, $B(u, C) = 1$.

In the literature, *modularity* (Q) [18] is the most popular function to quantify the quality of the detected community structure in binary networks. It measures the difference between the actual density of edges within the detected communities and the density in the communities if the edges are randomly distributed in the network. Later, Newman [19] extended the concept of modularity to weighted network as follows.

$$Q = \frac{1}{2m} \sum_{uv} \delta(u, v)(A_{uv} - \frac{k_u k_v}{2m}), \tag{2}$$

where A is the adjacency matrix (A_{uv} represents the weight of the edge between node u and v, and if there is no edge between u and v, $A_{uv} = 0$), $m = \frac{1}{2} \sum_{uv} w_{uv}$ (the weights of all the edges in network), k_u and k_v are the degrees of node u and v. $\delta(u, v)$ yields one if nodes u and v are in the same community, zero otherwise.

However, (2) does not take overlapping communities into consideration, where a node can belong to more than one community. Thus, we generalize Q to measure overlapping communities as follows.

$$Q = \frac{1}{2m} \sum_{uv} \delta(\rho_u, \rho_v)(A_{uv} - \frac{k_u k_v}{2m}), \tag{3}$$

where

$$\rho_u = \underset{C \in \mathcal{C}_u}{\arg} \max B(u, C). \tag{4}$$

$\mathcal{C}_u$ denotes the community set to which node u is assigned, and $\mathcal{C}_u$ is a subset of $\mathcal{C}$. ρ_u denotes the community in $\mathcal{C}_u$, to which node u has the most weight of belonging degree. Similar to (2), $\delta(\rho_u, \rho_v)$ yields one if ρ_u and ρ_v are the same, zero otherwise. For non-overlapping communities, where each node is assigned to only one community, i.e., $|\mathcal{C}_u| = 1$, ρ_u actually denotes the community to which node u is assigned, and if u and v are in the same community, $\delta(\rho_u, \rho_v) = 1$. Thus, (3) is consistent with the definition of modularity for non-overlapping communities in [18], and (3) can be used to quantify both overlapping and non-overlapping community structure.

2.2 Conductance Function

We use conductance to identify the community that has more internal connectivity than external connectivity. Conductance is a natural and widely-adopted notion of community goodness [20] and is also known as the normalized cut metric. The conductance $\Phi(C)$ of community $C \in \mathcal{C}$ in network G is defined as

$$\Phi(C) = \frac{cut(C, G \backslash C)}{w_C}, \tag{5}$$

where $cut(C, G\backslash C)$ denotes the weights of the cut edges of C and w_C denotes the weights of all edges in community C including the cut edges. For example, for the community that consists of nodes b and d ($C = \{b, d\}$) as shown in Fig. 1, $cut(C, G\backslash C) = w_{ab} + w_{ad} + w_{cb} + w_{cd} = 4$, $w_C = w_{ab} + w_{ad} + w_{cb} + w_{cd} + w_{bd} = 15$, thus $\Phi(C) = \frac{4}{15}$. With lower conductance, more edge weights are within the community and the identified community is better. However, minimizing the conductance of community structure (the sum of the conductance of individual community) is NP-hard as proved in [21] by reducing it to the *Normalized Cut* problem. Thus, we propose a heuristic algorithm in the next subsection.

Unlike existing community detection algorithms in weighted networks, such as *COPRA* and *Strength*, our algorithm explores the property of community (i.e. conductance) and exploits it as the criterion to detect community. As a result, our algorithm can detect communities more accurately and efficiently than existing works (see Section 5).

2.3 The Community Detection Algorithm

Different from algorithms designed for binary networks, the edge weight should be taken into consideration in weighted networks. The intuition behind our algorithm is as follows:

- If the edge weight between two nodes is high enough, the two nodes should be in the same community.
- If the belonging degree of a node to one community is high enough to decrease the conductance of the community, the node should be included in the community.

Our detection algorithm works as follows. For a given network G, we first identify two nodes that are connected by the highest weight edges as a community C and calculate its conductance $\Phi(C)$. In the expanding process, nodes adjacent to C (denoted as N_C) are found. We then choose the node in N_C with the highest belonging degree to C and combine it with C to form a new community C'. If $\Phi(C') < \Phi(C)$, the expanding process is continued for community C'; otherwise, C is designated as a detected community. Then, the edges within community C (denoted as E_C) is removed from the edge set, and the whole process is repeated until the edge set is empty. For completeness, the pseudo code of the detecting algorithm is shown in Appendix D, available in the online supplemental material.

We use an example to illustrate how the expanding process works. As shown in Fig. 1, assuming edge (h, i) has the highest weight among the remaining edges and $C = \{h, i\}$ is identified as a community, the conductance of C is $\Phi(C) = \frac{4}{9}$. Then we find all the adjacent nodes to C, which are f and g. The belonging degree to C of node g and f are $\frac{1}{2}$ and $\frac{2}{13}$, respectively. Thus, node g is chosen to form C' ($\{g, h, i\}$) together with C. Since $\Phi(C')$ is $\frac{4}{11}$ which is less than $\Phi(C)$, node g is added into C and

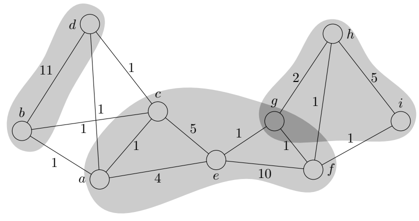


Fig. 1: Community detection in weighted networks.

hence $C = \{g, h, i\}$. The expanding process continues for newly formed C. Among the adjacent nodes e and f, f has higher belonging degree to C than e. Thus, f is selected to form C' ($\{f, g, h, i\}$). Since $\Phi(C') = \frac{11}{21}$ and $\Phi(C') > \Phi(C)$, the expanding process for C stops and $C = \{g, h, i\}$ is designated as the detected community.

Unlike binary network where a threshold is used to determine a community such as in [4], for weighted networks we cannot fix the threshold of conductance due to the heterogeneous distribution of weights. Instead, we use the conductance of the current community as the criterion to determine whether the community should be expanded. The threshold value is dynamically changed as the chosen community varies and it is updated after each iteration. A neighboring node is eligible to be added into a community only if the newly formed community has a lower conductance. This is because larger community has more internal connections, and then the conductance becomes smaller. According to (5), if $\Phi(C) > \Phi(C')$, we have:

$$\begin{aligned}
\frac{cut(C, G\backslash C)}{w_C} &> \frac{cut(C, G\backslash C) + \Delta cut}{w_C + \Delta w} && (6)\\
\frac{cut(C, G\backslash C)}{w_C} &> \frac{cut(C, G\backslash C) + k_u(1 - 2B(u, C))}{w_C + k_u(1 - B(u, C))} && (7)
\end{aligned}$$

From (7), we have

$$\begin{aligned}
B(u, C) &> \frac{w_C - cut(C, G\backslash C)}{2 \times w_C - cut(C, G\backslash C)}\\
&> \frac{1 - \Phi(C)}{2 - \Phi(C)}.
\end{aligned}$$

When $B(u, C)$ is larger than a threshold $\theta = \frac{1-\Phi(C)}{2-\Phi(C)}$, node u will be added into community C.

The detected community may overlap with other communities, and our algorithm can detect overlapping communities. This is because we do not require each node to be exclusively included by one community and the temporary community can go cross existing communities during the expanding process. If two communities overlap and the overlapping nodes that belong to both communities are more than 50% of nodes in either of these two communities, they need to be merged. As observed in the experiments, for overlapping communities after community detection, two communities are either highly overlapped (overlapping nodes are much more than 50%) or rarely overlapped (overlapping nodes are much less than 50%). Thus, 50% is chosen as the threshold for community merging. Moreover, our algorithm can be applied for dynamic networks by adopting the techniques proposed in [4] with some changes (e.g., using $\sum_{C \in \mathcal{C}} \Phi(C)$ as the objective function instead of the internal density function used in [4]).

Figure 1 shows an example of the detected communities in weighted network using our detection algorithm, where three communities are detected and the overlapped part is also uncovered as shown in the darker shaded region.

To analyze the time complexity of the algorithm, initially $|V|$ nodes are viewed as $|V|$ individual communities. After the detection process, $|\mathcal{C}|$ communities are detected. As the node with the highest belonging degree is added to the temporary community at each expanding step, there are at most $|V| - |\mathcal{C}|$ expanding steps. For each expanding step, the node with the highest belonging degree is searched with time complexity $N_C * d$, where d is the average number of edges connected with each node. As N_C is at most as large as $|V|$, the worst time complexity of our detection algorithm is $|V|^2$.

In addition, we give the approximation ratio of our algorithm in terms of modularity Q in Appendix A.

3 Intra-centrality and Inter-centrality

Based on the community detection algorithm presented in the last section, we define two metrics: intra-centrality and inter-centrality, which can be used for data forwarding in DTN and worm containment in OSN.

Definition 1. *The **Intra-centrality** of a node is defined as the number of shortest paths between pairs of nodes in the same community that go through it.*

Let $\varphi_u(C)$ denote the intra-centrality of node u. Then,

$$\varphi_u(C) = \sum_{v,w \in C} \lambda(u, (v \to w)), \quad u \in C, C \in \mathcal{C},$$

where $(v \to w)$ denotes the shortest path between vertex v and w, and $\lambda(u, (v \to w))$ yields one when node u is on $(v \to w)$, zero otherwise.

The shortest path should not go beyond a community, which means that all the shortest paths should be within the community. To find the shortest path, we use weighted network analysis where the distance between two directly connected nodes is the reciprocal of the edge weight (for example, in Fig. 1, the shortest distance between node c and f is $\frac{1}{w_{ce}} + \frac{1}{w_{ef}} = \frac{3}{10}$). Since a node may belong to multiple communities, it may have multiple intra-centrality values, each corresponding to a community.

Intra-centrality measures the influence of nodes within a community. Within a community, nodes with higher intra-centrality are more popular; i.e., they have more connections with other nodes and contact them more frequently like hubs.

Definition 2. *The **Inter-centrality** of a node for two communities is defined as the number of shortest paths between two nodes in these two communities that go through it.*

Let $\phi_u(C_i, C_j)$ denote the Inter-centrality of node u for communities C_i and C_j. Then,

$$\phi_u(C_i, C_j) = \sum_{\substack{v \in C_i, w \in C_j \\ v, w \notin C_i \cap C_j}} \lambda(u, (v \to w)), \quad u \in V, C_i, C_j \in C,$$

where the overlapped nodes between two communities are excluded ($v, w \notin C_i \cap C_j$). Each node has an inter-centrality value for each pair of detected communities.

Inter-centrality measures the capability of nodes to connect two communities. Nodes with higher inter-centrality represent more connections between communities. That is, the communications between two communities most likely go through the nodes with higher inter-centrality and hence removing them will more likely isolate these two communities.

4 Community based Applications

In this section, we introduce two community based applications: data forwarding in DTN and worm containment in OSN. We also present the proposed algorithms for data forwarding and worm containment based on intra-centrality and inter-centrality.

4.1 Data Forwarding in Delay Tolerant Networks

Recent work (e.g., *BubbleRap* [6] and *AFOCS* [4]) has shown that community based data forwarding algorithms can significantly reduce the number of data replications while maintaining similar data delivery ratio and data delivery time in DTN. However, as community detections in these algorithms are based on binary networks, they also have some weaknesses. For example, *BubbleRap* may use inaccurate centrality and most traffic between two communities in *AFOCS* may not go through the overlapped nodes. Different from them, our solution is based on our community detection algorithm designed for weighted networks, where edge weight between nodes is formulated as their contact frequency in DTN. Based on intra-centrality and inter-centrality, we design a better forwarding algorithm.

Our data forwarding algorithm works as follow. Suppose a node (sender) has a message destined for another node (receiver). If the sender and the receiver are in the same community, the sender only forwards the packet to the node encountered (relay) that has higher intra-centrality value. If the two nodes are in different communities, the sender forwards the packet to the relay which satisfies one of the following conditions:

- The relay is in the same community of the receiver.
- The relay has higher inter-centrality than the sender if the belonging degrees of the sender and the relay to receiver's community are both zero or non-zero.

- The relay has non-zero belonging degree to the receiver's community if the belonging degree of the sender to that is zero.

The pseudo code and an example of the forwarding algorithm is provided in Appendix D and B, respectively.

Different from existing community-based algorithms, our forwarding algorithm categorizes data forwarding into forwarding within community and forwarding between communities. The intuitions behind our forwarding algorithm can be summarized as following:

- The nodes that are more influential between two communities (high inter-community) can forward the message to the community(s) of destination more quickly.
- The nodes that are more popular within community (high intra-community) have more chances to deliver the message to the destination.

By differentiating data forwarding within community and between communities based on intra-centrality and inter-centrality, our forwarding algorithm can achieve high data delivery ratio with less message overhead. Moreover, to perform data forwarding in DTN, the proposed algorithm requires that each node knows the values of inter-centrality and intra-centrality. With the common assumption that community is a social property and has some long-term characteristics, we can obtain these values based on some previous history. Once nodes have the values of these two metrics, they can perform routing in a distributed way.

Moreover, considering the load balancing problem (i.e. sending too many messages through the same node), according to the design of the forwarding algorithm, if most network traffic is within community (i.e., source and destination are in the same community), the forwarding algorithm may have the load balancing problem since the forwarding within community is carried out by the nodes with higher intra-centrality. Depending on the number of nodes with high intra-centrality and the capability of these nodes, there may or may not have congestion. However, this is common to most social based data forwarding algorithms which have been demonstrated to outperform other algorithms. If most traffic is between communities, the forwarding algorithm hardly has the load balancing problem unless most traffic is between a specific pair of communities since the forwarding between two communities is fulfilled by the nodes with higher inter-centrality between these two communities (these nodes are different for different pair of communities).

4.2 Worm Containment in Online Social Networks

With online social networks (OSN) such as Facebook and Twitter, people can always keep in touch with friends and family by sharing news, photos and videos. Recently, OSN becomes more and more popular, meanwhile it also becomes a fertile ground for virus and worm propagation.

Community based worm patching schemes for cellular networks and OSN have been studied in [12] and [4], respectively. The intuition behind these schemes is to contain worms within infected community before they spread out. In [12], separators (i.e., key nodes that separate network partitions) are patched. Similarly, the neighbors of overlapped nodes between two communities are patched in [4]. Both strategies choose the nodes located on the boundary of communities to be patched first. However, these schemes do not consider about containing the worm propagation within community. Without considering how to control the worms within community, the worms could spread the whole network quickly. Recently, Han *et al.* [22] proposed to vaccinate influential users identified through betweenness centrality to control infectious disease. Due to the similarity between infectious disease control and worm containment, their solution can also be applied to worm containment. However, since betweenness centrality measures global influence, it may not effectively contain worms within a group of nodes, and thus all the nodes would be infected eventually.

Different from [12] and [4], our worm containment strategy is based our community detection algorithm designed for weighted networks, where the edge weight between nodes is formulated as their contact (e.g., wall post in Facebook) frequency in OSN. Moreover, our worm containment strategy not only considers how to isolate communities, but also considers how to slow down the worm spread within a community. In worm propagation, after a node is infected, the malicious node may infect the hub node in the community. Then, most nodes in the community are infected quickly. Finally, neighboring community will be infected by the adjacent nodes. Due to the characteristics of slow start and exponential propagation exhibited by worms, by slowing down the worm propagation at beginning, we will have more opportunities to contain the worms within the infected communities and prevent them from spreading to other communities. The intuitions behind our strategy is as follows:

- Patching nodes with high intra-centrality will slow down the worm propagation within a community. Intra-centrality measures the node popularity within a community. Since nodes with high intra-centrality can be easily infected and infect others after they are infected, patching nodes with high intra-centrality is more effective and efficient.
- Patching nodes with high inter-centrality will slow down the worm propagation between communities. Inter-centrality measures the node influence on the connections between two communities, and thus patching nodes with high inter-centrality will isolate communities to prevent the worm propagation between communities.

When a node receives the patch, it will be immune to the worm. However, distributing patches to all nodes at

TABLE 1: Parameter settings in benchmark

Parameter	Value	Meaning	Parameter	Value	Meaning
N	1000, 5000	No. of nodes	μ_w	0.1, 0.3	Mixing parameter for edge weights
μ_t	0.1, 0.3	Mixing parameter for topology	ξ	2	Exponent for weight distribution
k_{max}	50, 100	Maximum node degree	τ_1	2	Minus exponent for degree sequence
k	30, 50	Average node degree	τ_2	2	Minus exponent for community size distribution
γ	0 to 0.5	Overlapping fraction	o_m	2	No. of communities of overlapping nodes

the same time may not always be feasible; e.g., there are bandwidth limitations in cellular networks. Thus, we determine the proper patching order for these nodes based on a patching score. The higher the patching score is, the sooner the node will be patched. The patching score is calculated by combining the normalized intra-centrality (normalized by the number of pairs of nodes within community) and inter-centrality (normalized by the number of pairs of nodes between communities). More specifically, for a node u, the patching score is calculated as

$$\psi_u = \beta\varphi'_u + (1-\beta)\phi'_u, \tag{8}$$

where $\beta \in [0,1]$, ψ_u denotes the patching score, φ'_u denote the normalized intra-centrality and ϕ'_u denotes the normalized inter-centrality.

For worm containment, the patching process is initialized when the *infection rate* (i.e., the fraction of infected nodes over all the nodes) reaches the predefined alarm threshold α. In [12] and [4], the patching scheme is designed regardless of α, which means the patching scheme is fixed. However, that may not be the best strategy. When α is low, worms propagate mainly within communities since nodes contact others more frequently within community than between communities. On the other hand, when α is high, worms may spread between communities since more nodes have been infected. Thus, the patching scheme should be adaptive based on the alarm threshold α.

By tuning β, our patching scheme becomes adaptive, and can achieve better performance. As discussed above, patching nodes with high intra-centrality will control the worm propagation within communities and patching nodes with high inter-centrality will contain the worm propagation between communities. When the alarm threshold α is low, the worms mainly spread within communities; thus, intra-centrality should have a large weight (β should be large). On the other hand, when α is high, nodes with high intra-centrality may have been infected; thus, inter-centrality should have a large weight (β should be small).

The pseudo code of computing the patching score is provided in Appendix D. As a node may have multiple inter-centrality and intra-centrality values, only the largest values are used to calculate the patching scores. By considering both intra-centrality and inter-centrality, our worm containment strategy can slow down worms within communities and between communities. It is worth to mention that our strategy can also be applied to other applications such as infectious diseases control, due to their similarity to worm containment.

5 Performance Evaluations

In this section, we evaluate the performance of the three proposed algorithms (i.e., community detection, data forwarding, and worm containment) and compare them to existing works.

5.1 Community Detection

We compare the performance of our conductance-based community detection algorithm (denoted as *Conductance*) with two algorithms – *COPRA* [15] and *Strength* [16], both of which can detect overlapping communities for weighted network. Since it is hard to verify the detected communities for real networks, the evaluation of these algorithms is based mainly on synthetic networks and also on real networks.

We use the synthetic weighted networks generated by the well-known benchmark proposed in [23]. It provides power-law distributions of node degree and community size, and it allows overlaps between communities. There are many parameters to control the generated network and the settings of these parameter are shown in Table 1. Note that nodes may belong to more than one community, which we call overlapping nodes. γ is the fraction of overlapping nodes over all nodes. The number of communities that overlapping nodes belong to is denoted by o_m. In addition, the benchmark also gives the community structure of the generated network, which is referred to as the *ground truth* used to compare with other algorithms. Compared to [15] 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 that of *COPRA* to make it more general. For example, the overlapping fraction can be changed from 0 to 0.5 in networks with 1000 nodes and 5000 nodes, respectively.

We evaluate these algorithms based on the following three widely-adopted metrics:

- **Normalized Mutual Information** (NMI), which has been proposed in [24] to measure the similarity between different parts of the network. NMI is normalized, where NMI$(X|Y) \in [0,1]$. Note that higher NMI is better. In the evaluation, we compare the detected communities by each algorithm with ground truth to obtain NMI.
- **Number of communities**. A good detection algorithm should detect roughly the same number of communities as that generated by the benchmark denoted as the *ground truth*.
- **Modularity**, as discuss in Section 2.1, it is defined as the fraction of edges within communities minus

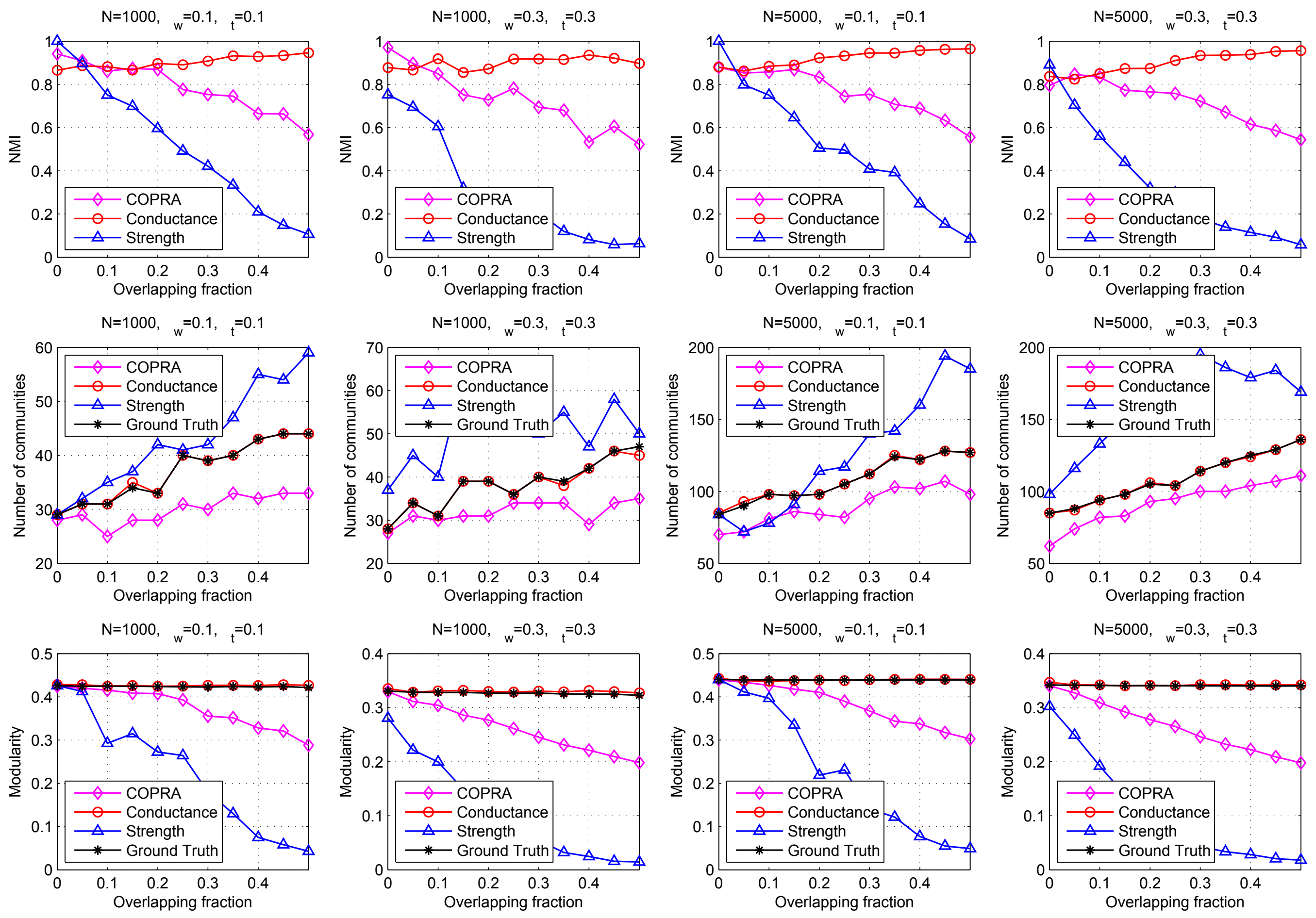


Fig. 2: The quality (in terms of **NMI**, **Number of Communities**, **Modularity**) of the community structure detected by **Conductance**, **COPRA**, **Strength** and **Ground Truth** for different parameter settings.

the expected value of the same quantity if the edges are assigned at random.

We evaluate the performance of these algorithms with these metrics for different parameter settings: μ_w=μ_t=0.1, 0.3, N=1000, 5000, and γ is changed from 0 to 0.5 for each setting. Fig. 2 shows the experimental results of community detection by these three algorithms: *Conductance*, *COPRA*, *Strength* and the ground truth.

Comparisons in terms of NMI: Fig. 2 shows the NMI between ground truth and each algorithm. Note that higher NMI is better (the NMI of the ground truth is always 1 since it compares to itself, and thus it is omitted). We can see from Fig. 2 that *Conductance* has very stable performance with varying overlapping fraction for all settings and *Conductance* outperforms other algorithms except when γ=0. From Fig. 2, we can also see the performance of *Strength* degrades dramatically when the mixing rates increase (from μ_w=μ_t=0.1 to μ_w=μ_t=0.3) and when the overlapping fraction γ increases. The performance of *COPRA* also decreases significantly when γ increases. Unlike these two algorithms, the NMI of *Conductance* is stable and even increases slightly with the increase of γ.

Comparisons in terms of the number of communities: Fig. 2 also shows the number of communities (in ground truth) and the number of communities detected by *Conductance*, *COPRA* and *Strength*. The number of communities detected by *COPRA* is always smaller than the ground truth. On the other hand, *Strength* always detects more communities than the ground truth. Moreover, the differences between ground truth and these two algorithms are getting larger when the mixing rates and γ increase. In contrast, the number of communities detected by *Conductance* is almost identical to the ground truth for all settings.

Comparisons on Modularity: As shown in Fig. 2, similar with NMI, for both *COPRA* and *Strength*, the modularity dramatically decreases in all settings with the increase of γ. *Strength* has the worst performance. *Conductance* and ground truth have almost identical modularity and the modularity is highly steady with the overlapping fraction.

In summary, the performance of both *COPRA* and *Strength* decreases with the raise of the overlapping fraction. That is because the label propagation adopted by *COPRA* and the maximal node strength employed by *Strength* cannot accurately recognize the overlapped communities; i.e., *COPRA* may identify two overlapped communities as one community while *Strength* may identify them as more than two communities. Although *COPRA* and *Strength* may have better performance in

TABLE 2: Summary of the MIT Reality trace

Trace	MIT Reality
Network type	Bluetooth
No. of nodes	97
No. of interval contacts	114046
Duration (days)	246
Granularity (seconds)	300
Pairwise contact frequency (per day)	0.10

terms of NMI when the overlapping fraction is small, *Conductance* always outperforms them significantly in terms of the number of communities and modularity. In addition, the performance of *Conductance* in term of NMI even increases with the overlapping fraction.

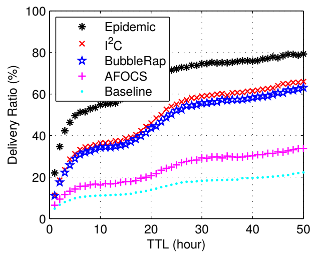
Thus, we can conclude that *Conductance* has the ability to detect both non-overlapping and highly overlapping communities for weighted networks. Under various conditions, the performance of *Conductance* is much better than existing algorithms in synthetic networks.

5.2 Data Forwarding

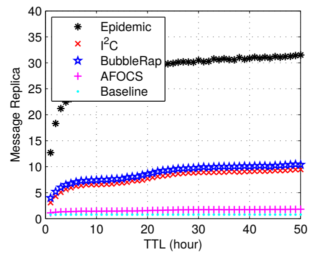
To evaluate the performance of our data forwarding algorithm in DTN, experiments are conducted based on the MIT Reality trace [25], which contains contacts among users carrying Bluetooth devices. Bluetooth devices periodically discover peers in the neighborhood and record their contacts. The detail of the trace is summarized in Table 2, where edge weight between two nodes represents the contact frequency.

In our experiment, each node sends 500 messages to other randomly selected nodes. Messages will be discarded if they are not successfully delivered within Time-to-live (TTL). We compare our **I**ntra-centrality and **I**nter-**C**entrality based forwarding algorithm (called I^2C) with other four forwarding strategies: 1) *Epidemic* [10], 2) *BubbleRap* [6] which uses *k-clique* [13] as the community detection algorithm, 3) *AFOCS* [4], and 4) *Baseline* where the source keeps the message until it encounters the destination. We evaluate all these algorithms on two message forwarding modes: forwarding with message duplication (Fig. 3a, 3b, 3c and 3d) and forwarding without message duplication (Fig. 3e and 3f). With message duplication, the algorithms are compared in term of data delivery ratio, data delivery time, and message replica. For forwarding without message duplication, data delivery ratio and data delivery time are considered. Note that the same random seeds are used to generate source/destination pairs for all algorithms in each simulation run. Moreover, the simulations are repeated 100 times (each with different seed) and the results are averaged.

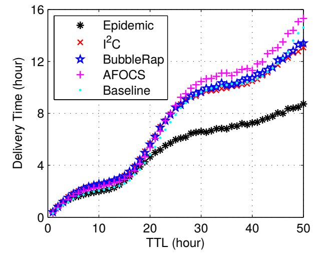
Fig. 3 shows the results of data forwarding on the MIT reality trace, where TTL varies from 1 to 50 hours. As in *Epidemic* the message is always forwarded to the node encountered, *Epidemic* has the highest delivery ratio and forwarding cost (message replicas) as shown in Fig. 3a, 3b. On the other hand, *Baseline* has the lowest delivery ratio and forwarding cost, since nodes in this algorithm forward the message only when it reaches the



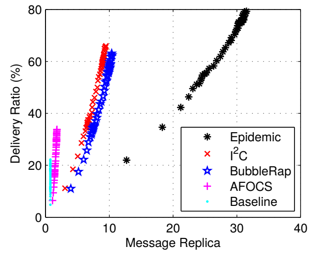
(a) deliver ratio



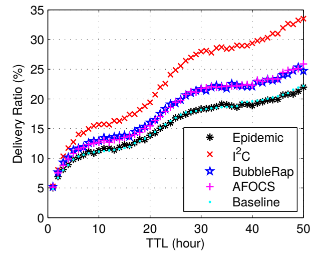
(b) message replicas



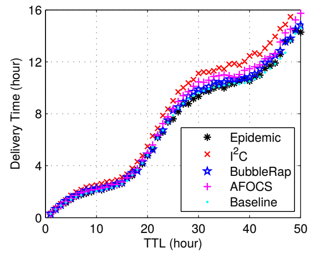
(c) delivery time



(d) relation between delivery ratio and message replica



(e) delivery ratio without message duplication



(f) delivery time without message duplication

Fig. 3: Performance of the data forwarding algorithms – **Epidemic**, $\mathbf{I^2C}$, **BubbleRap**, **AFOCS** and **Baseline** in terms of data delivery ratio, data delivery time, and message overhead based on the MIT reality trace.

destination. Fig. 3a shows that the delivery ratio of I^2C is higher than that of *BubbleRap*, and *BubbleRap* incurs more message replicas than I^2C, up to 40%, as shown in Fig. 3b. *AFOCS* has both the second lowest delivery ratio and message replicas. *Epidemic* has much lower delivery time than all other algorithms when TTL is larger than 20 hours, I^2C, *BubbleRap* and *Baseline* have similar delivery time and *AFOCS* is the worst as illustrated by Fig. 3c. Fig. 3d shows the relation between delivery ratio and message replicas for each algorithm. *AFOCS*, I^2C and *BubbleRap* span from *Baseline* to *Epidemic*, with I^2C outperforming the other two.

Fig. 3e and Fig. 3f show the delivery ratio and delivery time of the data forwarding algorithms without message duplication. Since there is no dedicated strategy behind *Epidemic*, *Epidemic* is just slightly better than *Baseline* in terms of delivery ratio. *BubbleRap* and *AFOCS* are equivalent, where *BubbleRap* is based on global centrality and *AFOCS* utilizes overlapped nodes as relay nodes for message forwarding. I^2C is the best one and the performance is up to 50% better than other algorithms, although the delivery time of I^2C is only slightly more than others.

TABLE 3: Facebook trace summary

Trace	Facebook
No. of nodes	12123
No. of edges	31932
Average node degree	21.4
No. of contacts	129462
Duration (days)	184

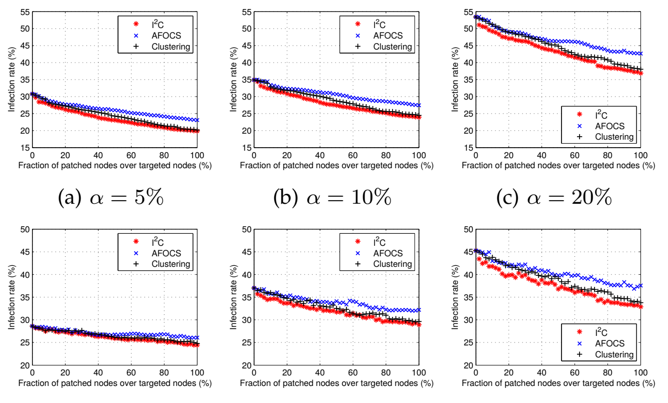
To summarize, there exists a tradeoff between delivery ratio and message replica. *Epidemic* and *Baseline* are the upper bound and the lower bound of performance and cost for data forwarding algorithms in DTN, respectively. All other algorithms (not just the algorithms evaluated) span between them as illustrated in Fig. 3d, where all other algorithms sit in the regions between *Epidemic* and *Baseline*. As demonstrated in Fig. 3, I^2C achieves a good balance between performance and cost, and is the most efficient algorithm. The reason that I^2C outperforms *BubbleRap* and *AFOCS* is two-fold: I^2C is based on the detected communities in weighted networks while *BubbleRap* and *AFOCS* are based on that in binary networks; I^2C is carried out based intra and inter-centrality while *BubbleRap* and *AFOCS* are only based on global centrality and overlapped nodes, respectively.

5.3 Worm Containment

To evaluate the worm containment strategies, we use the Facebook trace from [26]. It contains friendship information and wall posts among Facebook users in the New Orleans regional network for more than four years. We choose a partial trace which spans half year (from 7/1/2007 to 12/31/2007). The chosen trace is summarized in Table 3, where the contact between two nodes is the wall post and edge weight is the contact frequency. We use the worm propagation model similar to that in [12][4], which mimics the behaviors of the famous worm *Koobface* spread out in Facebook. We assume that the worms are able to explore the friendship information for propagation (such as sending out message including malicious links).

At the very beginning, we randomly choose 0.05% of nodes as the seed set of worm sources to initiate the infection process. Worms propagate when infected nodes contact other nodes with a *propagation ratio* (σ); i.e., when a node contacts with an infected node, it has a possibility of σ to be infected. The time taken for the worm to propagate from one node (user) to its friend is inversely proportional to the contact frequency between the two nodes. The patching process is initiated when the infection rate reaches the predefined *alarm thresholds* α (the infection rate is the fraction of infected nodes over all nodes). When a node receives a patch, it will be immune to worms and recovered if it was infected.

We compare I^2C based worm containment strategy with *AFOCS* [4] and the cluster based scheme [12] denoted as *Clustering*. Unlike I^2C and *Clustering*, where the sequence of nodes to be patched is determined, *AFOCS* chooses the neighboring nodes of overlapped nodes between communities to be patched. Thus, the nodes



(a) $\alpha = 5\%$ (b) $\alpha = 10\%$ (c) $\alpha = 20\%$

(d) $\sigma = 25\%$ (e) $\sigma = 50\%$ (f) $\sigma = 75\%$

Fig. 4: Performance of worm containment algorithms – **$\mathbf{I^2C}$**, **AFOCS** and **Clustering** based on the Facebook trace, where propagation ratio $\sigma = 100\%$ for (a), (b) and (c) and alarm threshold $\alpha = 20\%$ for (d), (e), (f).

to be patched, which are referred to as *targeted nodes*, selected by *AFOCS*, are deterministic for a network and the number is 985 for the Facebook trace. To compare the performance with different schemes, we choose the same number of targeted nodes for I^2C and *Clustering* according to patching score and priority, respectively. The worm propagation is simulated for 30 days after the alarm threshold is reached. Moreover, simulations are repeated at 100 times and the results are averaged.

In some cases, the alarm threshold α might be unknown. Thus, we first fix $\beta = 0.5$ for I^2C and compare it with *AFOCS* and *Clustering* for different propagation ratios and alarm thresholds (α is used only for performance comparison), and then evaluate its effects on I^2C.

Fig. 4a, 4b and 4c show the infection rates achieved by different algorithms for alarm threshold $\alpha = 5\%$, 10% and 20%, respectively, when $\sigma = 100\%$, with varying faction of patched nodes over targeted nodes. For $\alpha = 5\%$ (Fig. 4a), where the patching process is initialized at an early stage, the infection rates are relatively low after 30 days of worm propagation. From these figures, we can see late patching will result in higher infection rate. Among these algorithms, our worm containment scheme has the lowest infection rates, which demonstrates that patching nodes with high patching score can effectively contain the worm propagation. As the nodes selected by *AFOCS* and *Clustering* do not effectively block worm propagation between communities, they have higher infection rates. Fig.4d, 4e and 4f show the infection rates achieved by different algorithms with different propagation ratio: $\sigma = 25\%$, 50% and 75%. As expected, a lower propagation ratio yields a lower infection rate. However, the propagation ratio does not affect the relative performance of these schemes; i.e., I^2C always performs the best for different σ.

Although we use fixed β for I^2C, it still outperforms *AFOCS* and *Clustering* for all different alarm thresholds and different propagation ratios as shown in Fig. 4. That is because *AFOCS* and *Clustering* do not consider about

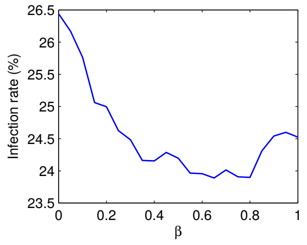


Fig. 5: Performance of $\mathbf{I^2C}$ with varying β ($\alpha = 10\%$).

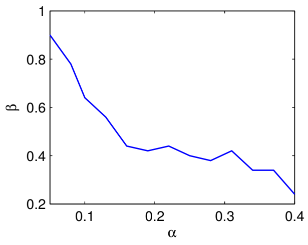


Fig. 6: Relation between α and β.

containing the worm propagation within community, and thus the worm can quickly propagate within community and eventually spread between communities.

Next, we evaluate the effect of β on the performance of I^2C. Fig. 5 shows the infection rate achieved by I^2C when $\alpha = 10\%$ and $\sigma = 100\%$ with varying β (the results when $\alpha = 5\%$, 20% and 40% are provided in Appendix C.1). As shown in Fig. 5, we can find the best β with which I^2C can achieve the lowest infection rate. Fig. 6 shows the relation between α and β, where I^2C achieves the best performance by adaptively adjusting β based on α. As analyzed in Section 4.2, when the alarm threshold is low (small α), since worms spread mainly within community, intra-centrality should have a large weight in determining the patching score (β is large). When the alarm threshold is high (large α), worms spread between communities and thus inter-centrality should have a large weight in determining the patching score (β is small). For a real system, β can be selected based on the relation between α and β shown in Fig. 6.

In summary, our community detection algorithm has better performance than existing algorithms. Based on the detected communities and the newly introduced two centrality metrics, I^2C algorithms outperform other forwarding algorithms and worm containment strategies.

More evaluation results including the performance of detection algorithms in real networks and the performance of I^2C algorithms based on the communities detected by these algorithms are provided in Appendix C.2 and C.3.

6 Conclusions

In the paper, we proposed a conductance-based community detection algorithm for weighted networks and designed an efficient data forwarding algorithm for DTN and a worm containment strategy for OSN based on two metrics – intra-centrality and inter-centrality. Simulation results on synthetic networks show that our community detection algorithm is much better than other algorithms in terms of NMI, the number of communities, and modularity. Simulation results on real DTN traces show that our forwarding algorithm outperforms other community-based algorithms in terms of data delivery ratio and data forwarding cost. Results on real OSN traces show that our worm containment strategy achieves lower infection rate than other algorithms and it is adaptive to different alarm thresholds.

References

[1] Z. Lu, Y. Wen, and G. Cao, “Community detection in weighted networks: Algorithms and applications,” in *IEEE PerCom*, 2013.

[2] M. Girvan and M. Newman, “Community structure in social and biological networks,” *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, 2002.

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

[4] N. Nguyen, T. Dinh, S. Tokala, and M. Thai, “Overlapping communities in dynamic networks: their detection and mobile applications,” in *ACM MobiCom*, 2011.

[5] E. Daly and M. Haahr, “Social network analysis for routing in disconnected delay-tolerant manets,” in *ACM Mobihoc*, 2007.

[6] P. Hui, J. Crowcroft, and E. Yoneki, “Bubble rap: Social-based forwarding in delay-tolerant networks,” *Mobile Computing, IEEE Transactions on*, vol. 10, no. 11, pp. 1576–1589, 2011.

[7] W. Gao, Q. Li, B. Zhao, and G. Cao, “Social-aware multicast in disruption-tolerant networks,” *IEEE/ACM Trans. Netw.*, vol. 20, no. 5, pp. 1553–1566, 2012.

[8] W. Gao, G. Cao, T. La Porta, and J. Han, “On exploiting transient social contact patterns for data forwarding in delay-tolerant networks,” *Mobile Computing, IEEE Transactions on*, vol. 12, no. 1, pp. 151 –165, 2013.

[9] T. Hossmann, T. Spyropoulos, and F. Legendre, “Know thy neighbor: Towards optimal mapping of contacts to social graphs for dtn routing,” in *IEEE INFOCOM*, 2010.

[10] A. Vahdat, D. Becker *et al.*, “Epidemic routing for partially connected ad hoc networks,” Technical Report CS-200006, Duke University, Tech. Rep., 2000.

[11] Q. Yuan, I. Cardei, and J. Wu, “Predict and relay: an efficient routing in disruption-tolerant networks,” in *ACM Mobihoc*, 2009.

[12] Z. Zhu, G. Cao, S. Zhu, S. Ranjan, and A. Nucci, “A social network based patching scheme for worm containment in cellular networks,” in *IEEE INFOCOM*, 2009.

[13] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, “Uncovering the overlapping community structure of complex networks in nature and society,” *Nature*, vol. 435, no. 7043, pp. 814–818, 2005.

[14] U. Raghavan, R. Albert, and S. Kumara, “Near linear time algorithm to detect community structures in large-scale networks,” *Physical Review E*, vol. 76, no. 3, p. 036106, 2007.

[15] S. Gregory, “Finding overlapping communities in networks by label propagation,” *New Journal of Physics*, vol. 12, p. 103018, 2010.

[16] D. Chen, M. Shang, Z. Lv, and Y. Fu, “Detecting overlapping communities of weighted networks via a local algorithm,” *Physica A: Statistical Mechanics and its Applications*, vol. 389, no. 19, pp. 4177–4187, 2010.

[17] P. Hui, E. Yoneki, S. Chan, and J. Crowcroft, “Distributed community detection in delay tolerant networks,” in *Proceedings of 2nd ACM/IEEE international workshop on Mobility in the evolving internet architecture*, 2007.

[18] M. Newman and M. Girvan, “Finding and evaluating community structure in networks,” *Physical review E*, vol. 69, no. 2, 2004.

[19] M. Newman, “Analysis of weighted networks,” *Physical Review E*, vol. 70, no. 5, 2004.

[20] J. Leskovec, K. Lang, and M. Mahoney, “Empirical comparison of algorithms for network community detection,” in *ACM WWW*, 2010.

[21] J. Shi and J. Malik, “Normalized cuts and image segmentation,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 8, pp. 888–905, 2000.

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

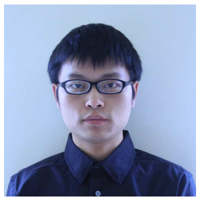
[23] A. Lancichinetti and S. Fortunato, “Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities,” *Physical Review E*, vol. 80, no. 1, 2009.

[24] A. Lancichinetti, S. Fortunato, and J. Kertész, “Detecting the overlapping and hierarchical community structure in complex networks,” *New Journal of Physics*, vol. 11, 2009.

[25] N. Eagle and A. Pentland, “Reality mining: sensing complex social systems,” *Personal and Ubiquitous Computing*, vol. 10, no. 4, pp. 255–268, 2006.

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

Zongqing Lu received the B.E. and M.E. degrees from Southeast University, China, and the Ph.D. 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, social networks, opportunistic networks, network privacy and security. He is a member of the IEEE.



Xiao Sun received the BS degree from Tianjin University, Tianjin, in 2008, and the ME degree from Chinese Academy of Sciences, Beijing, in 2011. He is currently working toward the PhD degree in the Department of Computer Science and Engineering, the Pennsylvania State University, University Park. His research interests include pervasive computing, mobile computing and mobile social networks. He is a student member of the IEEE.

Yonggang Wen received his Ph.D. degree in electrical engineering and computer science from Massachusetts Institute of Technology (MIT) in 2008. He is currently an Assistant Professor with School of Computer Engineering at Nanyang Technological University, Singapore. Previously, he has worked in Cisco as a Senior Software Engineer and a System Architect for content networking products. He has also worked as 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.

Guohong Cao received the BS degree from Xian Jiaotong University, China. He received the MS degree and PhD degree 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 board of IEEE Transactions on Mobile Computing, IEEE Transactions on Wireless Communications, and has served on the organizing and technical program committees of many conferences. He was a recipient of the NSF CAREER award in 2001. He is a Fellow of the IEEE.

Thomas La Porta received the BSEE and MSEE degrees from the Cooper Union, New York, and the PhD degree in electrical engineering from Columbia University, New York. He is a distinguished professor in the Department of Computer Science and Engineering, The Pennsylvania State University (Penn State), where he is also the director of the Networking and Security Research Center. Prior to joining Penn State, he was the director of the Mobile Networking Research Department, Bell Laboratories (Bell Labs), where he led various projects in wireless and mobile networking. He is the founding editor-in-chief of the IEEE Transactions on Mobile Computing. His research interests include mobility management, signaling and control for wireless networks, mobile data systems, and protocol design. He has published more than 100 technical papers and is the holder of 35 patents. He received a Thomas Alva Edison Patent Award. He is a fellow of the IEEE, the IEEE Computer Society, and Bell Labs.

Supplemental Material: Algorithms and Applications for Community Detection in Weighted Networks

Zongqing Lu, *Member, IEEE,* Xiao Sun, *Student Member, IEEE,* Yonggang Wen, *Member, IEEE,* Guohong Cao, *Fellow, IEEE,* and Thomas La Porta *Fellow, IEEE*

Appendix A Modulariy Analysis

It is hard to compare the conductances between the community structure detected by our algorithm and the one detected by the optimal solution. Since modularity Q is used to quantify the quality of the community structure as mentioned in Section **??**, in the following we compare our algorithm with the optimal solution in terms of modularity Q.

(**??**) denotes Q as a sum over all the edges. We can easily rewrite Q as a sum over the communities as

$$Q = \sum_{C \in \mathcal{C}} \left[\frac{w_C^{in}}{m} - \left(\frac{k_C}{2m} \right)^2 \right], \tag{9}$$

where $w_C^{in} = w_C - cut(C, G \backslash C)$ denotes the total weight of edges within community C, and k_C is the total degrees of nodes in C. In (9), the first term is the fraction of total weights inside the community, whereas the second term represents the expected fraction of weights.

According to the definition of conductance, we have

$$\Phi(C) = \frac{k_C - 2w_C^{in}}{k_C - w_C^{in}}. \tag{10}$$

Thus,

$$w_C^{in} = k_C \times \frac{1 - \Phi(C)}{2 - \Phi(C)}. \tag{11}$$

- *Z. Lu, X. Sun, G. Cao and T. La Porta are with the Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802. E-mail: zongqing@cse.psu.edu, xxs118@cse.psu.edu, gcao@cse.psu.edu and tlp@cse.psu.edu.*
- *Y.G. Wen is with the School of Computer Engineering, Nanyang Technological University, 639798 Singapore. E-mail: ygwen@ntu.edu.sg.*

From (9) and (11), we have

$$\begin{aligned} Q &= \sum_{C \in \mathcal{C}} \left[\frac{w_C^{in}}{m} - \left(\frac{k_C}{2m} \right)^2 \right] \\ &= \sum_{C \in \mathcal{C}} \left[\frac{k_C}{m} \times \theta - \left(\frac{k_C}{2m} \right)^2 \right] \\ &= \sum_{C \in \mathcal{C}} \frac{k_C}{m} \times \theta - \sum_{C \in \mathcal{C}} \left(\frac{k_C}{2m} \right)^2. \end{aligned} \tag{12}$$

As $\theta = \frac{1-\Phi(C)}{2-\Phi(C)}$, if we have the distribution of $\Phi(C)$, we can also derive the distribution of θ. Let random variable X denote $\Phi(C)$ and Y denote θ, respectively. Then, we have

$$\begin{aligned} F_Y(y) = P(Y \le y) &= P(\frac{1-X}{2-X} \le y) \\ &= 1 - P(X < \frac{1-2y}{1-y}) \\ &= 1 - \int_0^{\frac{1-2y}{1-y}} f_X(x) dx, \end{aligned}$$

and thus

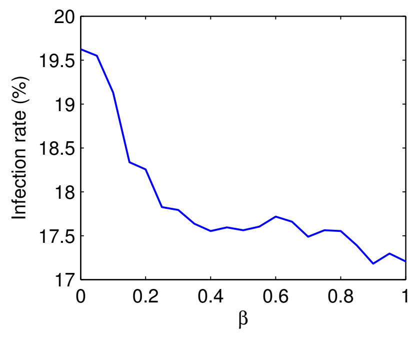
$$f_Y(y) = F'_Y(y) = \frac{1}{(1-y)^2} f_X(\frac{1-2y}{1-y}).$$

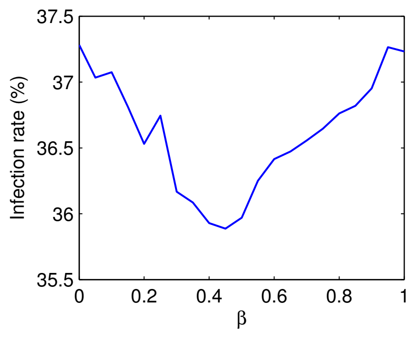
Then, the expected value of θ can be calculated as

$$\begin{aligned} E(\theta) &= \int_0^{\frac{1}{2}} y f_Y(y) dy \\ &= \int_0^{\frac{1}{2}} \frac{y}{(1-y)^2} f_X(\frac{1-2y}{1-y}) dy \end{aligned}$$

From (12), we can see it is hard to theoretically compute Q without a specific case. Thus, to approximate Q, we replace θ with the expected value in (12) and denote the modularity of our algorithm as Q', and since $\sum_{C \in \mathcal{C}} \frac{k_C}{2m} = 1$, we have

$$Q' \approx 2E(\theta) - \sum_{C \in \mathcal{C}} \left(\frac{k_C}{2m} \right)^2. \tag{13}$$


(a) $\alpha = 5\%$


(b) $\alpha = 20\%$

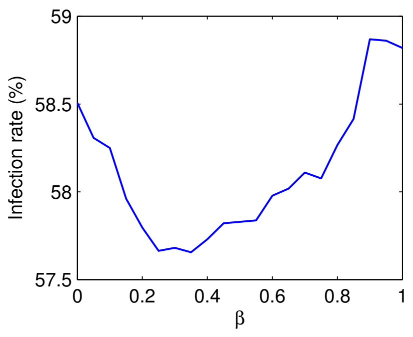

(c) $\alpha = 40\%$

Fig. 7: Performance of $\mathbf{I^2C}$ with varying β ($\sigma = 100\%$).

Let Q^* denote the modularity of the optimal solution. We assume the optimal solution always has the best θ for each community ($\Phi(C) = 0$) and thus $\theta = \frac{1}{2}$. So, we have

$$Q^* \leq 1 - \sum_{C \in \mathcal{C}} \left(\frac{k_C}{2m}\right)^2. \tag{14}$$

When the number of communities in network goes large (an implicit assumption is that there is no "giant" communities whose size is a large fraction of the network), $\sum_{C\in\mathcal{C}} \left(\frac{k_C}{2m}\right)^2$ is approaching to zero and thus negligible. Therefore, we have

$$\frac{Q'}{Q^*} \approx 2E(\theta).$$

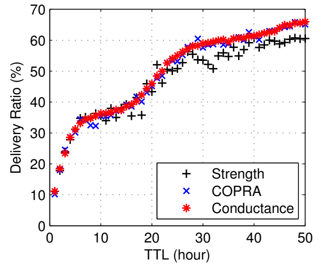
This ratio depends on the distribution of the conductance of communities. For example, if the conductance is an uniform distribution, then

$$\begin{aligned} E(\theta) &= \int_0^{\frac{1}{2}} \frac{y}{(1-y)^2} f_X(\frac{1-2y}{1-y})dy \\ &= \int_0^{\frac{1}{2}} \frac{y}{(1-y)^2} dy \\ &= \frac{1}{1-y} + \ln(1-y)\Big|_0^{\frac{1}{2}} = 1 - \ln 2 \end{aligned}$$

and thus $\frac{Q'}{Q^*} = 2 - 2\ln 2$.

Appendix B
Example of Data Forwarding Algorithm

Let us give an example for our data forwarding algorithm. As shown in Fig. **??**, node d sends a message to node i. By calculating the inter-centrality between community C_1=(b, d) and C_2=(g, h, i), we have $\phi_c(C_1, C_2) = 6$, $\phi_e(C_1, C_2) = 6$, $\phi_f(C_1, C_2) = 4$ and all the others are 0. When node b carries the message and encounters node a, as $B(b, C_2) = B(a, C_2) = 0$ and $\phi_b(C_1, C_2) = \phi_a(C_1, C_2) = 0$, node b will not forward the message to a. However, node b will forward the message to c if it encounters c, since $\phi_b(C_1, C_2) < \phi_c(C_1, C_2)$. When node c holds the message, it will forward the message if it encounters node e or f, because $B(e, C_2) > 0$ and $B(f, C_2) > 0$ while $B(c, C_2) = 0$. Assuming node f receives the message, it will forward the message to any encountered node that belongs to C_2 (assuming node g receives the message from f). Within community C_2, the message is routed to the destination i according to intra-centrality. If node g encounters node h, it will forward the message to node h since $\varphi_h(C_2) > \varphi_g(C_2)$ and finally node i will receive the message from h.


(a) data forwarding

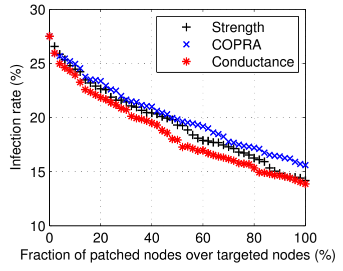

(b) worm containment

Fig. 8: Performance of $\mathbf{I^2C}$ over community structures detected by different detection algorithms.

Appendix C
Additional Performance Evaluations

C.1 $\mathrm{I^2C}$ Algorithm in Worm Containment

Fig. 7 shows the infection rate achieved by I^2C when $\alpha = 5\%$, 20% and 40% with varying β. As shown in Fig. 7, for each α, we can find the best β with which I^2C can achieve the lowest infection rate. From Fig. 7, we can also see that for worm containment considering intra-centrality and inter-centrality together has better performance than considering only intra-centrality ($\beta = 1$) or only inter-centrality ($\beta = 0$).

C.2 Detection Algorithms in Real Networks

We also evaluate *Conductance*, *Strength* and *COPRA* in real networks, i.e., MIT Reality and Facebook. However, for a real network such as MIT reality and Facebook, it is hard to find the ground truth, i.e., we need to survey all the nodes to get the ground truth. Thus, in the evaluation, we only give the comparison of the algorithms in terms of modularity. As shown in Table 4, *Conductance* also outperforms *COPRA* and *Strength* in terms of modularity in real networks.

TABLE 4: Modularity of **Conductance**, **COPRA** and **Strength** in real networks

Network	Conductance	COPRA	Strength
MIT Reality	0.146562	0.140398	0.007366
Facebook	0.375441	0.106819	0.292693

C.3 I^2C Algorithms on Different Detection Algorithms

We also compare the performance of I^2C algorithms based on the community structures detected by *Conductance*, *Strength* and *COPRA* as shown in Fig. 8.

Since *Conductance* outperforms *Strength* and *COPRA* in both synthetic networks and real networks, I^2C algorithms based on more accurate community structure should perform better. As expected, for data forwarding in MIT Reality trace, I^2C performs better on the top of *Conductance* than *Strength* and *COPRA* as shown in Fig. 8a, since *Conductance* outperforms *COPRA* and *Strength* in MIT Reality as discussed above. Similarly, for worm containment in Facebook trace, I^2C also performs better on the top of *Conductance* than *Strength* and *COPRA* as shown in Fig. 8b, where $\alpha = 5\%$ and $\sigma = 75\%$ (the results are similar for other settings of α and σ, and thus omitted here).

Appendix D
Algorithms

Algorithm 1: Detection Algorithm

```
Input : $G = (V, E)$
Output: $\mathcal{C}$
1  Initialize: $\mathcal{C} = \emptyset$;
2  while $E \neq \emptyset$ do
3      $C = \{u, v\}$, where $(u, v) = \arg\max_{(u,v)\in E} w_{uv}$;
4      while $N_C \neq \emptyset$ do
5          $C' = C \cup \arg\max_{w\in N_C} B(w, C)$;
6          if $\Phi(C') < \Phi(C)$ then
7              $C = C'$;
8          else
9              break;
10         end
11     end
12     $E = E \backslash E_C$;
13     $\mathcal{C} = \mathcal{C} \cup C$;
14 end
```

Algorithm 2: Forwarding Algorithm

```
Input : $u, v, d \in V$
/* node $u$ has the message, $v$ is the encountered
node, $d$ is the destination                              */
1  if $\mathcal{C}_u \cap \mathcal{C}_d \neq \emptyset$ then
2      if $\mathcal{C}_u \cap \mathcal{C}_v \cap \mathcal{C}_d \neq \emptyset$ then
3          foreach $C \in \mathcal{C}_u \cap \mathcal{C}_v$ do
4              if $\varphi_u(C) < \varphi_v(C)$ then
5                  $v$.AddMessage();
6                  break;
7              end
8          end
9      end
10 else
11     if $\mathcal{C}_v \cap \mathcal{C}_d = \emptyset$ then
12         foreach $C_i \in \mathcal{C}_u$ do
13             foreach $C_j \in \mathcal{C}_d$ do
14                 if $(B(u, C_j) = 0$ && $B(v, C_j) = 0)$ ||
                   $(B(u, C_j) \neq 0$ && $B(v, C_j) \neq 0)$ then
15                     if $\phi_u(C_i, C_j) < \phi_v(C_i, C_j)$ then
16                         $v$.AddMessage();
17                         break;
18                     end
19                 else if $(B(u, C_j) = 0$ && $B(v, C_j) \neq 0)$ then
20                     $v$.AddMessage();
21                     break;
22                 end
23             end
24         end
25     else
26         $v$.AddMessage();
27     end
28 end
```

Algorithm 3: Algorithm for Computing the Patching Score

```
Input : $G$ and $\mathcal{C}$
1  Initialize: $\phi'_u = \varphi'_u = \psi_u = 0, \forall u \in V$;
2  foreach $u \in V$ do
3      $\varphi'_u = \max_{C\in\mathcal{C}_u} \frac{\varphi_u(C)}{\binom{|C|}{2}}$;
4  end
5  foreach $C_i \in \mathcal{C}$ do
6      foreach $C_j \in \mathcal{C}$, $C_i \neq C_j$ do
7          foreach $u \in C_i$ do
8              if $B(u, C_j) \neq 0$ then
9                  if $\phi'_u < \frac{\phi_u(C_i,C_j)}{(|C_i|-|C_i\cap C_j|)(|C_j|-|C_i\cap C_j|)}$ then
10                     $\phi'_u = \frac{\phi_u(C_i,C_j)}{(|C_i|-|C_i\cap C_j|)(|C_j|-|C_i\cap C_j|)}$;
11                 end
12             end
13         end
14     end
15 end
16 foreach $u \in V$ do
17     $\psi_u = \beta\varphi'_u + (1-\beta)\phi'_u$;
18 end
```