Community Detection in Weighted Networks: Algorithms and Applications

Zongqing Lu*, Yonggang Wen* and Guohong Cao†
*Nanyang Technological University
{luzo0002, ygwen}@ntu.edu.sg
†The Pennsylvania State University
gcao@cse.psu.edu

***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 weighted such as social networks, DTN or OSN, in this paper, we address the problems of community detection in weighted networks and exploit community for data forwarding in DTN and worm containment in OSN. We propose a novel community detection algorithm, and then introduce two metrics called intra-centrality and inter-centrality, to characterize nodes in communities. Based on these metrics, we propose an efficient data forwarding algorithm for DTN and an efficient worm containment strategy for OSN. Extensive trace-driven simulation results show that the data forwarding algorithm and the worm containment strategy significantly outperform existing works.**

I. 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 [1]. Community has been well studied in biology, sociology, psychology, business, etc. Recently, it has been exploited for designing network protocols such as data forwarding in Delay Tolerant Networks (DTN) and worm containment in Online Social Networks (OSN).

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 community [2][3][4][5][6][7] outperform traditional approaches [8][9].

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 [10][6]. In [10], the bridge nodes between communities are first patched so that they can be isolated to contain the worm propagation. In [6], 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 the community becomes an important issue. Community detection has attracted lots of attention. However, most of the existing detection algorithms focus on binary networks including [11][12][13], since many networks are naturally binary, such as biological networks where the edge between two nodes either exists or not. Among these detection algorithms, *CFinder* [11] and *RAK* [12] 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. *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 have been formulated as binary networks in many existing works. For example, best friends are treated the same as normal friends in OSN. Multiple contacts or single contact are both counted as having a contact 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 ten times in the past.

Recently, there is some research on community detection in weighted networks, e.g., *COPRA* [14] and *Strength*[15]. *COPRA* is based on *RAK* [12], and hence, similar to *RAK*, it does not converge to a constant state during label propagation most of time and the detected community is not deterministic. *Strength* [15] 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.

In this paper, we address the problems of community detection in weighted networks and exploit community for data forwarding in DTN and worm containment in OSN. Our

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

detailed contributions are as follows:

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

The rest of this paper is organized as follows. Section II presents our community detection algorithm for weighted networks. Then, we introduce the concepts of intra-centrality and inter-centrality in Section III. In Section IV, we present our forwarding algorithm for DTN, and worm containment strategy for OSN. Section V evaluates the performance of our proposed algorithms, and Section VI concludes the paper.

II. Community Detection in Weighted Networks

A. 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 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, $B(u, C) = 1$.

B. Conductance Function

In this paper, we use conductance to identify the community that has better internal connectivity than external connectivity. Conductance is a natural and widely-adopted notion of community goodness [16] 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

$$\Phi(C) = \frac{cut(C, G \backslash C)}{w_C}, \tag{2}$$

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 as shown in Fig. 1, $cut(C, G\backslash C) = 4$ and $w_C = 15$. With lower conductance, more edge weights are within the community and the identified community is better. However, it is generally NP-hard to optimize the conductance in community detection [16]. Thus, we propose a heuristic based algorithm in the next subsection .

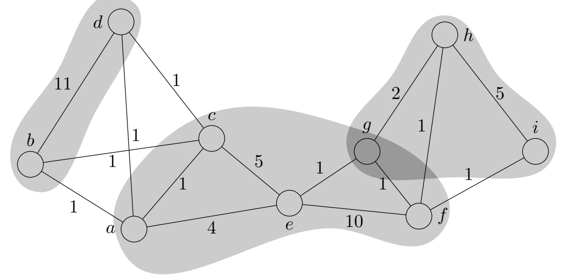


Fig. 1: Community detection in weighted networks.

C. Detection Algorithm

Different from algorithms designed for binary networks, the edge weight should be taken into consideration in weighted networks. If the edge weight between two nodes is high enough, the two nodes should be in the same community.

Our detection algorithm works as follow. For a given network G, we first choose those nodes that are connected by the highest weight edges as a temporary community C and calculate its conductance of $\Phi(C)$. Then, there is an expanding process, which finds all nodes that are adjacent to C (denoted as N_C). 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 Algorithm 1.

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
```

Unlike binary network where a threshold is used to determine a community such as in [6], for weighted networks we cannot fix the threshold of conductance due to the heterogeneous weight distribution. Instead, we use the conductance of temporary 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 the temporary 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 Eq.2, if $\Phi(C) > \Phi(C')$, we have:

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

When $B(u, C) > \frac{1-\Phi(C)}{2-\Phi(C)}$, node $u \in N_C$ will be added into community C. When $B(u, C) > \frac{1}{2}$, whatever $\Phi(C)$ is, node u is eligible to be included in community C. This is reasonable in practice since node u should belong to C if more than half of degree of node u is connected with 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. The overlapping part may be very large (more than 50%) in some cases, thus they need to be merged.

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 shaded region.

In our 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 is $|V|^2$.

III. 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,

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

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

Note that the shortest path cannot go beyond a community, which means 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. 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.

Definition 2. ***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,

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

where the overlapped nodes between two communities are excluded ($v, w \notin C_i \cap C_j$). Each node will have 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 to isolate these two communities.

IV. Community based Applications

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

A. Data Forwarding in Delay Tolerant Networks

Recent work (e.g., *BubbleRap* [3] and *AFOCS* [6]) has shown that community based data forwarding algorithms can significantly reduce the number of data replication while maintaining similar data delivery ratio and data delivery time in DTN. However, as the community detection in these algorithms is based on binary network, they also have some weaknesses. For example, *BubbleRap* may use inaccurate centrality and most traffic between two communities in *AFOCS* may not go through overlapped nodes. Different from them, our solution is based on our community detection algorithm designed for weighted networks. Based on intra-centrality and inter-centrality, we design a new forwarding algorithm.

Our data forwarding algorithm works as follow. Suppose a node (sender) has a message destined for another node (receiver). If they are in the same community, the sender only forwards the packet to the node encountered (relay) that has higher intra-centrality. If they are in different communities, the sender forwards the packet to the relay which satisfies the following: (1) the relay is in the same community of the receiver; or (2) the relay has higher inter-centrality than the sender when the belonging degrees of the sender and relay to receiver's community are both zero or non-zero; or (3) when the relay has non-zero belonging degree to the receiver's community, and the belonging degree of the sender is zero.

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) are more probably and more quickly to forward the message to the community(s) of destination.
- The nodes that are more popular within one 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.

B. Worm Containment in Online Social Networks

Recently, social network based worm patching schemes for cellular network and OSN proposed have been proposed in [10] and [6], respectively. The intuition behind these schemes is to contain worms within infected community before they spread out. In [10], separators (key nodes that separate network partitions) are patched. Similarly, the neighbors of overlapped nodes between two communities are patched in [6]. Both strategies choose nodes located on the boundary of communities to be patched first. However, in dense networks, every node can be boundary node, thus the patch of boundary nodes is not efficient and effective. Moreover, the selected nodes in both [10] and [6] cannot effectively block the traffic between communities, since the nodes do not have much influence on the connections between communities according to their definition.

Our worm containment strategy not only consider how to isolate communities, but also consider how to slow down the worm spread within a community. In most 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 characteristics of slow start and exponential propagation exhibited by worms, by slowing down the worm propagation we will have more opportunities to contain the worms within infected communities and prevent the worms from spreading to other communities. The intuitions behind our strategy is as following:

- Patching nodes with high intra-centrality will slow down the worm propagation within community.
- Patching nodes with high inter-centrality will slow down the worm propagation between community.

When a node receives the patch, it will be immune to the worm and can be used to redistribute the patch further. However, distributing patches to all nodes at the same time may not be feasible due to the bandwidth limitation in cellular networks, or the limited availability of vaccines. Thus, we have to determine the proper patching order for these nodes based on patch score. The higher the patch score is, the sooner the node will be patched. The patch score is calculated by the combination of normalized intra-centrality (normalized by the number of pairs of nodes within community) and normalized inter-centrality (normalized by the number of pairs of nodes between communities). As a node may have multiple intra-centrality and inter-centrality values, only the largest values are used to calculate patch scores.

By considering intra-centrality and inter-centrality together, our worm containment strategy can slow down the worm spread within community and between communities.

V. Performance Evaluations

In this section, we evaluate the performance of the data forwarding algorithm and the worm containment strategy, and compare them to existing work.

A. Data Forwarding Algorithm

To evaluate the performance of our data forwarding algorithm in DTN, experiments are conducted based on the MIT Reality trace [17], 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 I.

TABLE I: MIT Reality trace summary

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

In our experiment, each node sends 500 messages to other randomly selected nodes. Messages will be discarded if they are not successfully delivered within the 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* [8], 2) *BubbleRap* [3] which uses *k-clique* [11] as the community detection algorithm, 3) *AFOCS* [6], and 4) *Baseline* where the source keeps the message until it encounters the destination.

We evaluate these algorithms on two message forwarding modes: forwarding with message duplication and forwarding without message duplication. For message duplication, the algorithms are compared in term of data delivery ratio, data delivery time, and message replica. For the mode without message duplication, data delivery ratio and data delivery time are considered. As message duplication is seen as data forwarding cost, delivery ratio achieved in the mode without message

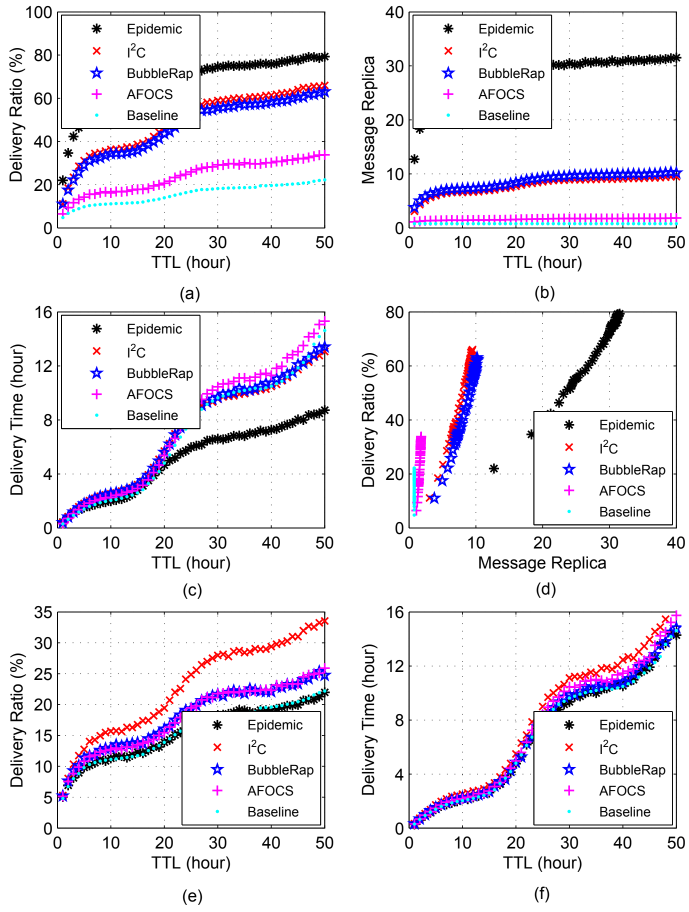


Fig. 2: Performances of data forwarding algorithms – **Epidemic**, **I^{2}C**, **BubbleRap**, **AFOCS** and **Baseline** in terms of data delivery ratio, data delivery time, and message overhead based on the MIT reality trace.

duplication is the efficiency of data forwarding algorithms. Experiments are repeated and the results are averaged for consistency.

Fig. 2 shows the results of data forwarding on the MIT reality trace, where TTL varies from 1 to 50 hours. In *Epidemic* the message is always forwarded to the node encountered, thus it has the highest delivery ratio and forwarding cost (message replicas) as shown in Fig. 2a, 2b. 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 destination.

Fig. 2a shows that the delivery ratio of I^2C is higher than that of *BubbleRap*, and *BubbleRap* incurs much more message replicas than I^2C, up to 40%, as shown in Fig. 2b. *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. 2c. Fig. 2d gives the frontier of each algorithm in terms of delivery ratio and message replicas. Although I^2C, *BubbleRap* and *AFOCS* sit in the similar region, I^2C always performs better than the other two algorithms.

Fig. 2e and Fig. 2f 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 relays for message forwarding. I^2C is the best and its performance is up to 50% better than other algorithms, although the delivery time of I^2C is only slightly more than others.

To summarize, there exists a tradeoff between data 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. 2d, where the frontiers of all other algorithms sit in the regions between *Epidemic* and *Baseline*. As demonstrated in Fig. 2, I^2C achieves a good balance between performance and cost, and is the most efficient algorithm. By selecting different criterion for data forwarding within community and between communities, respectively, I^2C outperforms *BubbleRap* and *AFOCS*.

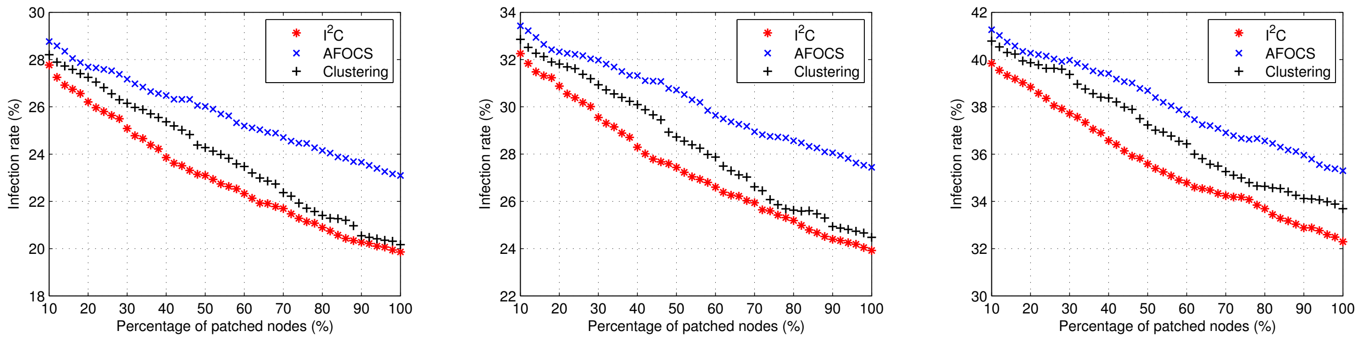
B. Worm Containment Strategy

To evaluate the worm containment strategies, we use the Facebook trace from [18]. It contains friendship information and wall posts among the Facebook user in the New Orleans regional network for more than four years. We choose a partial trace which spans half year. The chosen trace is summarized in Table II, where the contact between two nodes is the wall post and edge weight is the contact frequency. Without losing generality, we use similar worm propagation model in [10][6] that mimics the behaviors of the famous worm *Koobface* that once spread out on Facebook. We assume that the worms are able to explore the friendship information for propagation (such as sending out messages including malicious links). The probability of node activating the worm received from his friend is proportional to the contact frequency between them. The time taken for the worm to propagate from one user to his friend is inversely proportional to contact frequency between them. Finally, the worm starts to propagate right after it successfully infects the user.

TABLE II: Facebook trace summary

Trace	Facebook
No. of nodes	12123
No. of edges	31932
Average node degree	21.4
No. of contacts	129462
Duration (days)	180

At the very beginning, we randomly choose 0.05% of nodes as the seed set of worm sources to initiate the infection process. When the infection rate (the fraction of infected nodes over all nodes) reaches the predefined alarm threshold α,



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

Fig. 3: Performance of worm containment algorithms – $\mathbf{I^2C}$, **AFOCS** and **Clustering** in terms of infected rate on Facebook trace with alarm threshold α =5%, 10% and 20%.

the patching process will be initiated. The experiments are conducted with α=5%, 10% and 20%, respectively.

We compare I^2C based worm containment strategy with *AFOCS* [6] and cluster based scheme [10] shown as *Clustering*. Unlike I^2C and *Clustering*, where the sequence of nodes to be patched is determined, *AFOCS* chooses a particular part of nodes to be patched. The number of patched nodes selected by *AFOCS* in Facebook trace is 985. So, in order to compare the performance with different schemes, we choose the same number of nodes selected by *AFOCS* for I^2C and *Clustering* according to patch score and priority, respectively. The worm propagation is simulated for 30 days after the alarm threshold is reached.

Fig. 3 shows the results of infected rate achieved by different algorithms for alarm threshold α=5%, 10% and 20%, respectively. For α=5% shown in Fig. 3a, where the patching process is initialized at very early stage, the infected rates are still relatively low after 30 days worm propagation. From Fig. 3, we can see that patching late will result in higher infected rate. Among these algorithms, our worm containment scheme has the lowest infected rates, which demonstrates that patching nodes with high intra-centrality and inter-centrality 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 infected rates.

VI. Conclusions

In the paper, we proposed a conductance-based community detection algorithm for weighted networks and designed a data forwarding algorithm for DTN and a worm containment strategy for OSN based on two metrics – intra-centrality and inter-centrality. The experiments on real DTN traces show that our forwarding algorithm outperforms other community-based algorithms in terms of data delivery ratio and data forwarding cost. The experiments on real OSN traces show that our worm containment strategy has lower infection rate than other algorithms.

References

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

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

[3] P. Hui, J. Crowcroft, and E. Yoneki, “Bubble rap: social-based forwarding in delay tolerant networks,” in *ACM Mobihoc 2008*.

[4] W. Gao, Q. Li, B. Zhao, and G. Cao, “Multicasting in delay tolerant networks: a social network perspective,” in *ACM Mobihoc 2009*.

[5] W. Gao and G. Cao, “User-centric data dissemination in disruption tolerant networks,” in *IEEE INFOCOM 2011*.

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

[7] 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, jan. 2013.

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

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

[10] 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*.

[11] 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.

[12] 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.

[13] Y. Ahn, J. Bagrow, and S. Lehmann, “Link communities reveal multiscale complexity in networks,” *Nature*, vol. 466, no. 7307, pp. 761–764, 2010.

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

[15] 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.

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

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

[18] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, “On the evolution of user interaction in facebook,” in *Proceedings of the 2nd ACM SIGCOMM Workshop on Social Networks (WOSN’09)*, August 2009.