Detecting Temporal Social Communities for Content Dissemination in Indoor Environments

BOWEN Yu[1], YU Zhang[2] and YUWEI Xu[3,*]

College of Computer and Control Engineering, Nankai University, China

[1]bowenyu@mail.nankai.edu.cn

[2]{zhangyu1981, xuyw}@nankai.edu.cn

Keywords: mobile social computing, mobile device, community detection.

Abstract. Mobile social computing services such as location-based services and cooperative computing services are emerging for mobile devices. Those services promise to give us more opportunities to enhance social connectivity with neighbors and nearby places. This paper focuses on the issue of how to be aware of neighbors and social events in indoor environments. We propose a dynamic social community detection algorithm by considering not only the relationship between users but also that between user and place. Simulation results indicate that our community detection algorithm outperforms the others in modularity and stability.

Introduction

Recent years have witnessed a significant rise in the development of mobile social computing services in academia and industry [1]. These services want to make use of inter-personal affinities, mobile resources and real-time location to help people connect with neighbors and nearby surroundings toward further collaboration and communication. While services vary, two essential issues has arisen. The first issue is to make mobile device holders aware of neighboring device holders with common interests. Second, due to different service features such as publisher-subscriber pattern [2] or cooperative computing [3] as well as different mobility patterns [4], content dissemination strategy remains largely unexplored.

Previous researches [5,6] mention that human movement is made up by a string of indoor environments such as cafes and shopping malls, and many users are willing to enjoy mobile services in them. Meanwhile, there comes out many location-aware mobile social applications such as Loopt and Foursquare. Therefore, it is very meaningful and practical to concentrate on those two essential issues in indoor environments.

Currently, many social applications can utilize GPS to identify or recommend friends. In general, the community partition algorithm in indoor environments has three missions. a) to merge the users with closer social relationship into one community, b) to keep all communities relatively stable, and c) to dynamically support users moving in and out of communities. To achieve the above three missions, in this paper we propose a two-stage detection algorithm. The Heavy-Edge and Average-Vertex Partition stage (HEAVPS) is to achieve the community stability and merge correlated users, and the Community Match Stage (CMS) is responsible for dynamically updating communities.

Dynamic Social Community Detection

An undirected weight graph $G = (V, E)$ is used to represent the mobile social network in an indoor environment. The vertex set V represents the current users in the environment. We measure the user-place relationship by using visit frequency, residence time and the number of mobile services use in that place. Each vertex is annotated with 3 normalized cost weights (W_f, W_t and W_n). In order to facilitate the computation, we use a composite vertex weight $W(v) = \alpha W_f + \beta W_t + \gamma W_d$ to represent them, where α, β and γ can be dynamically adjusted to different mobile services. The edge set E of the graph stands for user-user relationship such as common interest and similar background. The weight

of an edge as the strength of the relation-$W(e(u, v))$ can be measured by the Jaccard similarity. Based on this social model, our community detection algorithm has three targets. (1) try to merge the nodes with closer relationship in one community, (2) keep all communities relatively stable, and (3) support variation in the number of people due to their entering or leaving. We will introduce two main stages of our algorithm to achieve the above three goals.

The problem of social community partition is similar to that of partitioning an undirected graph into a certain number of subsets while fulfilling some given goals, which is known as NP-complete. The HEAVPS tries to find an optimal solution to obtain k disjoint communities V_1, V_2 …V_k which satisfies: (1) Each user should be absorbed into one and only one community; (2) The sum weight of the edge-cut should be minimized to achieve closer users in one community; (3) The difference of community weight should be minimized to make all communities stable.

To implement above requirements, we define edge merging metric (EM) and vertex merging metric (VM) as follows:

$$EM(V_i, V_j) = \frac{\sum_{u \in V_i \cap v \in V_j} W(e(u,v)) + \sum W(e(V_i)) + \sum W(e(V_j))}{(V_i + V_j)(V_i + V_j - 1)/2}, \tag{1}$$

$$VM(V_i, V_j) = \left| \frac{\sum_{u \in V_i} W(u) + \sum_{v \in V_j} W(u)}{V_i + V_j} - \frac{\sum_{q \in V} W(q)}{V} \right|. \tag{2}$$

Since the EM represents the average edge weight in the complete graph, the higher EM implies the closer relation between these communities. Meanwhile, the VM is the distance between the average vertex weight of merged community and that of whole vertexes. Apparently, if all communities share equal average vertex weight, they can be equally stable and that equal value must be the average vertex weight of whole vertexes(i.e. $V_1/n_1 = V_2/n_2 = \ldots = V_k/n_k = (V_1 + V_2 + \ldots + V_k)/(n_1 + n_2 + \ldots + n_k)$). Therefore, merging those communities with the lowest VM can benefit the overall stability. Taking account of both metrics, we further define a composite merging metric which embodies the heavy-edge and average-vertex:

$$CM(V_i, V_j) = \lambda_1 \times EM(V_i, V_j) + \lambda_2 / VM(V_i, V_j), \tag{3}$$

where λ_1 and λ_2 are important factors. If one of them is assigned to zero, the metric will be regressed to EM or VM. So, in this paper we assign them both to 1. In theory, there can be more than one community in relation to V_i sharing the same maximum CM. So we propose the following approach to select the most suitable one:

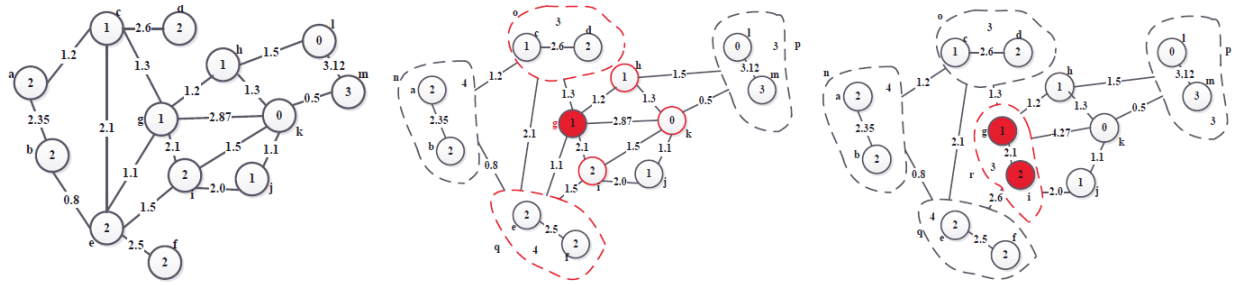
$$MS(V_i, V_j) = \frac{\sum_{u \in V_j \cap v \in V_k} W(e(u,v))}{e(V_j).size}, e(V_i) \cap e(V_k) \neq \varnothing. \tag{4}$$

We will choose the community with the highest MS since it is tightly link with V_i and neighbors of V_i. If there is still more than one, we will randomly select a community from them.

The HEAVPS is an agglomerative procedure. Initially, each vertex in the graph can be regarded as a community. For any community, we find out which neighbor is the most suitable one to merge with according to the CM and MS.

We utilize a simple case to illustrate the “Selecting Neighbor” in Fig. 1. Fig. 1(a) presents the original social network. After having finished some combination (e.g. node *a* and node *b* are aggregated to a big node *n* whose vertex value is the sum of *a* and *b*), we focus on which node is suitable for combining with node *g* whose neighbors have been depicted in figure 1(b). In order to

facilitate computing and avoid the "divide zero error" in real programming, we make a trick that the equation is transformed to $VM(V_i,V_j)=(\sum W(u)+\sum W(v)-(V_i+V_j)\times avg)^2+1$. Next, we will use this equation to calculate the CM for node *g*. We find that the community *i* and *k* share the same maximum CM. Therefore, we further calculate the MS. Finally, we will choose community *i* to combine with community *g* as shown in figure 1(c) since they can achieve the best CM and MS.



(a) Original Graph (b) Social Neighbors(Node g) (c) Combination Result(Node g)

Fig. 1. Case of Community Detection

In order to achieve the best overall performance, we use the Newman modularity to measure the appropriateness of communities and the standard deviation (called Similarity) to evaluate the stability of communities. In practice, when the modularity is approaching the peak, it will increase slowly while the similarity still fluctuates sharply. Therefore, the HEAVPS takes modularity as the criterion in the initial phase, and then uses the similarity when modularity has little change.

Since everyone is free to enter or leave a location, the social graph in indoor environments may change at any time. It is impractical to repartition the changing social graph. So we set up the CMS to speed up partition. In CMS, first we define a Community Header on behalf of each community. The Header should be the one that has the tightest relation with its community and stays long in this location. For simplicity, this paper only considers the latter and thus regards the node with highest vertex weight in one community as the header. Then, when a person enters a location and submits his information to the partition server, the server will only measure the relation between him and all community headers by using the Jaccard similarity. Finally, that person will be included in the community whose header has the closest relation with him. Before leaving the location, he will inform the server and reclaim updated individual information such as visit frequency and residence time from the server.

Performance Evaluation

We leverage four widely used real data sets: Zachary's karate club (34 nodes), Les Miserables (77 nodes), American College football (115 nodes) and Neural network (297 nodes) from Newman's website [7] to estimate our social partition algorithm. We add the vertex weight ourselves because these data sets do not contain it. Since we find that the relation between people and place is similar to that between user and online social websites (User Ranking is the aggregation of visit frequency, residence time and the rate of communications with others), we obtained 84,313 samples from RenRen and find that nearly 55% user ranking is lower than a quarter of average. 24% user ranking is between a quarter of average and average. 16% is between average and four times the average while less than 5% is above average. Therefore we utilize these probabilities to generate the vertex weight.

The figure 2 show the community modularity (left) and the similarity (right) of different data sets. We can see that, with the increase of node size, the performance of modularity in Fast-CNM will gradually catch up with others while the similarity is the highest still. This is because the Fast-CNM uses the greedy propagation method which will suffer from monster community problem. It means a few communities take up too many nodes, and the smaller a graph size is the more serious the problem is. Meanwhile, that problem also hurts the overall community stability. Shortest Path is good at handling small graph while it is difficult to identify the right bridges in complicated graph. Therefore as the graph size grows, the partition algorithm performance degrades. Both modularity

and similarity in our algorithm are better than the others since we not only consider the tightest edge but also more average vertex. Therefore, our proposed partition algorithm is more suitable to identify the social communities in indoor environment.

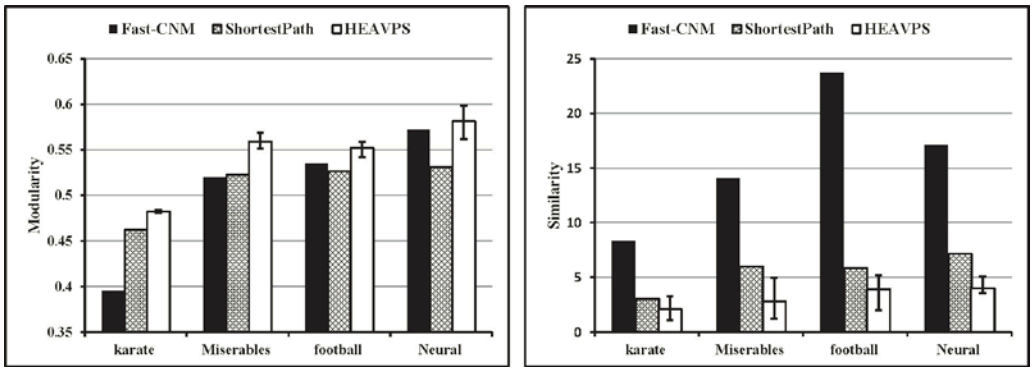


Fig. 2. The Modularity and Similarity of communities under different partition algorithms

Conclusion

In this paper, we introduce a novel community partition algorithm by considering both people-people and people-place relationships to address the issue of common interest identification for mobile social services. It not only achieves high modularity and stability but also robustly supports a dynamic social network in indoor environments.

Acknowledgements

This work is partially supported by the Tianjin Municipal Science and Technology Commission under Grant (No. 13ZCZDGX01098), the Natural Science Foundation of Tianjin (No. 16JCQNJC00700) and the Science Foundation for Young Scholars of Nankai University (No. ZLA2035946).

References

[1] N. Kayastha, D. Niyato, P. Wang. "Applications, architectures, and protocol design issues for mobile social networks: A survey." Proceedings of the IEEE, (2011).

[2] P. Costa, C. Mascolo, M. Musolesi. "Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks." J-SAC, (2008).

[3] D. G. Murray, E. Yoneki, J Crowcroft. "The case for crowd computing." Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds. ACM, 2010.

[4] D. Karamshuk, C. Boldrini, M. Conti. "Human mobility models for opportunistic networks." Communications Magazine, IEEE, (2011).

[5] J. Fan, J. Chen, et al. "Delque: A socially aware delegation query scheme in delay-tolerant networks." Transactions on Vehicular Technology, IEEE, (2011).

[6] J. Wu, M. Xiao and L. Huang. "Homing spread: Community home-based multi-copy routing in mobile social networks." INFOCOM, 2013 Proceedings IEEE. IEEE, 2013.

[7] A. Keränen, T. Kärkkäinen and J. Ott. "Simulating Mobility and DTNs with the ONE." Journal of Communications, (2010).