Detect User Transition Areas on Community Structure with Noise in 5G Mobile Environment

Mintao Liu, Tunzi Tan, Kangyu Shang, Suixiang Gao, Wenguo Yang and Yang Yang

ABSTRACT

Detecting communities problems have been studied under various background and networks in last two decades. This paper poses a new problem, named community detection with noise, to find communities where nodes connect densely to each other under the background of user movement in 5G. Because the prospective small cells in 5G would make data lengthy [1] [2], this paper aims to figure out meaningful areas without seldom visited cells named as “noise”. Two kinds of methods are proposed. One is based on the normal community detection problem. Noise nodes are picked and deleted by proper quantities. The other is a heuristic algorithm that starts with the most important node, adding closely connected nodes into the community. Compared with classic GN algorithm, these two methods outperform in quality metrics that focus on internal density. And a practical example that gives user movement areas is showed.

INTRODUCTION

A mobile terminal gets served by connecting to the base station which offers the roughly strongest signals. Usually, the nearest base station provides the best signals. As user moves, the mobile terminal's connection switches among base stations to maintain the quality of service (QoS). Thus, the current location of a user can be represented by the location of the connected base station and a sequence of the historical connected base stations denotes the trajectory of the user in terms of mobility.

Through such mobile data, a lot of studies on people's mobility are done. Specially, more attention is paid to location prediction because of its practical usefulness in wireless networks, mobile computing and so on. Many have studied methods in Markov chain, pattern detection and neural networks while most of them focus on the next location or base station of a user.

However, predicting only the next base station meets some challenges. In conventional cellular networks, a handover between base stations does not always
relate to a move of the user, which is well known as ping-pong effect. It is a phenomenon that user stays in the intersection of several stations while the connection switches back and forth. Also, the meaning and importance of the stations to the user are neglected if only the next move is predicted. Furthermore, it is commonly assumed that the fifth generation (5G) mobile network will deploy small cells due to the move to mm-wave frequencies [1] [2]. From this trend, we can safely assume that cell granularity level will be much smaller, which makes user transition data lengthy and jumbled.

Considering the challenges mentioned above, we propose a problem named community detection with noise to figure out the areas that mean a lot to user as a foundation of location prediction and some mobile services. By transforming the categorical mobile station handover sequences into an undirected graph with weights, two kinds of algorithms are designed to find meaningful station clusters of a user or a group of movement-similar people. To our knowledge, we are the first to apply community structure methods in mobile management.

Studies in community structure have developed from early graph partition, whose goal is to minimize the connections between separate components. The classic representatives are Kernighan-Lin algorithm as local [3] and spectral partition based on Laplace matrix as global [4]. The community structure problem often discussed nowadays is proposed by Girvan and Newman in 2002 [5]. Compared to graph partition, they treat community structure as inner nature of complex networks. They use "betweenness centrality" to find edges in boundary and introduce "modularity" to evaluate the quality of the partition of the community structure [6]. And over the past few decades, many methods have been developed [7] while we cannot use them directly.

Almost all the community structure methods give a partition of the graph and every node is included into one community or more (known as overlapped communities). In the transition graph of a mobile user, however, it is not necessary to put every node/station into a community which means an area frequently visited by the user here. It is studied that people's mainly movement patterns are simple although they may have a lot of historical data [8]. Apparently, most people live and work in some certain areas and occasionally leave for a travel. In this case, we only interested in the areas that the user appears frequently and stays for a long time. So, the problem named community detection with noise is proposed, meaning some nodes can be treated as noise (such as the travel places) and is not included in any community.

**METHODS AND METRICS**

Assume each time a user's terminal handovers from a station to another station \( s_i \) at the time \( t_i \), a pair \((s_i, t_i)\) is recorded. The historical trajectory of a user is denoted by a sequence
For each sequence $S$, a corresponding transition graph $G(S)$ can be defined as follows:

**Definition 1.** A transition graph $G(V,E)$ generates from a pair sequence $S$. Vertices is the set of the stations, with stay time as node weights,

$$\omega^n(s) = \sum\{t_{i+1} - t_i | s_i = s, 1 \leq i \leq n\}$$

There is an edge between two nodes for each transition

$$E(s) = \{\{s_i, s_{i+1}\} | 1 \leq i \leq n\},$$

and the weight of an edge is the number of handovers between the two stations

$$\omega^e(\{s, s'\}) = |\{i | \{s_i, s_{i+1}\} = \{s, s'\}, 1 \leq i \leq n\}|.$$ 

Notice the graph is undirected, which means handover direction is not considered here. Both nodes and edges are weighted indicating the importance of the places and the close connections between the nodes.

**Definition 2.** The transition matrix $T$ of a transition graph $G$ has entry $T_{ij} = \omega^e(\{i, j\})$.

Suppose graph $G(V,E)$ has $N$ vertices and $K$ edges. For unweighted graph, the adjacent matrix $A$ equals to transition matrix $G$, where entry $a_{ij} = 1$ if there is an edge between node $i$ and node $j$, and $a_{ij} = 0$ otherwise. In weighted graph, besides $a_{ij}$, weights of edges $\omega_{ij}^e$ in transition matrix is also considered. As mentioned in Introduction part, community structures are detected to play the part as meaningful areas to the users. Assume cluster $S$ is such a community set and there are $n_S = |S|$ nodes and $m_S = |\{(u,v) : u \in S, v \notin S\}|$ edges in the cluster. $c_S = |\{(u,v) : u \in S, v \notin S\}|$ denotes the edges connect $S$ and other communities.

With transition graph above, community detection problem is discussed here to find some meaningful areas. For example, a user may usually stay at home, school or office. In 5G, there may be more than 50 small cells that the user has visited in school. But only 5 cells, like office or dorm, are visited frequently and meaningful to user. This paper aims to find such important areas. The cells representing such area tend to transfer to each other frequently due to user's movement. So the community structure problem is introduced. The difference is that not every cell/station is supposed to be assigned to a community, for example, a distant restaurant that has been visited only once is not important and should be treated as a noise node in the community detection. To solve the community detection with noise problem, two kinds of algorithms are designed as below.
Reformed Community Detection Algorithms

To begin with, picking out the noise nodes from the detected community is intuitive and simple. That is using any well-developed community detection algorithms to find the communities at first, then in each community the noise nodes are deleted according to a structure quantity until criteria are not satisfied.

The quantities used to identify noise nodes should reflect the connections between nodes, whether the nodes are linked closely in a cluster. In [9], similar quantities are summarized as measures of segregation. There are two main kinds of measurements that can be used to find out the noise nodes. We do not give a definition to noise here because of the complexity in graphs and requirements. But we can sort the nodes according to the degree of being noise by calculating the quantities below.

The first to discuss is clustering coefficient. This concept reflects how the neighbors of a node connect to each other, which comes from the social networks that friends of a person also be friends. In complex network, clustering coefficient is first defined by [10]. For an individual node $i$, its clustering coefficient is

$$ C_i = \frac{E(i)}{k_i(k_i - 1)/2}, $$

where $k_i$ is the degree of node $i$ and $E(i)$ denotes the number of edges that connect the neighbours of the node. The denominator gives maximum possible number of such edges. For undirected graph with weighted edges, several generalized clustering coefficients are defined and [11] gives a comparison and analysis of four generalizations.

The other quantity can be used is efficiency. This concept is introduced by [12], giving a physical explanation to the concept of small-world. Here the definition on vertex is showed rather than on whole graph:

$$ E_i = \sum_{j \neq h} a_{ij}a_{ih}[d_{jh}(N_i)]^{-1}, $$

where $N_i$ is the set of neighbors of $i$, $d_{jh}(N_i)$ is the length of the shortest path between $j$ and $h$ inside the $N_i$-induced subgraph.

To generalize this concept under the application scene, we propose 2-level efficiency. $N_i$ is replaced by $N_i^{(2)}$, the set of the first and second neighbors of $i$ considering the variety of paths in actual. In weighted graph, 2-level efficiency is generalized as:
\[
E_i^{(2)} = \frac{1}{k_i(k_i - 1)} \sum_{j \neq h} (\hat{\omega}_{ij}^e \hat{\omega}_{ih}^e \frac{\sum_{P_{jh}} \hat{\omega}_e}{d(P_{jh})^2})^{\frac{1}{3}},
\]

where \(\hat{\omega}_{ij}^e = \omega_{ij}^e / \max(\omega^e)\). \(P_{jh}\) denotes the path between \(j\) and \(h\) in \(N_i^{(2)}\)-induced subgraph. For all the edges in path \(P_{jh}\), \(\sum_{P_{jh}} \hat{\omega}_e\) means the sum of their weights and \(d(P_{jh})\) is their number. In the last fraction, the average weights of the path correspond to 1 in the original definition and \(d(P_{jh})\) to \(d_{jh}(N_i)\).

The value of noise nodes in above quantities is smaller compared to that of the nodes connected to each other closely. For a node \(i\), the quantities are evaluating how the neighbors of \(i\) links in different views. With its neighbors clustered, the node itself is also a part of the community. Using corresponding measurements in unweighted or weighted graphs, nodes could be picked in the order of “noise”.

Based on the quantities above, community detection algorithm can be reformed as follows, taking GN algorithm as an example:

**Algorithm 1 Reformed GN Algorithm**

Input: Transition Matrix \(T\), Node Weight \(W\)  
Output: Group Partition \(C\)  
1: Change nonzero value in \(T\) into 1, denoted as \(T_{adj}\)  
2: Calculate edge betweenness \(Be\) in \(T_{adj}\)  
3: for \(i = 1:\) length(\(Be\)) do  
4: \(Be(i) = Be(i) / \omega^e\)  
5: end for  
6: Delete the edge with smallest value in \(Be\)  
7: if Criteria 1 meet then  
8: Return to step 2  
9: end if  
10: for each community found do  
11: Calculating a noise evaluation quantity \(V_n\)  
12: while Criteria 2 meet do  
13: Delete the node with smallest value in \(V_n\)  
14: end while  
15: end for

Here, the result Group Partition \(C\) is a vector. \(C_i = 0\) indicates that node \(i\) is a noise node while \(C_i = 1/2/\ldots\) means node \(i\) belongs to a community labeled 1/ 2/ others. The nodes with the same value in \(C\) are in the same community.

The algorithm showed above is based on the classic GN algorithm. The step 3 to 4 is adapting edge betweenness in weighted graph: that is dividing each betweenness by the weight of the corresponding edge [13]. Then the GN algorithm stops by Criteria 1, which can be any proper conditions like modularity. The number of current communities is used in current example. From step 10, nodes is picked and
labeled as noise in each community. Criteria 2 defers in graph as below according to experiments:

- Unweighted: stop if \( \frac{m_S}{n_S} \) declines.

- Weighted: stop if \( \frac{\sum_{S} (\omega^{p}) \sum_{S} (\omega^{e})}{m_S} \) declines.

The criterion aims to hold more edges among the nodes. So in some definite number of nodes, the more the edges, the bigger the \( \frac{m_S}{n_S} \). The situation is similar in weighted graph while more average weights in both nodes and edges are welcomed.

The result is visualized by Graphviz in Figure 1 and Figure 2. Figure 1 shows three communities giving by GN algorithm, based on which noise nodes is denoted as grey by step 10 to end in Figure 2.

![Figure 1. Community Structure by GN algorithm.](image-url)
Local Growing Method

The above community detection methods are based on the classic algorithms with every nodes being partitioned at first. In some large complex networks, especially in the movement transition graph, most of the nodes have low weights and degrees, making partition and subsequent delete unnecessary. Naturally, the next heuristic algorithm is designed.

Unlike deleting nodes in reformed method, here we start with the most important node as the first node in the current community. Then find all the neighbor nodes of the community and calculate the values measuring their connections with the community. Add the neighbor node with the maximum value into the community. The iteration of finding and adding neighbor nodes continues until some conditions based on actual needs or community-measuring scores are not satisfied. The algorithm can be specified as follows:

**Algorithm 2 Local Growing Algorithm**

- **Input:** Transition Matrix $T$, Node Weight $W$
- **Output:** Group Partition $C$

1: $N_C = \emptyset$ denotes all checked nodes
2: $C = \emptyset$ means all nodes start as noise
3: while length(NC) ≠ |node| do
4:    Pick Nmax = max(W) as the first node in current community Com = [Nmax]
5:    if Degree of Nmax ≤ 2 then
6:        Break
7:    end if
8:    Find all neighbor nodes of Com, denoted as Neigh
9:    for each node i in Neigh do
10:       Calculating v(i) = sum(T(Com, i) * W(i))
11:    end for
12:    if Criteria meet then
13:       Com = Com ∪ Neigh(argmaxi(v))
14:       Back to Step 8
15:    else
16:       NC = NC ∪ Com
17:       Label Com in C
18:    end if
19: end while

The algorithm stops if all nodes have been checked or remained most important node is not important enough with a degree smaller than 3. For the iteration of adding nodes, the stopping criteria used should adapt to reality, the size of community for instance. If nature of graph is the only consideration, similar to the reformed algorithm, we present tested expressions as criteria for node-adding in unweighted and weighted graphs separately. However, an identical condition needs to be satisfied: that is adding the first few nodes unconditionally to start a community so that the following values can work:

- Unweighted: stop if \( \frac{mS}{nS} \) declines.
- Weighted: stop if \( W(\argmaxi(v)) < q_2 \) and \( \frac{\sum(e)_{nS}}{nS} \) declines.

The calculation is the same with reformed algorithm in unweighted graph while there are some changes for weighted graph. \( W(\argmaxi(v)) < q_2 \) denotes the weight of the added node should be larger than upper quartile, \( q_2 \), of node weights. \( \frac{\sum(e)_{nS}}{nS} \) continues to decline so \( \frac{\sum(e)_{nS}}{nS} \) is used instead. The result is the same as Figure 2.

ANALYSIS AND APPLICATION

Quality Metrics

To evaluate the quality of the partition, different kinds of metrics are developed. [14] gives a well-rounded summery of the scores for a single community in the
unweighted graphs. In our case, the whole communities as well as the noise set need to be taken into consideration. Also, weights in nodes and edges play an important part in the network structure. So the metrics used here need to be generalized to whole weighted graphs.

Assume graph $G$ have been partitioned into $n$ communities as $S_1, S_2, \ldots, S_n$. The metric $f(S_i)$ is the quality score of community $S_i$. And the score of whole graph partition could be denoted as

$$\text{Quality}(G) = \sum_{i=1}^{n} f(S_i) - \alpha f(\text{Noise}).$$

The score of every community is summed up and the score of noise nodes set is also considered by parameter $\alpha$. An ideal partition should have high quality communities as well as noise nodes set that is scattered.

Different from other community detection problem, the problem community detection with noise focuses on the inner connection. We have calculated all the metrics summarized by [14] in the simple example showed in Figure 1 and Figure 2. Here $\alpha = 0$ in the case with noise nodes. And it turns out that communities found with noise outperforms in the following metrics than that without noise:

- Internal density: $f(S) = 1 - \frac{m_S}{n_S(n_S-1)/2}$
- Maximum-ODF: $f(S) = \max_{u \in S} \frac{|\{(u,v): v \not\in S\}|}{d(u)}$
- Flake-ODF: $f(S) = \frac{|\{u: u \in S, |\{(u,v): v \in S\}| < d(u)/2\}|}{n_S}$

Note that the lower the value of $f(S)$, the better the quality. And the value is calculated in the Table I and Table II.

Internal density is the edge density inside the community. Maximum-ODF(Out Degree Fraction) is the maximum fraction of edges pointing outside among nodes in the community. And Flake-ODF is the fraction of nodes in $S$ that have fewer edges pointing out. The definition is based on edge numbers and node numbers in unweighted graph. It is natural to generalize these into a weighted graph as $m_S = \sum (\omega^e(u,v): u \in S, v \in S)$, $c_S = \sum (\omega^e(u,v): u \in S, v \not\in S)$, where edge weights are added instead of counting the edge numbers.

Asides from the metrics given above, average weight in Table 2 is a new metric defined by us as following:

$$f(S) = \frac{\sum_S (\omega^n_S) \sum_S (\omega^e_S)}{n_S m_S},$$

which is also the stopping criterion used in Reformed method in weighted graph. The higher the value, the better the quality. All the metrics listed above emphasize the inner density compared to the others. So the tables show that new methods
outperform normal partition like GN by metrics above in both weighted and unweighted graph.

There are some differences between the reformed method and local growing method. In terms of complexity, the reformed methods largely relied on the based algorithm. For instance, the complexity of GN algorithm is $O(n^3)$ in sparse graph like transition graph here, while the complexity is $O(mn)$ when it comes to the local growing method in a graph with $m$ edges and $n$ nodes. In terms of result, reformed method is more flexible due to the variety of community detection algorithms and convenience of implementing. Local growing method is quick and could follow some predefined conditions easily.

**TABLE I. SCORES IN UNWEIGHTED CASE.**

<table>
<thead>
<tr>
<th>methods</th>
<th>Internal density</th>
<th>Maximum-ODF</th>
<th>Flake-ODF</th>
</tr>
</thead>
<tbody>
<tr>
<td>GN</td>
<td>2.409</td>
<td>1.333</td>
<td>0.138</td>
</tr>
<tr>
<td>GN-clustering coefficient</td>
<td>1.405</td>
<td>0.833</td>
<td>0</td>
</tr>
<tr>
<td>GN-efficiency</td>
<td>1.405</td>
<td>0.833</td>
<td>0</td>
</tr>
<tr>
<td>GN-2-level efficiency</td>
<td>1.405</td>
<td>0.833</td>
<td>0</td>
</tr>
<tr>
<td>local</td>
<td>1.405</td>
<td>0.833</td>
<td>0</td>
</tr>
</tbody>
</table>

**TABLE II. SCORES IN WEIGHTED CASE.**

<table>
<thead>
<tr>
<th>methods</th>
<th>Internal density</th>
<th>Maximum-ODF</th>
<th>Flake-ODF</th>
<th>Average weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>GN</td>
<td>-3.984</td>
<td>2.564</td>
<td>1.219</td>
<td>72.719</td>
</tr>
<tr>
<td>GN-ccf1</td>
<td>-11.431</td>
<td>1.676</td>
<td>0.325</td>
<td>102.951</td>
</tr>
<tr>
<td>GN-ccf2</td>
<td>-11.431</td>
<td>1.676</td>
<td>0.325</td>
<td>102.951</td>
</tr>
<tr>
<td>GN-ccf3</td>
<td>-11.431</td>
<td>1.676</td>
<td>0.325</td>
<td>102.951</td>
</tr>
<tr>
<td>GN-2l eff</td>
<td>-11.431</td>
<td>1.676</td>
<td>0.325</td>
<td>102.951</td>
</tr>
<tr>
<td>local</td>
<td>-22.633</td>
<td>1.748</td>
<td>0.417</td>
<td>154.017</td>
</tr>
</tbody>
</table>

**TABLE III. SCORES OF A PERSON'S 100-DAYS CASE.**

<table>
<thead>
<tr>
<th>methods</th>
<th>Internal density</th>
<th>Maximum-ODF</th>
<th>Flake-ODF</th>
<th>Average weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>GN-ccf3</td>
<td>-14.382</td>
<td>2.214</td>
<td>-19.307</td>
<td>2.71E+04</td>
</tr>
<tr>
<td>local</td>
<td>-51.474</td>
<td>2.028</td>
<td>-9.471</td>
<td>1.35E+05</td>
</tr>
</tbody>
</table>

**Application Example**

Since there is no 5G user movement data now, we use the data from [15] to give a sense of practical user transition graph. One person's 100 days of movement data is transferred into a graph. And the communities are found separately by GN-based methods and local weighted algorithm. There scores with $\alpha = 0.5$ are calculated as the Table III.

Figure 3 shows the communities found by Local Growing method while Figure 4 by GN-based methods. From the scores and figures, we found that the local growing method outperforms the GN-based methods. In the somewhat chaotic graph
like a person's transition graph, GN algorithm has a hard time finding community structure, which leads to the results above.

Figure 3. A Practical Example: Local Growing Method.
CONCLUSION

This paper brings up a new problem in community structure that considers getting rid of some loosely connected nodes. Two methods are showed on a simple example evaluated by proper quality metrics. The motivation of this work comes from the background in 5G and we expect its application in other cases.

Before the arrival of 5G, there is no access to user data while problems and methods could be studied based on prospective 5G structure. In the future, we plan to figure out more 5G features related to users and suggest more methods to satisfy the various needs in 5G.

ACKNOWLEDGEMENT

This work is supported by the National 973 Plan project under Grant No. 2011CB706900, the National 863 Plan project under Grant No. 2011AA01A102, the NSFC (11331012, 71171189, 11571015), the “Strategic Priority Research Program” of CAS (XDA06010302) and Huawei Technology Co. Ltd.
REFERENCES