Summary

The project is mainly about detecting communities in spatial networks. Nowadays, as more and more complex systems are represented in a way by the network which nodes and edges are embedded in space, thus, knowing how to detect communities in spatial network is of great significant. Comparing with the common community detection methods may sometimes be misleading and cannot be able to find the underlining information of nodes, on the other hand, some researches have shown that it is possible to discover hidden underlining information by using the proper methods.

The list of things have been done in the project shows as follows:

- Review the some of the most widely used community detection algorithms and different versions of calculating the modularity.
- Created the algorithm for calculating the modularity applied for the spatial network based on the null mode proposed in the paper [14].
- Implemented a modularity maximization algorithm for spatial network based on CNM algorithm, see chapter 3.

Acknowledgements

I would like to express many thanks to my project supervisor Dr Steve Grogery, without his advice, guidance, encouragement and support, it is impossible for me to research in this field, or even accomplish the project.

I would like to thank Dr Andrew Calway and Dr Ian Holyer, their comments and feedbacks helped me a lot during the period of doing the project.

Finally, I want to thank my family for their support.

Table of Contents

Abstract

The project mainly focuses on the community detection for the spatial networks. As spatial network is a special kind of network when comparing with the other networks, the nodes in the network represent their positions in the space.

In the condition that recently more and more complex systems are in the way that nodes and edges organized embedded in space, however, the normal community detection measures may be misleading and cannot find underlining information about the nodes in spatial network. Thus, developing a proper method for detecting communities in spatial network is of great significant for this project.

1.introduction

This chapter will explain the aims of the project and will also explain the organizational structure of this article.

1.1 Aims of the project

The aim of project is to develop and demonstrate the methods for the spatial network community detection. Try to do some experiments with real networks and find out whether we can get communities or not even after ignoring the distance or the effects of proximity.

1.2 Key point

The key point of the project is to find out whether distance is taken account of when finding the communities.

1.3 Structure of the article

This article mainly consists of four Chapters. Firstly, the background explains the information about networks and especially the spatial network properties, community structure and some widely used community detection algorithms, then gives the definition of modularity and what has been done about modularity and the modularity based community detection methods and illustrates some problems with the modularity optimization. Secondly, in the experiment chapter, it gives the details about the two experiments' data and methods. Then, in the result discussion chapter, it depicts the ways of testing methods and comes to some conclusion of the two experiments. Finally, the author discusses about some future works.

2 Background

2.1 Network background

As we all know, network has been studied for ages, and nowadays, it becomes an interdisciplinary, which appears not only in computer science area, but also the other fields like mathematical sociology, economics, biology and even transportation industry. With the data of complex network getting more and more complicated and the scale of network growing larger and larger, how to describe and analysis complex network system is becoming more and more important [1].

Graph theory can be used to describe and analysis network system which contains different elements. In the graph theory, a network includes different items. Nodes, for example, sometimes called vertices, others like edges connect different nodes, together build up a network. The simplest network is just about some edges joint some vertices, shows in Fig. 1.

2.1.1 network types

Network sometimes can be complex, with more than one type of nodes and edges. In addition, vertices and edges may carry some different properties as well, set a social network as an example, nodes may stand for male or female, their age, living place and so on, while, edges may describe the relationship between two people. If edges in the graph run only in one direction, in this situation the graph sometimes called directed graph (shows in Fig. 1). On the other hand, the edges run in both two directions, the graph sometimes called undirected graph (shows in Fig. 2).

In the network, edges may carry weights (For example, in social network, when edge stand for the relationship between two people, weight maybe how well each person know other people), in this case, the graph or network is weighted (shows in Fig.2), otherwise, un-weighted. In other case, the vertices may be divided into two different types, and edges connect each of the vertex [2].

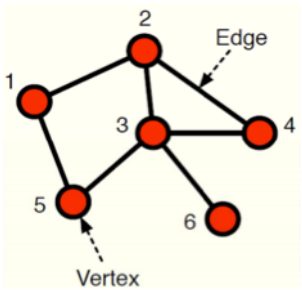


Fig.1 a simple undirected graph. Ref. [18]

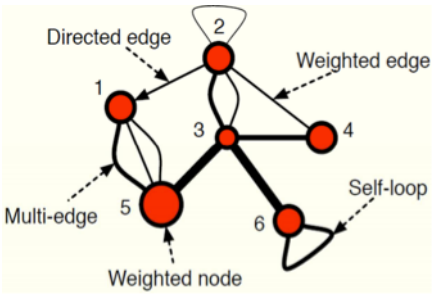


Fig.2 weighted and directed graph. Ref. [18]

2.1.2 random network

Random network, or random graph, the graph is not in an order, that means the possibility for two nodes connecting each other or there exits an edge between the two nodes is equal for all the other nodes in the graph. Most of the nodes have the almost same degree (shows in Fig. 3).

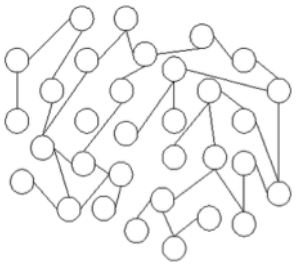


Fig.3 a simple random network. Ref. [19]

Fig.4 a simple real network. Ref. [20]

2.1.3 real network categories

Real world network is quite different from random network, its organization structure shows the network is in an order (shows in Fig. 4). Nodes are organized not appears at random. It sometimes follows the power law and small world effects which will be mentioned in late part of this article.

Network in the real world can be divided into different categories, for example: social network, biological network, information network and technological network.

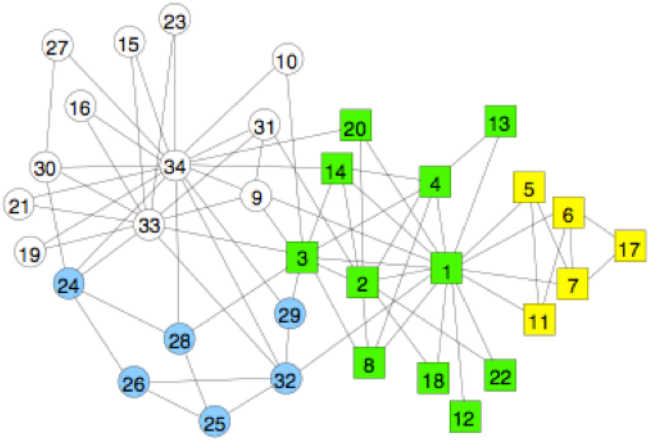


Fig.5 a sample of social networks. Ref. [3]

Social network is a network that nodes stand for an individual, a group, or an organization, and edges represent the relationship or interaction between them [3]. Thus, in short, social network is actually a relationship structure. The relationship may be about the friendship, kinship, business etc. According to different relationships, different attributes of nodes, the analysis or specific

detection methods can be different. As shows in fig. 5, it is an example of social network, it is a very famous network benchmark used to test different kinds of methods of community detection, it is known by Zachary's karate club.

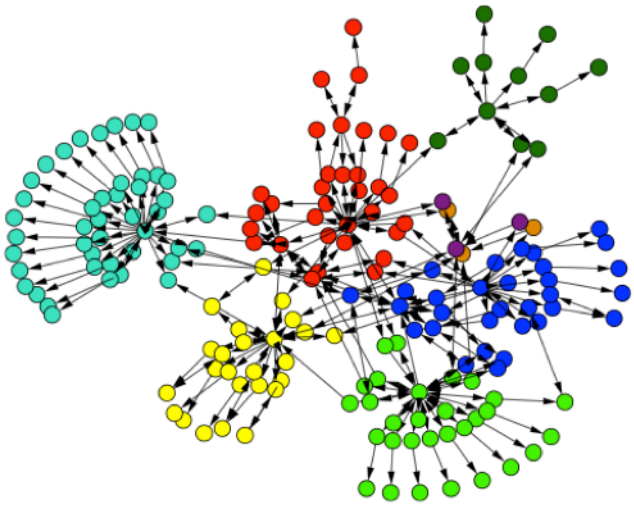


Fig.6 a sample of technological networks. Ref. [6]

Information network also named knowledge network, in which the vertices contains information of the structure. A good example is the citation network [4].

Technological networks (shows in fig. 6) are man-made networks in order to arrange the resource distribution. A typical case is the World Wide Web system [5]. In the fig. 5, it is a web network that made up by a lot of web pages and the links shared between them. The nodes are distinguished by different colors.

Biological networks, just as the name implies, represent the biological systems. The food web is a widely studied example [6]. The nodes in the food web stand for different species, and edges connect them represent the prey relationship between them.

Also, there are many other types of networks.

2.2 Spatial network

As the real world networks mentioned above, in fact, many networks that make up the complex systems, their nodes and edges are often embedded in physical space, or we may say it is relevant to space. Examples contain road networks, social networks, mobile phone networks, travel flow networks, power grids and other networks that are relevant to space.

2.2.1 spatial network definition

Spatial networks are networks which nodes or edges are embedded in the either two or three-dimensional physical space [8]. For the most of the real word network application system, space always means the Euclidian space in the two

dimensions, we can call the network planar network, either [9]. Nevertheless, spatial network is not just planar.

From the definition of spatial network, we can get the information that generally the possibility that two nodes will connect with each other, or there exits a link between the two nodes is decreasing as the distance increases.

2.2.2 Generalities

2.2.2.1 spatial constraints

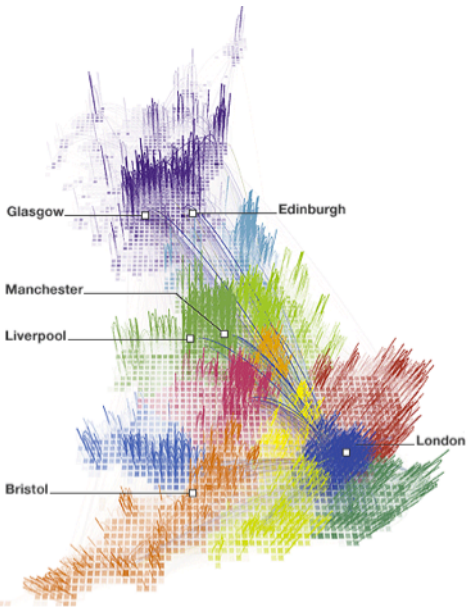


Fig.7 the phone call networks of Great Britain reference from BBC news

Space may lead to constrains lying on the link which connect two nodes, one step closer, it may even contribute to the changing of network architecture or connectivity patterns.

In the social network, for example, people tend to make friends near their home or in their neighborhood. The flight connections or the urban road networks, rely on the distance obviously. In the biology network, say, the neural networks, since the axons' length in the brain carries a cost [7], thus, spatially close or not have a great difference on the probability of the connection between the regions in the brain. Above all, we know that the spatial distance can be not only the geographic distance, but also social distance, or even anything other form of distance that measures the cost between the nodes. Thus, we can conclude that distance plays a key role in influencing the network topology.

As shows in fig. 7, it is a network that represents the phone call of Great Britain, the colors indicate different regions in UK. A team of MIT university did the experiment that recorded 12 billion an anonymous calls of landline phone and the results reveal that the people are more prefer to communicate with the other people who live geographically near them. We can clearly see the dense communities in different regions in the figure.

Duo to the contribution of space, the network topology or architecture is distinct from that of random networks [10]. In the random network or random graph, the edges connect between a pair of vertices have the same probability, or say, the edges distributed uniformly among those nodes.

2.2.2.2 gravity model

Gravity model often depicts the flows in the spatial network [11]. It has used for a long time to model the flows of interplay of different districts influenced by distance. The equation shows below:

$$T_{ij} = N_i N_j f(d_{ij})$$

Where i and j stand for two different nodes, and N_i shows the importance of node i, for the same, N_j indicates the significance of node j, the function f give us a description of how space influence the interaction between node i and j. d_{ij} is the physical distance between node i and j.

From the equation, we can get the information that the intensity of interplays of node i and node j is proportion to the distance between d_{ij} and the possible interact numbers $N_i N_j$.

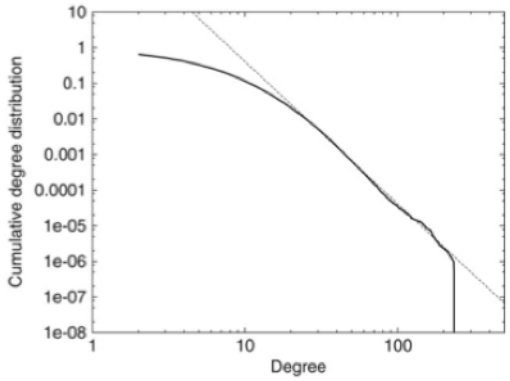


Fig.8 degree distribution behaves in the example of mobile network. Ref[12]

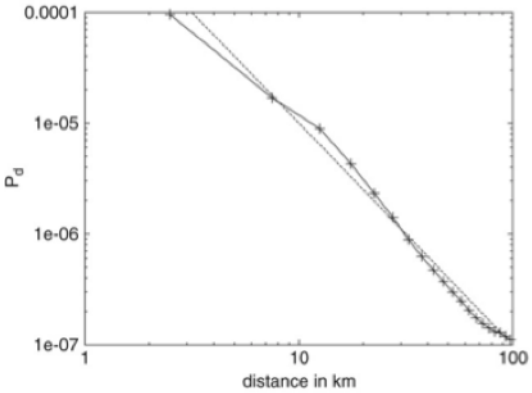


Fig.9 the probability two people living at a distance follows power-law. Ref[12]

In many communication networks or socio-economic systems, the function f may follow the power law, like $d_{ij}^{-\partial}$. In a typical example [12], the paper analysis a communication network's properties which data comes from a mobile phone company from Belgian. The communication network is made up by 2.5 million customers that every customer has gave his home address zip code as geographical information. And in a 6 months period, the network produced 810 million communications, including phone calls and text messages. The empirical analysis reveals that the probability that two customers living at a distance d_{ij} will make a phone call to each other follows the gravity model d_{ij}^{-2} (shows in fig. 9), in other words, the possibility will decrease as distance increase. And the degree distribution follows the power-law K^{-4} (as shows in fig. 8 the dashed line).

2.2.2.3 Scale-free networks

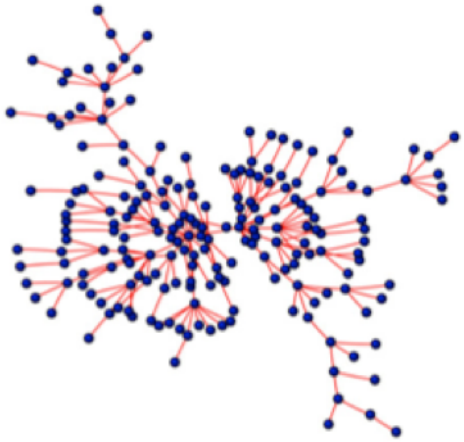


Fig.10 an example of scale-free network. Ref[18]

Scale-free networks are networks with the degree distribution of power-law property. Many real networks are scale-free networks, where the distribution of nodes is heterogeneous, many nodes have few nodes to connect with, however, some of the nodes have a huge numbers of neighbors, sometimes is called "hub". There are many examples about the scale-free networks, one of which shows as fig. 10, the network contains 100nodes and the average degree is 2.

Besides the properties of power law, there still other properties of the complex network, such as the Small world effect.

2.2.2.4 Small world effect

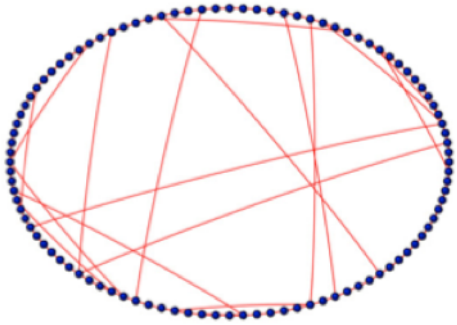


Fig.11 an example of small world effect. Ref[18]

Small world effect is the fact that most of the nodes can be arrived by a short path, however, the nodes are not actually neighbors to each other.

In the small-world network, the distance between two nodes that randomly picked growing logarithmically. The small world effect is often seen in real networks, for instance, the information spread very fast through the internet. An example shows in the fig. 11, the network has 100 nodes and the probability of connection is 0.1.

2.3 Metrics for analyzing spatial network

Although more and more models have been built for spatial network, metrics or tools which reveal the information of great use for the spatial networks' topology is rare. Therefore, using the metrics [14] that ignore the useful information of the spatial position or structure about the nodes while analyzing spatial network, definitely, the result we get may not be relevant for those non-spatial networks. As a result, analyzing spatial network with an appropriate metric is in great request.

Some basic measures for analyzing spatial networks described as follows.

2.3.1 Basic measures

2.3.1.1 Clustering coefficient

Clustering coefficient, or sometimes is called transitivity, is an important way to measure network. For example, in a network, if node A is connected to node B, and node B is connected to node C, we may conclude that there is a high probability that node A will be connected to node C. From the network topology's view, clustering coefficient implies the number of triangles in the network where contains groups of three nodes, each one is connected to the others. The equation for calculating clustering coefficient is as follows:

$$C = \frac{(number\ of\ triangle)\times 3}{(number\ of\ triples)}$$

As the properties of spatial network mentioned before, it is not difficult to understand that spatial networks are naturally clustered.

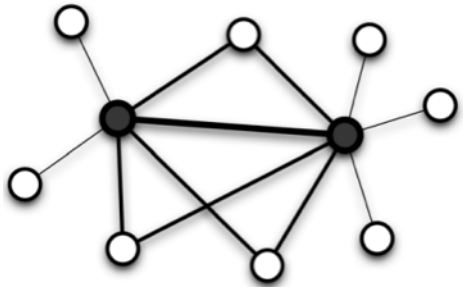


Fig.12 an example of edge clustering coefficient, ref [28]

As it showed in fig.12, the two nodes that in dark color each have five and six nodes to connect with. As we can see from the figure, the clustering coefficient is $C = \frac{3}{5}$.

2.3.1.2 Adjacency matrix

Given a graph with N nodes and E edges, the graph can be described by an adjacency matrix A. A_{ij} is the adjacency matrix ($N\times N$) in the graph with N nodes and E edges. The matrix A_{ij} can be described as follows:

$$A_{ij} = \begin{cases} = 1 & (if\ i\ \ and\ \ j\ are\ connected) \\ = 0 & \text{others} \end{cases}$$

The matrix A_{ij} is symmetric on the condition that the graph is undirected.

2.3.1.3 Degree distribution

Degree distribution is another important element for analyzing spatial network. At first, degree means the number of edges that a vertex connected to in the network. p_k is defined to be the probability that a vertex has the degree k. The equation for counting the degree of a node is below:

$$k_i = \sum_j A_{ij}$$

In the equation above, A_{ij} is the adjacency matrix. Using the equation, we are able to manufacture a histogram that shows the different p_k for different nodes. Thus, the histogram gives us the network degree distribution.

In the random network, for the reason that each one of the edges has the same probability to connect different nodes, or say each vertex has the same degree, thus, the degree distribution is binomial or Poisson, or in the simple words, the distribution is homogenized. While, in the real-world networks, the degree distribution is totally different from that of the random network. Edges and vertices are not well distributed, instead, are following a power low. That is to say, in the graph, many low degree vertices are in coexistence with those vertices with high degree. What is more, the feature that the density of edges within some areas of the graph connect to vertices is much higher than that of connect with other area, in the real world network is called community structure [13].

2.4 Community structure

Community, or on the other scene, called modules or clusters, as it shows in fig.13, is a group of vertices which has a higher degree of edges inside the group than between the different groups. Thus, it is a mesoscopic structure of the network.

Just like our real life society, we are in different groups or circles, such as family circles, friend circles, etc. While on the internet, people who feel difficult to meet friends form the friends circles online with people in the similar interests. Communities can not only be found in social life, or say social network, but also happen in other network systems, such as engineering, biology, politics, etc.

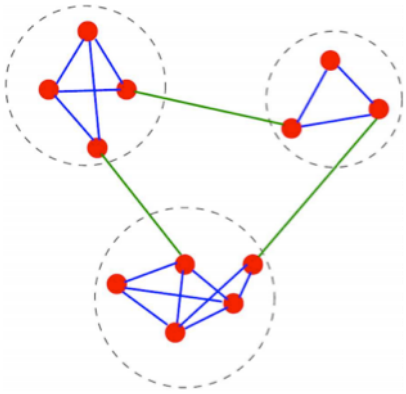


Fig.13 the illustration of community structure ref. [20]

2.4.1 reasons cause community structure

The reason that cause the community structure maybe, “homophily”, which means nodes tend to connect to the other nodes that with similar property, or “introduction”, which states that if different nodes have the same neighbor, they have the huge probability to link to each other.

2.4.2 community structure catergories

Community structure has lots of different types: disjoint community, overlapping, hierarchical (shows in fig.14), fuzzy overlapping and ordered community.

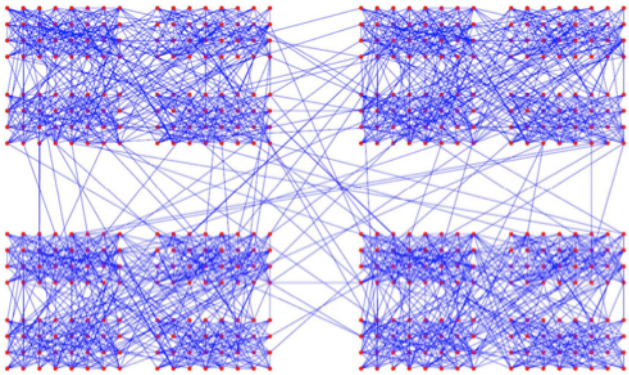


Fig.14 hierarchical community structure ref. [15]

Disjoint structure means in the graph, a partition consists of different disjoint communities, each vertex is in one community. As it shows in the fig.15, the disjoint community is: $\{\{a, b, c\}, \{d, e\}\}$.

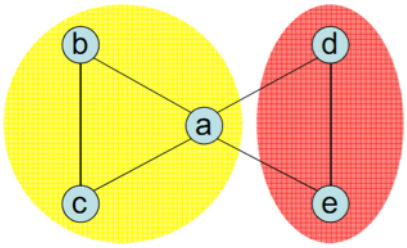


Fig.15 disjoint community structure ref. [20]

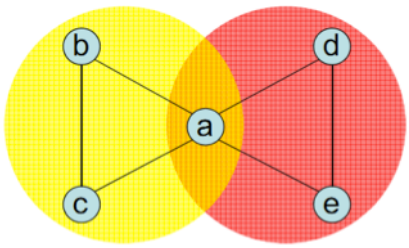


Fig.16 overlapping community ref. [20]

If there exists a vertex that is in different communities, thus, the community structure is overlapping. As the illustration in fig.16, the overlapping community is: $\{\{a, b, c\}, \{a, d, e\}\}$.

Inside a big community, if it contains a smaller community, or the other communities within this community, then we can call this community structure hierarchical structure. Our human body is a good example for describing hierarchical structure, it is consist of organs, and organs of tissues, and tissues of cells, etc.

2.5 Community detection

2.5.1 why we need community detection

Community detection in network origins from the research work on sociology, related theory including graph theory, pattern recognition and so on. Traditional community detection (or graph partitioning) methods always treat the network as can be divided, the number of community is decided by people. Nowadays, the

real-world networks sometimes appear in large scale. The organization of nodes and the way edges connect different nodes are more complex.

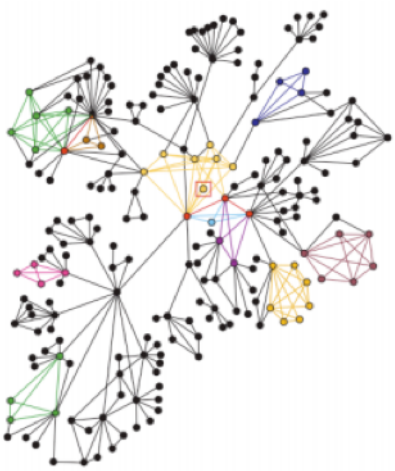


Fig.17 a phone call network ref. [22]

As a result, discovering the communities (clusters, modules) is of great significance for many fields and many complex systems. Revealing the information of community will help us understand the network structure better (such as shows in fig. 17), know the uncover information between nodes which have similar property better, and make it possible for us to draw or visualize the network better.

Community detection also helps us develop good applications. Such as the recommendation systems on the website of online shopping mall. By applying community detection, we can cluster the customers into different communities, each community is composed by customers with similar interests and live near each other by geographically, say in the same city or same country. Thus, sellers are able to recommend for the customers much better and improve the possibility of product selling. An example of the purchase network shows in fig. 18 below, the network includes 1684 communities, for each community the average number of items for each community is 243.

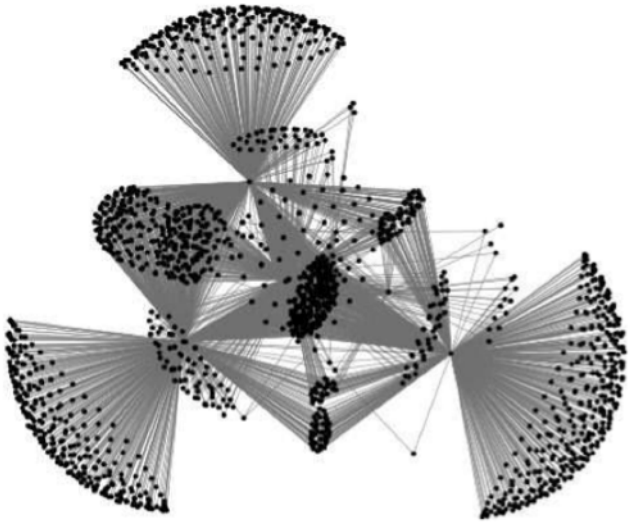


Fig.18 purchase network of amazon.com ref. [17]

As community detection is so important and worth researching, we should focus on it. Facing the problem of detecting communities in large-scale real world network, we need to find the answer for two questions. First question is whether the network contains community structures. The second question is how to find the community structure efficiently.

2.5.2 traditional method for detecting community structure

Traditional method of detecting communities (shows in fig. 19) is as follows:

1. Calculate every pair of node's weights in the network, for example, for node i and j, the weight is W_{ij} , it tells how close the two nodes are connect to each other.

2. Pick the *n* nodes that not exits edges connect with them, and then add edges between them in an order that the largest weight comes first and the smaller one comes after and so forth, till the weight is the smallest.

3. Repeat adding edges, finally form to be a community.

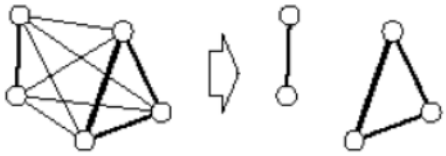


Fig.19 Traditional method for detecting communities ref. [21]

2.5.3 new method for detecting community structure

Recently, numbers of community detection algorithms have been designed. Among them, there are two kinds of community detection methods that are commonly used, first kind is based on the betweenness (node betweeness or edge betweeness) or other calculation methods, the other one is based on network modularity.

The simplest way for community detection is to as follows:

1. Find the edges that connect nodes that belong to different communities, the so-called edge of inter-community.

2. Delete these inter-community edges.

3. Keep checking and find out whether exits any edges left or not.

Thus, we finally get the community that separate from other communities. Comparing with the traditional method, in the new method, we concentrate on discovering boundaries, moreover, boundary detecting is rely on the betweenness.

2.5.4 definition of betweenness

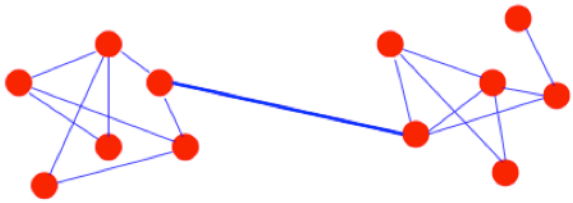


Fig.20 illustration of edge betweenness ref. [18]

The most widely used community detection algorithm Girvan and Newman algorithm is based on betweenness.

Betweenness is the value which shows the edges' participation frequency [26]. As shows in fig. 20, the edge that in dark blue solid line has the largest edge betweenness. According to Girvan and Newman algorithm's definition, there are three betweenness: geodesic degree betweennesss (usually known for the edge betweeness), random walk betweenness and current-flow betweenness.

Right now, we focus on the edge betweenness, edge betweenness is the number of the shortest paths between all the nodes that passing by and running along the edge. Because we know that the edges called inter-community edges connect two different communities, therefore, the paths from one community to anther have to pass the inter-community edge.

2.5.5 Calculation of edge betweenness

For the simple case, if there is only one shortest path, the steps are as follows:

1. There is only one path between the top S to the rest of nodes, so, start at the bottom and set the value to 1.

2. Then, calculate the sum, which is the sum of all the children edges.

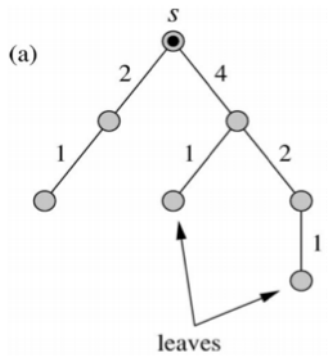


Fig. 21 simple case of betweenness ref. [23]

As in the fig. 21, we start at the leaves, set the value 1 to them. Then calculate the sum of the values that show in the figure. Its direction is from bottom to up, the paths are forming like a tree. What's more, the computational complexity can be

expensive, for one pair the computational complexity is O(m), and O(n^2) for all the n nodes pairs.

For the other situation, there exits more than one shortest path (shows in fig. 22), the steps for calculation betweenness is as follows:

1. Firstly, compute the number from S to other nodes.

2. Secondly, set an appropriate weight for path that will count.

3. Then, get the sum of the nodes that reached.

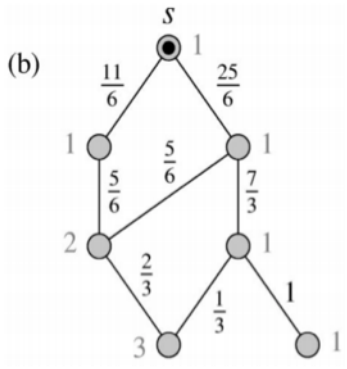


Fig. 22 complex case of betweenness ref. [23]

As we can see from the fig. 22 above, the result of betweenness equals to the nodes that is reachable. The details of how to calculate the shortest path of distance is as follows according to fig. 22:

1. For example, at the beginning, we set the distance of the source node s $d_s = 0$ and weight $w_s = 1$.

2. After that, set the nodes which next to source node s the distance and weight, such as, node i, $d_i = d_s + 1 = 1$, and $w_s = w_i = 1$.

3. After this initialization, we need to set the distance value the weight for other nodes which connect to those nodes like node i mentioned before. For each node, we should set the values according to three situations.
    a) At first, if node j has no distance at the moment, we need to set the $d_i = d_s + 1$ and the weight value $w_s = w_i$.
    b) Secondly, when node j already has the distance value $d_i = d_s + 1$, we should set the weight value as $w_j = w_j + w_i$.
    c) In the other situation, that if node j has a distance value, however the value $d_j < d_i + 1$, then we do not need to take action.

4. Repeat the step 3 for other nodes until every node has the distance value.

In the fig. 22, the number 3 in the bottom of the figure is the number of shortest paths.

And the details about how to calculate weight of edge is as follows:

1. Find out each leaf node t, here, leaf means that from source node s to node t without any other nodes. In the fig. 21, it shows what leaf is.

2. For every node i which is the node t's neighbor, we set the edge weight as w_i/w_t.

3. After that, from the bottom of the fig. 22 up till to source node s, if a node j, where the distance from node s to node j is larger than between node i and node j, we set the value plus one for the neighbor edges. And then multiply by the value w_i/w_t .

4. Repeat the third step until it reached node s.

Thee time complexity for this situation is O(mn) in every iteration.

2.5.6 random-walk and current-flow edge betweeness

As we all know, information are spreading in a random for the reason that signals flow at random rather than by the shortest paths. Thus, in this situation, we cannot treat the edge betweenness as the number of shortest paths any more, here, edge betweenness should be the frequency of a random walker walking by the edges on the network. Just like the random network's property, the random walk moves from one node to another node's possibility is the same.

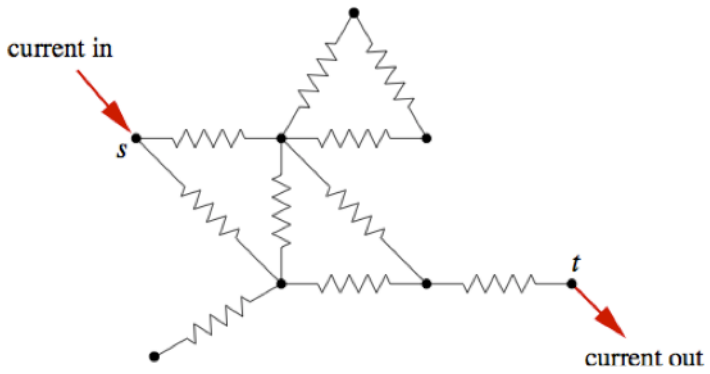


Fig. 23 an example of the resistor networks. ref. [6]

Before knowing how to calculate current-flow betweenness, we need to know the resistor networks first (as shows in fig. 23). Resistor network is the network which edges carry the unit resistance. For the reason that the network is created by putting the unit resistance on each edge. As we can see from fig. 23, the unit current source is at node s and ends at node t. Current flows in resistor network followed the Kirchhoff's laws. If voltage for node i is V_i, the for all the nodes i, it satisfies the Kirchoff's equation as follows:

$$\sum_j A_{ij}\left(V_i - V_j\right) = \delta_{is} - \delta_{it}$$

A_{ij} is the adjacency matrix for the network, $\sum_j A_{ij}\left(V_i - V_j\right)$ is the current flow from node i to the other node j in the resistor network, and $\delta_{is} - \delta_{it}$ is the source of the current and the end of the current.

Thus, through all the nodes in the network, the current-flow betweenness can be calculated by calculating the average of current value for all the edges. For the reason that in the random walks situation and current-flow's voltage both satisfy the similar equation just like the Kirchoff's equation above, thus, it is not strange that why the current-flow betweenness calculation method is similar to the random-walk betweeness calculation measure.

The time complexity for calculating the random walk betweenness and current-flow betweenness is $O(n^3)$, while, as it mentioned above, the complexity for the edge betweenness is $O(n^2)$.

It is clearly that the calculation of edge betweenness is much quicker. Actually, the Girvan-Newman algorithm based on calculating edge betweenness normally get better results than others.

After knowing how to calculate betweenness, the Girvan-Newman algorithm seems not that hard.

2.5.7 GN algorithm

The steps of the Girvan-Newman algorithm are as below:

1. Compute the all edges' betweeness;
2. Choose the edge with the largest betweenness and delete it;
3. After deleting, recalculate all edges' betweenness again.
4. Notices that, until no edges left, or we should repeat the iteration.

The calculation for the betweenness is based on the breath first search. A lot of researches indicate that the third step of the four steps listed above is of importance to community detection for those significant communities. This step produces another factor to the time complexity. And it is easy for those strong community structure networks to partition.

There are numbers of modify version or refinement version for the Girvan-Newman algorithm. However, for the original version, there is no measure to judge which community is best, in other words, modularity has not applied to it yet.

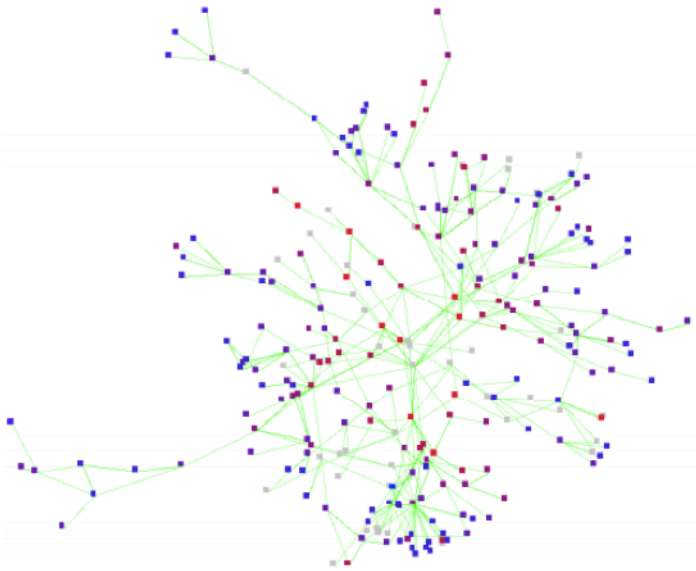


Fig. 24 an example of HP labs email network. ref. [27]

One modify version of Girvan-Newman algorithm is posted by Tyler, the merit is that it gains the speed of calculation. The new algorithm do the things as follows, at first chooses a node at random which treated as the center, and then calculate the edge betweenness for all the paths to find the contribution of the node to edge betweenness. And then do the same for all the other nodes.

The algorithm do not need to apply the modularity to it but based on another community definition. The partition is a community on the condition that if the partition has n_0 nodes and the edge betweenness value for each edge is no larger than $n_0 - 1$.

Although choosing different center nodes leads to detecting different communities, we may get good results by repeating the calculation steps for lot of times and even may improve the speed for the calculation.

One of the examples is the algorithm applied to the email network (shows in fig. 24). The drawback of the algorithm is that the nodes located between two communities may not be stable, say, sometimes one node is in a community, and the other time in another community. On the other hand, the algorithm has its own strong point, that is it can apply to overlapping community detection comparing to the original version of Girvan-Newman algorithm cannot achieve this.

Rattigan post another modification version of Girvan-Newman algorithm, comparing with the original version, it is faster. This algorithm applies the network structure index to the calculation of edge betweenness, which including the node annotation and distance method. Normally, the network is divided into different communities and we need to calculate the distance for every node in every community at the same time. This version of algorithm shows better performance on benchmark network and other networks like collaboration network or citation network.

The idea for another refinement version of Girvan-Newman algorithm posted by Chen and Yuan is that we should count the non-redundant paths instead of counting the shortest paths for the edge betweenness calculation step. Here, non-redundant paths mean that the end points are different from each one.

The reason why we should only calculate the non-redundant paths is that calculation of shortest paths may sometimes cause unbalanced communities, which the size of the community is different. The calculation of edge betweeness in the new measure generates better results than the ordinary one.

For the reason that each node is set into one community, thus, Girvan and Newman algorithm is not able to use for the overlapping communities.

While for the overlapping communities, one version proposed by Pinney and Westhead calculate the edge betweenness, either. However, nodes in the overlapping communities in this measure will be split into different communities.

CONGA (Community Overlap NG Algorithm) is another algorithm that suitable for the overlapping communities [24], where the GN algorithm is extended for overlapping usage. In CONGA, the node with the highest betweenness is split into two communities.

Besides calculation the edge betweenness, other methods like label propagation, clique percolation and fitness function maximization and so on are all help to find communities. And different methods have their own way of calculation, strong points and drawbacks.

For the reason that the later experiment stage, author will use the COPRA for the community detection part, coming after is the detail introduction about CORPRA.

2.5.8 Label propagation

The brief idea of community detection based on label propagation is as follows:

1. Label each node by using an unique id, as in the fig. 25(a), the id is from a to f;

2. Substitute each node's label that chosen the largest number from the neighbors for each label. If there are two nodes that the number is tied, we chosen it randomly. As in the fig. 25(b), if label a has the largest number and label b is just smaller than a, and so forth, f is the smallest number. Thus, a is replaced by b which is the largest number in label a's neighbor, and so forth for other labels.

3. Repeat the first and second step until each node has the largest number of their neighbors.

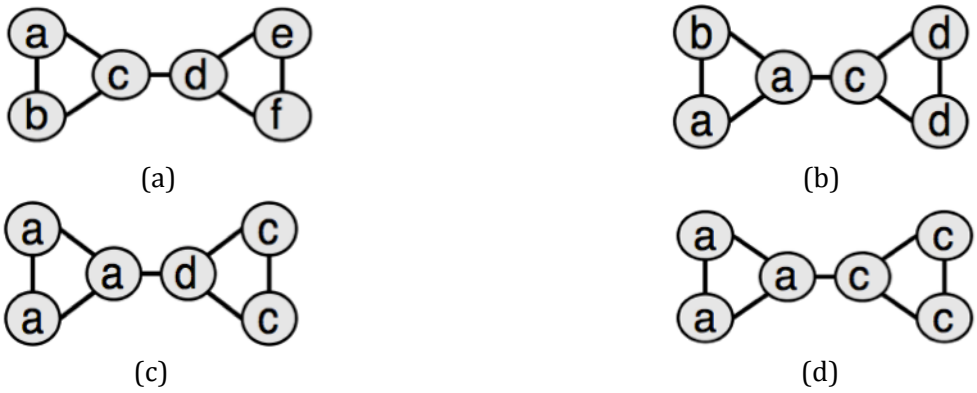


Fig. 25 case for label propagation ref. [25]

Using fig. 25 as an example to explain the details of how label propagation works, (a) is the origin network contains nodes: a, b, c, d, e and f. The sort of the number for the nodes in the network is as follows: a > b > c > d >e > f.

At first, for node a, it connects with node b and c, obviously, the number of node b is larger than node c, thus, a is replaced by b. While, for node b, its neighbors are node a and c, clearly, node b is replaced by node a. Do the same procedure for other nodes in the network of fig. 25 (a), we get the result as fig. 25 (b) shows. While, in a particular situation, for node b, it connect with neighbors are all labeled as a, thus, b is replaced by a. Finally, we get the final result as fig. 25 (d) shows.

The time complexity for label propagation is nearly linear time, $O(n)$ for the initialization and $O(m)$ for each step.

While for the overlapping communities, COPRA (Community Overlap Propagation Algorithm) is the algorithm that extended the label propagation algorithm to apply to overlapping community detection.

For the reason that in the overlapping network, one node always belongs to different communities, thus, applying label propagation to overlapping network, we need the more than one community id for node label.

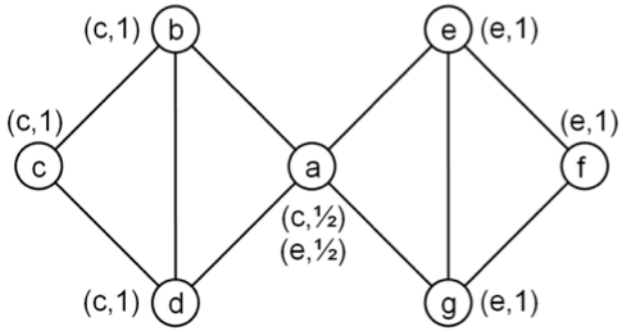


Fig. 26 case for label propagation on overlapping network ref. [28]

As the fig. 26 shows, every node now includes two elements, one is the label or usually called community identifier, the other one is the thing named belonging coefficient.

2.6 Social network and space

In the most cases, the communities found in the spatial network show heavy rely on the geographical factors and reveal almost none of information that may determine the network structure.

For example, in the social network, the nodes have both the structural information and non-structural properties. Structural information is just about the nodes' structural position. On the other hand, non-structural properties are mainly about the similarity and the constraint.

Similarity, for example, people who has the similar interest or hobby, may more likely to have a stronger relationship.

Constraint depicts that social relationship is directly or indirectly determined by the geographic proximity or distance. For instance, people who live near each other or in the proximity have the greater opportunity to meet each other in person than those who not live nearby.

Similarity and constraint are distinct at first glance, however, the coaction between them may lead to changing the network structure. For instance, people who have the similar interests may contact with each other frequently then they may become friends, afterwards, they are likely to live nearby. Mangle similarity with constraint, we can uncover the information of both the non-structural and structural.

Accordingly, it is possible for us to remove the influence of constraint, or space, so as to reveal more about the other factors of similarity hidden under the structural when the positions of nodes have been known by us.

For example, the Belgian mobile network that has been mentioned before, the network consists 571 modulus, 19 of the 571 modulus are merged into one in the capital city Brussels. The data is collected in a 6 moths period from the customer-to-customer communication by phone. And customers have given their zip code of where they living.

Therefore, this mobile phone network is a weighted, undirected and with 571 zip codes real network. The total number of calls is $\{A_{ij}\}_{i,j=1}^{571}$, and in each community, the number of customers is N_i for the community i. And the weights are the average value of communication time between tow people, the node i and node j in different communities.

Considering the social factors about the data, it is important for us to know that in Belgium, there exist two large linguistic communities, one called Flemish community mainly in the north and the other one is French community in the south [29].

After getting the data that wanted, the experiment can clearly show the different results according to different strategies. Firstly, if we consider both the similarity and constraint influence, the visualization result can be yielded shows in Fig. 27.

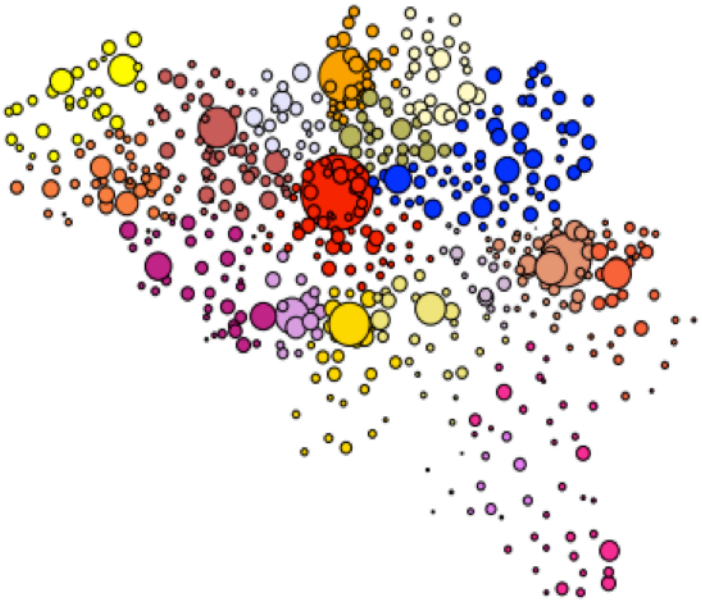


Fig. 27 case for label propagation on overlapping network ref. [28]

From the figure, the result clearly shows that eighteen communities have been found which the nodes in the communities are decided by the short distance forces.

The partition is in agreement with the linguistic division of Belgium, however, the two linguistic communities are not clearly detected. Thus, we may come to the other strategy.

Secondly, in another situation, if we remove the remove the influence of constraint, or space, so as to reveal more about the other factors of similarity hidden under the structural, the result shows as the follow fig. 28.

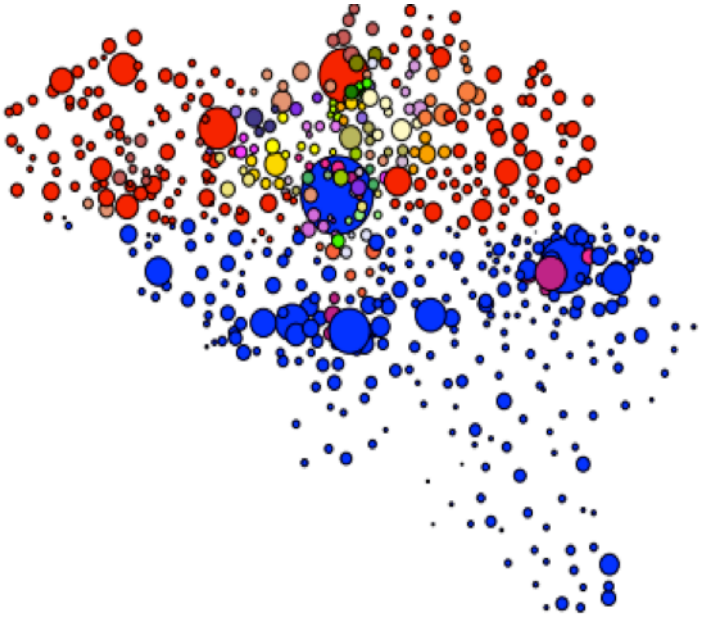


Fig. 28 case for label propagation on overlapping network ref. [28]

In this case, under the strategy it obviously discovers the two large communities. As much as 75 percent of the whole communities are perfectly match to the country's linguistic separation. Specifically, in the capital city Brussels, the big red circle in fig. 27 and the big blue circle in fig. 28,roughly 80 percent of people speak French, it has been discovered, either.

Thus, the key point in the paper is that how can we figure out whether distance is taken account of when finding communities in the spatial network. That is to say, how can we detect communities when we remove the effects that determined by space or influenced heavily by space in the spatial networks?

As we know the fact that the standard modularity for finding community ignores the spatial nature. Therefore, it is incapable of revealing the information due to other underlying factors.

Consequently, we need to optimize the Newman-Girvan modularity to adapt to the spatial network and go beyond the standard method to reveal the important information. While at first, we need to know what is modularity and how to use it.

2.7 Modularity

As it known to all, the basic idea of community detection is to divide nodes into communities. After getting the communities, how can we judge which community is the best one?

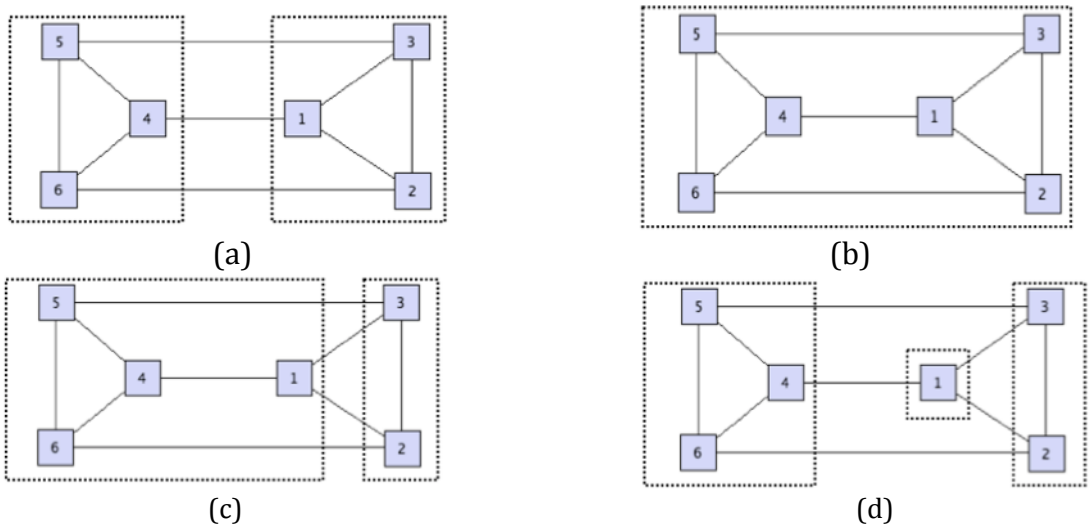


Fig. 29 four results of community detection ref. [26]

As the fig.29 shows above, a small network has 6 nodes, by applying community method, we can get lot of different results, the fig. 29 shows 4 (a, b, c, d). In the fig a and fig c, the network is divided into 2 communities. In fig b, the network as a whole is a community. Network in fig d is divided into 3 communities.

According to the definition of community, a group of vertices that has a higher degree of edges inside the group than between the different groups. If we just count number of edges to judge the quality of the community detection is not a good way, for the reason that a good community detection result is not only the one which exits less number of edges between communities, it is the one that less than expected.

In other words, between the groups, if the number of edges is less than the possibly number that we expect, normally, by random, we may say there exits a good community. If the difference is obvious, we may find a better one. This way of thinking can be able to apply to qualification equation named modularity.

Modularity is in fact a function which judges how good a community is when the graph is partitioned into different communities. As a result, we are able to calculate the value of the quality function for each partition and sort out the good partition. Thus, to find the better partition of the network is to find a larger value of modularity.

For simplicity's sake, author tends to talk about modularity in the undirected, weighted network. At the moment, the most popular or widely used method is the Newman and Girvan modularity [6].

2.7.1 Newman-Girvan modularity

The NG modularity is based upon the idea that comparing the actual fraction of edges within the community with the expected fraction of edges in the null model. The modularity can be written as below:

$$Q = (fraction\ of\ links\ whithin\ communities) - (expected\ fraction\ of\ links)$$

In the mathematic way:

$$Q = \frac{1}{2m}\sum_{ij}\left(A_{ij} - P_{ij}\right)\delta(C_i, C_j)$$

In this function, A_{ij} is the adjacency matrix that is symmetric, if node i and j are connected, A_{ij} equals 1, if not, A_{ij} is 0. m is the total number of edges in the graph. $\delta(C_i, C_j)$ function equals 1 if node i and j are in same community, if not, $\delta(C_i, C_j)$ equals 0. P_{ij} is the total number of the edges which are expected to be exist in null model between the node i and j.

One of the merits of this modularity measure is that the null model plays a n important role. Although there are many ways can be applied to find communities, actually, the null mode which assumed help to make the judgment to a community definition.

2.7.2 NG null model

Null model is used for comparing with the graph that you are studying and determine if the graph shows the community structure or not. For instance, you can choose random graph that we have mentioned and keeps the original graph's structure properties.

You can choose any null model as you like, there are exists some possible choices. For example, the Bernoulli random graph, the probability that edges connect two nodes are the same, however, for the real world network, it is not suitable for the reason that real networks' degree distribution is totally different. Thus, for the real network, we should choose an appropriate null model.

The most frequently used null model is the Newman and Girvan null model [16]. In the NG-null model, the degree distribution should roughly the same as the network you are studying. In order to achieve this goal, the expected node's degree within the community should conforms to the actual degree of the node that in the real network. Thus, the degree of node i satisfies the expression:

$$\sum_{j} P_{ij} = k_i$$

$k_i$is the strength of the node i. For the reason that the node i is chosen and can be connected by edge to any node randomly that only rely on the degree k_i, thus, it is the same for the node j. In such a manner, we can conclude that the expected number P_{ij} obey the expression:

$$P_{ij}^{NG} = \frac{k_i k_j}{2m}$$

Accordingly, we may set the Newman-Girvan modularity function as follows:

$$Q_{NG} = \frac{1}{2m}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta(C_i, C_j)$$

Fig. 30 two examples of computing modularity ref. [6]

In the fig .10, on the left, the modularity may be $Q = 6/7 - 1/2 = 0.357$ depends on the null-model you choose. The modularity on the right may compute like $Q = 5/7 - 29/492 = 0.122$

The Newman-Girvan null model is mainly about the random network constrained by the degrees of the nodes, we may have questions, it the constraint so important? Or what if the nodes has extra non-structure information?

2.7.3 Spatial null model

As it mention before, it is in great demand of finding a proper metric for analyzing the spatial network, especially for the community detection. There are lots of practical applications related to spatial network, however, the up to the minute method for discovering community in the spatial network is the standard Newman-Girvan modularity or the optimized one, which is not good enough for the spatial network. The standard Newman-Girvan modularity totally ignores the spatial property of the practical system.

Coming next I will focus on the modularity specialized in the spatial network. As mentioned before, the Newman-Girvan null-model based on the adjacency matrix under the rudimental community structure.

Thus, the information that we get from the NG-null model is only the basic information about the nodes, it provide poor information about the additional information like the hidden factors that changing the network shape.

Consequently, it is in great demand to include non-structural information in the definition of modularity so as to reveal the underlying information of the nodes, especially for those networks that distance plays an important role.

In those networks which distance makes a big difference on the nodes' connectivity, that is to say, distance determines the two nodes' probability of connecting, the spatial null model should be applied.

The spatial null model is influenced by gravity model, the equation shows below:

$$P_{ij}^{Spa} = N_i N_j f(d_{ij})$$

In the equation, N_i or N_j depicts the significance of node i or j. And the function $f(d_{ij})$ defined as follows:

$$f(d) = \frac{\sum_{i,j|d_{ij}=d} A_{ij}}{\sum_{i,j|d_{ij}=d} N_i N_j}$$

From the equation we know that the determinant of the connectivity of the nodes is the data we get. The meanings of N_i or N_j is decided by the network you study, for example, N_i maybe the age of a person in social network, the number of the residents in a country or the other cases. Specifically, if space fails to act or not make any difference between the nodes, then the function $f(d_{ij})$ can be substitute by Newman-Girvan null model.

Duo to taking account of the non-structural information, the spatial null model's modularity Q_{Spa} will reveal the communities dominated by underlying forces or other non-structural factors and may give better result on the spatial distant nodes which forced by the force similar to the gravity force when compared with Q_{NG}.

2.7.4 methods rely on modularity

Newman-Girvan modularity has been an important and widely used quality function for the community detection algorithms.

As it mentioned before, community detection methods can separate into two categories, one is based on calculation on such as edge betweenness, label propagation or some other calculations, the other one needs modularity.

From the Newman-Girvan modularity equation we can learn that if the value of the equation is large, it means that the partition we get is good. Because the more number of edges are within the community that more than the expected, the better the community we get. Therefore, we can imagine that we may get the best partition by maximizing the value of modularity.

The modularity-based techniques are mainly about different modularity maximization algorithms. The exhaustive maximization for the quality of community detection Q may not be possible, actually, the theoretical maximum value may not be able to get, and the maximization of modularity has been treated as an NP-complete problem by a lot of researches [30].

Recently, there are some algorithms apply to community detections based on modularity, they are: greedy algorithm, simulated annealing algorithm, external and spectral optimization algorithm and some other algorithms.

2.7.4.1 greedy techniques

Greedy algorithm is proposed by Newman [31] which in order to maximize the modularity.

Greedy algorithm begins with the network that contains n communities, and each node is in a community. At this initialization step, there is no edges exist. Edges are added during each step. The number of communities is n at first. Adding one edge will lead to decreasing one for the number of the communities. Therefore, a new community will appear.

Choosing which two nodes to connect the edge is the key point in this algorithm, because the edge added is in order to increase the modularity value comparing with the previous value of modularity. Do the same for the other edges, finally, the community detection result corresponding to the largest modularity value is the final result of the algorithm.

In recent studies, the CNM algorithm and CliqueMod algorithm are algorithms using the greedy techniques.

CNM algorithm is briefly described as follows:

1. We set n communities, and each community has only one node. Then, each time we will compute the difference value ΔQ when the edge joining two communities and if the communities have merged, the modularity value should raise.

2. If the difference value is the largest, merge the two communities. Carrying on compute the difference value ΔQ till we get the community that we expect.

The cliqueMod algorithm is based on the CNM algorithm, but has been improved from CNM.

The initial state of CNM algorithm is that all the communities are only with one node, in the CliqueMod, we can find cliques instead at the very first stage, which will lead to a better result. For the reason that CNM will be used in the late part of the article, the details about CNM algorithm will by discussed in later chapter.

At every step for the greedy algorithm, we need to calculate the value of the difference value ΔQ of the modularity, and this step's time complexity is only $O(m)$ for the reason that the calculation can be achieved in the constant time.

Recently, there are many other modification versions of the greedy techniques for modularity maximization proposed by different people, such as Danon, Wakita and Tsurumi and so on [32].

2.7.4.2 simulated annealing

Simulated annealing is the technique mainly about finding the global extreme value of the function, through using the probabilistic methods. The technique has been applied into different applications and fields.

2.7.4.3 spectral optimization

Spectral optimization of modularity is a method proposed by Newman using the eigenvalues of matrix [16].

As mentioned before, the modularity equation is:

$$Q = \frac{1}{2m}\sum_{ij}\left(A_{ij} - P_{ij}\right)\delta(C_i, C_j)$$

By importing the index vector s, the $\delta\left(C_i, C_j\right)$ function can be denoted as follow equation:

$$\delta\left(C_i, C_j\right) = \frac{1}{2}(s_i s_j + 1)$$

Therefore, the modularity value can be written as:

$$\begin{aligned} Q &= \frac{1}{4m}\sum_{ij}\left(A_{ij} - P_{ij}\right)\left(s_i s_j + 1\right) \\ &= \tfrac{1}{4m}\textstyle\sum_{ij}\left(A_{ij} - P_{ij}\right) s_i s_j \end{aligned}$$

The equation above can be written by applying it into matrix way:

$$Q = \frac{1}{4m} s^T B s$$

Element B is the modularity matrix, it contribute to the maximization of modularity.

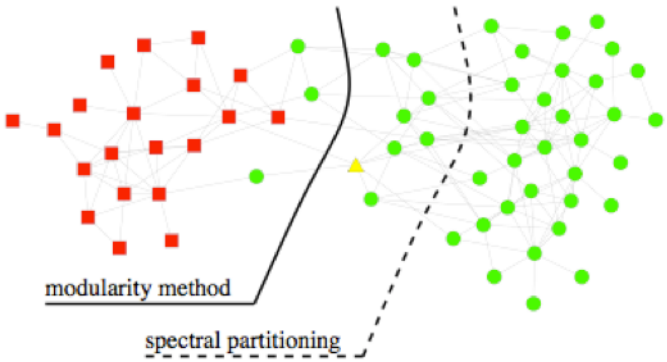


Fig. 31 comparing result by using different methods on dolphin social network ref. [16]

Fig. 31 is the dolphin social network which is a good example for comparing the method based on the modularity with the standard spectral partition method. The network is divided into two parts. For the standard spectral method, in the figure, the dotted line, it divided the network into two same size of partition. However, the result is not what we expected, in other words, it is not a good result in practical.

Obviously, the spectral optimization method did a better job. As the 62 nodes represent 62 dolphins are divided into two right communities, as the black solid line separated the network shows.

The other merit of the spectral optimization algorithm is that it is very fast because using the modularity matrix.

The extreme optimization, known as EO, is another algorithm based on modularity [33]. Actually, it relies on the local variables optimization.

2.7.4.5 modularity based methods conclusion

Different types of algorithms are applied to different applications in order to optimize modularity.

For those small networks (order of magnitude is less than 10^2), we use the simulated annealing. The simulated annealing algorithm is about seeking the optimum function in the maximum space.

Then, we make use of the spectral methods aim at optimizing modularity for the intermediate size (order of magnitude: 10^2-10^4). Spectral optimization is by introducing the eigenvalues, and uses the eigenvectors of the special matrix to optimize the modularity [16].

If the network is in large size (order of magnitude is above 10^4), then we may employ the greedy algorithms. The CNM algorithm mentioned before is also called greedy agglomerative algorithm. The greedy techniques' target is finding a way that may not be the best solution, but a timesaving and fast one.

2.7.5 Modularity modifications

Recently, there are many researches on modifications of modularity, different modifications are due to different community detection problems and different networks.

There are two modifications of modularity that are sometimes useful.

The first one is the modification of modularity applied to the weighted networks. The equation for this situation is as follows:

$$Q_w = \frac{1}{2W}\sum_{ij}\left(W_{ij} - \frac{s_i s_j}{2W}\right)\delta(C_i, C_j)$$

In the equation for weighted edges networks, $s_i s_j$ take the place of the $k_i k_j$ of the NG modularity, here, $s_i s_j$ is the strength of node i and node j. W is the total weights of all edges. Thus, we are able to get informed that the equation means that in the null model, the total weight of the network minus the expected weight between node i and node j.

The other modification of modularity equation is used for the directed networks. And the equation shows as follows:

$$Q_d = \frac{1}{m}\sum_{ij}\left(A_{ij} - \frac{k_i^{out} k_j^{in}}{2m}\right)\delta(C_i, C_j)$$

In this equation, we get to know that the direction of the edge relies on the $k_i^{out} k_j^{in}$, if the degree of the out is larger, for example, from node i to j, if in-degree is larger, the direction maybe j to node i.

The spatial null mode that will be used later may be treated as a kind of modification of modularity, either.

2.7.6 Modularity problems

Since modularity is not perfect, it meets problems.

The first problem is, modularity optimization suffers from the resolution limit. The resolution limit means that modularity optimization probably will stop the community detection step even the value of modularity is very small.

The failure for finding communities is according to the size or the degree of the network, what is more, even for those networks which communities are clearly defined. Not only in the artificial networks but also in some real networks, such as biological networks, the problem exists.

Or even for the cliques, which are defined clearly. Upon that, modularity maximization cannot find the communities smaller than the order of $\sqrt{m}$.

As a result, we are not able to clarify whether the community is a just a single community or a clique which has smaller communities inside it. This resolution limit has a great impact on the practical systems or applications [34], for the reason that in the real network, communities are in different size, thus, the communities with small size may not be found. At the same time, it may be even worse when it happens in social network or biology network, put the nodes into the wrong community or put the nodes into the same community which it intended not yield a bad result.

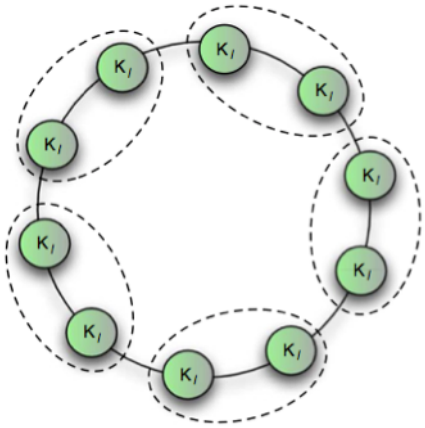


Fig. 32 modularity optimization resolution limit ref. [34]

As the fig. 32 shows, the optimization modularity method does not discover the natural structure of communities in the network, which the network made up by each clique.

The resolution limit origins from null model, by reason that in the null model every node has the same probability to connect with the other node. If apply this to the large network, the problem will apparently arise. And the resolution limit has had a big impact on weighted network as well according to lots of recent researches [35].

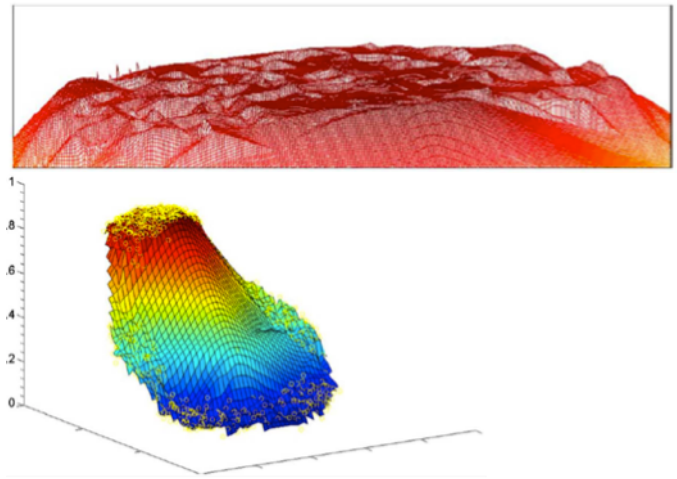


Fig. 33 modularity landscape ref. [36]

Another problem is that when the modularity value is very near the maximum (as shows in fig. 33), the network's different partitions will arrive at an exponential number, which make us impossible to find out the global maximum, especially for those real networks that possess hierarchical community structure.

2.8 Summary

This background Chapter gives the background knowledge about networks, the definition and properties on spatial network, community structure and the modularity and explains what has been done in this area.

At first, it depicts the different kinds of networks and the basic properties about networks. After that, it explains the different kinds of community structure and the widely used community detection methods. Then it gives the definition of modularity and the corresponding configuration null modes and describes how to calculate the modularity. Finally, it shows some modification version of the modularity and their problems.

Understanding the background knowledge help to know the experiment chapter mentioned following.

3 Experiments

For this project, the author constructed 2 experiments.

The first one is an experiment planning to test the Read-Keeling networks, apply the network to the normal Newman-Girvan modularity and compare with the new spatial modularity.

The second experiment aims to try the real spatial network, using the normal community detection method and the modified version of modularity maximization community detection method for spatial separately, and comparing the results.

3.1 Data

3.1.1 Read-Keeling network

For the first experiment, the network is a Read-Keeling network, in other words, is a computer generated contact network [37].

The Read-Keeling network originally is used for analyzing the disease revolution. As knowing how disease will evolve according to the situation is important to our public health. Thus, finding out how the disease will do to change the transmission route according to the infected individual and susceptible individual is of great importance. Therefore, the network is constructed to describe the disease's possibly transmission routes.

Network is generated using the graph generator, and the graph generator is able to generate two kinds of networks. One is the local network, and the other is the global network.

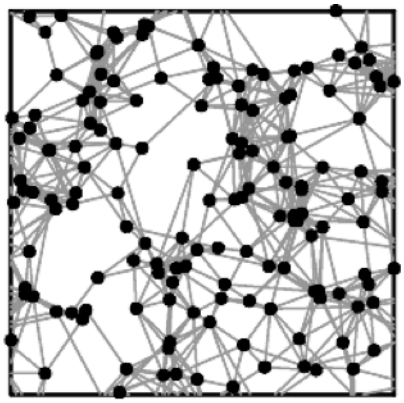


Fig. 34 local network ref. [37]

As shows in fig. 34, the small network, as we can see apparently, is highly-clustered, the nodes may come in the similar contact structure.

While, as for the global network, as we can see in fig. 35, the network is not clustered, however, it shows a obvious characteristic that the nodes are highly influenced by the long distance connections randomly.

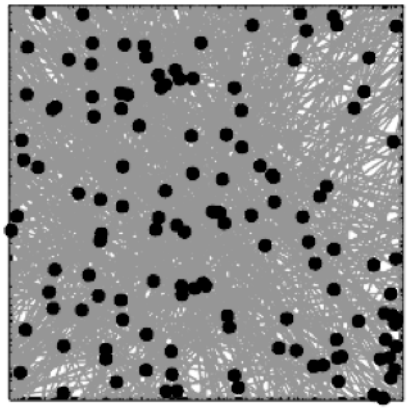


Fig. 35 global network ref. [37]

The network consists of N nodes and no matter what the size of network is, the nodes are evenly located in the dimensions of $\sqrt{N}\times\sqrt{N}$. And the possibility of two nodes that will connect to each other by a distance d is defined as connectivity kernel K.

The relationship between connectivity and distance is as follow equation:

$$K = p.\exp\left(\frac{-d^2}{2D^2}\right)$$

In the equation, D is the average value of two connected nodes; distance. When we set D=1, the graph generator will generate a local network. And when $D = 50$, the network is a global one.

As for the first experiment, the properties of the network that the author want to generate is as follows:

Number of nodes	D	Average degree needed	Dimensions
100	1	5.0	2

As a result, the graph generator generate a Reed-Keeling network with 100 nodes in a file named temp.txt. At the same time, it generated the positions of the nodes in the file called groups-temp.txt.

Actually, for the network file named after temp.txt, the important parameters is as follows:

Max node ID	Number of edges
99	95

And for the position file, in the file listed the node index and the positions of each node, thus, we can use it to calculate the distance between any two nodes.

3.1.2 Travel flow network

Network for the second experiment is a real spatial network, it origins from the website, the statistic is about the travel flow network. And the data has been processed.

The original network is a directed network, however, by replacing the edges like (u, v) and (v, u) by just the edge (u, v). And the weight is the sum of the original one.

The network is without those self-edges by deleting them from the original network. For example, people may live and work in Bristol, so their trips are self-edges, need to be deleted.

Along with the network file, there is an attributes of the node file named Places.txt, it contains the node index, the city's name and the coordinates of each node. The coordinate represent each node's value of meters of east, and north, with the coordinate the distance of any two nodes can be calculated.

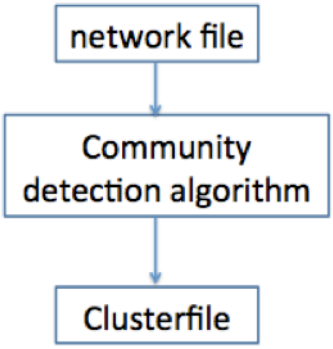
The important information of the travel flow network is as follows:

Max node ID	Number of edges	Number of nodes
379	1928	372

3.2 Methods for experiments

Speaking of the first experiment, firstly, after getting the Read-Keeling network, we should choose a community detection algorithm in order to get the cluster-file. Cluster-file is a file contains the communities after applying the community detection method, normally, in the form of txt file each line is a community.

For the first experiment, the method COPRA is chosen as the community detection algorithm, for the Read-Keeling network is un-weighted, non-overlapping and uni-partite, after applying the Read-Keeling network into the algorithm, we get a cluster-file with 76 communities. The figure bellow shows the structure of how community detection algorithm works.



When getting both the network file and cluster-file, it is able to calculate the modularity. First of all, we need to calculate the normally widely used Newman-Girvan modularity for calculation.

The steps for calculation can be described as follows:

1. Firstly, things need to be done is reading in the network file, getting the important knowledge of the network, such as node number, edge number and building up the adjacency matrix A_{ij} for the network.

2. Secondly, read in the cluster-file, then initialize an array for storing a symmetric matrix e with $k \times k$, k in this experiment is 76 thus, the matrix e is a 76×76 symmetric matrix. While, the element e_{ij} of the matrix e represent the proportion of nodes in community i which can connect to nodes in community j in the whole network.

3. Afterwards, calculate the sum $a_i = \sum_j e_{ij}$, a_i depicts the fraction of nodes outside community i connect to the community i of the network.

4. By using two loops, we are able to calculate the modularity as the equation shows:

$$Q = \sum_i (e_{ii} - a_i^2)$$

   And the equation can be described that the fraction of the edges inside a community minus the proportion of the edges expected outside the community.

Calculation for the spatial modularity is a little bit different from the Newman-Girvan modularity. The key point is that the attribute file also should be read in as to calculate the distance.

As discussed in the background chapter, the null mode is of great importance for the modularity calculation. As mentioned before, spatial null model includes those nonstructural effects, that is to say, distance plays an important role, we already acknowledge that the equation for the null mode is:

$$P_{ij}^{Spa} = N_i N_j f(d_{ij})$$

And also if the deterrence function equation is as follows:

$$f(d) = \frac{\sum_{i,j|d_{ij}=d} A_{ij}}{\sum_{i,j|d_{ij}=d} N_i N_j}$$

Therefore, we can get the result that as follows:

$$\sum_{i,j|d_{ij}=d} P_{ij}^{Spa} = \sum_{i,j|d_{ij}=d} A_{ij}$$

While, as a result, we can conclude that the weights between node i and node j is known at a value of distance d.

In the first experiment, the spatial null mode is as the equation described bellow:

$$P_{ij}^{Spa} = k_i k f(d_{ij})$$

It means that N_i is the node i's degree.

As for the second experiment, the initial part is similar with the first experiment. The network-file and attribute file should be read in and use different parameters to store the important values of the network.

The community detection algorithm for the second experiment contains then CNM algorithm and the modification of CNM algorithm for the spatial one.

As it mentioned in the background part, CNM is the algorithm based on modularity maximization.

In order to implement the CNM algorithm, different data structures should be built:

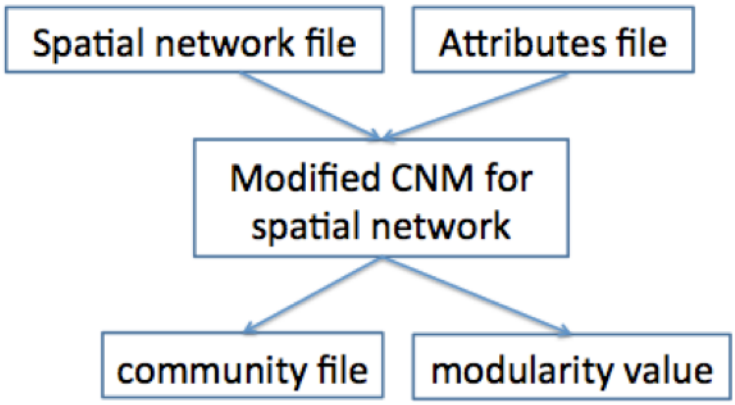
1. Firstly, an array-list (or others similar data structure) should be initialized to store the communities. At the first stage, each node is community.

2. Then, another array-list (or other similar data structure) should be defined to store and maintain the ΔQ_{ij} value.

3. Also, a hash-map data structure also needs to construct to store the max ΔQ_{ij} and the node index i and j.

The algorithm can be described as follows:

```
Algorithm for community detection(CD){
C = partition of communities, each nodes is in one community // initialization
Q = modularity //
While C is more than the number you desire {
 for all the pairs, compute the dQ{i,j} and find the largest
dQ{i,j}   //each time will calculate the modularity value
C = Merge(C, i , j)   // merge community i and j
Q += dQ{i, j}  // update Q value each time
If dQ{i,j}  > 0 {   // find the largest modularity community
maxC = C   //
maxQ = Q //
  }
 }
}
```



The figure above shows the structure of the modified CNM algorithm for spatial network.

At first, the desired number should be inserted which is the community number you wish to get.

As for the modularity or the delta modularity, the null mode is based on the spatial null mode just mentioned for the first experiment.

The details of the algorithm is as follows:

1. Initialization. The modularity value $Q = 0$, and set each node is in a single community. Initialize for the degree value k_i for each node and the value of m. And the definition of ΔQ_{ij} is as follows:

$$\Delta Q_{ij} = \begin{cases} \frac{1}{2m} - \frac{N_i N_j f(d_{ij})}{2m} & when\ i\ and\ j\ are\ connected \\ 0 & other\ situation \end{cases}$$

Obviously, ΔQ_{ij} is a sparse matrix.

2. Finding the maximum ΔQ_{ij} in the matrix's each row and store in the H.

3. Picking the max ΔQ_{ij} in the H, and then merge the community i to community j. After that, update the value of ΔQ, H. Along with that, calculate the modularity value Q every time.

4. Repeat the step 2 and 3, until getting the community number wanted.

4 Result discussion

4.1 widely used results analyzing methods

When the community detection algorithm and modularity calculation method have been chosen or designed. It is of necessity to test your algorithms, or sometimes called performance. There are different ways for testing methods.

4.1.1Comparing algorithms

Firstly, we can test the heuristics. That is to say, testing the different Q values yielded from different algorithms, such as the CNM algorithm, PL algorithm, WT algorithm or other algorithms. At the same time, we may test the time complexity as well for comparing.

4.1.2 Using empirical data

Secondly, testing on the empirical data of the real network for comparing is anther method.

As the example mentioned in the background chapter, we can visualize the network, and analysis the differences clearly from the figure we get. As the fig. 27 and fig. 28, we can apply the method into the comparison method that compare the NG modularity result with the spatial modularity way and to find whether the space will come to an effect.

4.1.3 Benchmarks

For the numerical validation, we can use benchmark, either.

Benchmark is about testing an algorithm by introducing a given problem that we have already known about the solution and applying the algorithm we want to test to the problem, then compare the results. On the other words, it is about building artificial networks that with the known community structure.

In most cases, we use the computer-generated graphs in the controlled settings [21]. There are many computer-generated benchmarks for us to choose, such as Girvan and Newman benchmark, LFR benchmark.

4.1.3.1 planted ℓ-partition model

Planted ℓ-partition model is a model aim at solving the graph bisection problem [22].

In the model, the graph with n nodes will be divided into ℓ groups. In the same group, vertices will connect with other vertices with a probability of p_{in}. If the vertices want to connect to other groups' vertices, the probability is p_{out}.

Thus, when $p_{in} > p_{out}$, the graph exists a community, for the reason that the density of edges in the community is larger than the density of outside edges.

If $p_{in} = p_{out}$, the group is a random graph. Thus the degree is the same for all the vertices:

$$\langle k \rangle = p_{in}(g-1) + p_{out}g(\ell - 1)$$

g is the number of vertices in a group.

4.1.3.2 Girvan and Newman benchmark

Girvan and Newman benchmark is regarded as a particular case of the Planted ℓ-partition model [13]. Using the Girvan and Newman benchmark, we should set $\ell = 4$, g=32, n=gℓ=32*4=128, $\langle k \rangle = 16$. The values of the expected inside and outside of degree are z_{in} , z_{out} respectively. And $z_{in} = p_{in}(g-1)$, $z_{out} = p_{out}g(\ell - 1)$.

When $z_{out} < 8$, it means that there exits a community and the community is also well structured, for the reason that the inside edges' density surpass the outside density.

Whereas, if $z_{out} = z_{in} = 8$, that depicts that $p_{in} \approx 4$, $p_{out} = \frac{1}{12}$, is not the threshold of the Planted ℓ-partition model. From the partition until $z_{out} \approx 12$, you are expected to detect the model [23].

Thus, testing the algorithm using the Girvan Newman benchmark, we should compute the similarity between the partitions, and calculate the partition of graph that partitioned by nature in four groups with equal size. Then repeat the steps to get different z_{out} values. Finally, we get the results and the average similarity is z_{out} function.

4.1.3.3 LFR benchmark

Owing to the reason that even though most of algorithms working properly when the value z_{out} is small, they will meet problem when z_{out} close to 8.

While LFR benchmark that introduced by Lancichinetti et al, configure that the distribution degree and the size of the community follows the power laws. The exponents are τ_1 and τ_2, respectively.

In order to use LFR benchmark to generate graphs, we assume that every node has a probability of $1-\mu$ to connect with the other nodes in the same community, and has a probability of μ to connect the nodes in the other

communities. μ is in the range of : $0 \leq \mu \leq 1$, μ is also called mixing parameter. k_i is the degree of the node i.

Thus, the steps of generating graphs are as follows [20]:

1. get the communities in order by picking numbers at random from the power law distribution that with exponent τ_2.
2. get the exponent τ_1 and let every node i has the internal degree of $k_i(1-\mu)$
3. set all the nodes connect to the other nodes at random by edges in the same community, until no more edges does not connect to nodes.
4. At the end, every node i are added μk_i edges, then link to nodes of different communities randomly, until all the edges are finished.

BY using LFR benchmark make it possible for us to create graphs with different orders of magnitude sizes and the graphs generated by LFR benchmark will be more complex.

The LFR benchmark for overlapping communities and for weighted networks that extended from LFR benchmark in use recently.

4.1.3.4 Benchmark built for spatial network

In the recently study, the paper [14] presents us a way to test the method's validation in a controlled setting, based on the computer benchmark, especially for spatial network.

The thought is to set up a network based spatial, that is to say that whether two nodes will be connected or not is relied on the distance and the community that the nodes belong to.

The benchmark is built by randomly choosing 100 nodes in the two-dimensional square, which dimension is: 100×100. After that, the 100 nodes are put into two communities evenly, 50 nodes in each community. What's more, the size of nodes is fixed. Then paper [14] set the values of the possibility of the node i can connect to node j is as follows:

$$P_{ij} = \frac{\lambda(c_i, c_j)}{Z d_{ij}}$$

In the equation, c_i means the community which node i in, so as for c_j. The lambda function $\lambda(c_i, c_j)$ depicts the discernible lever of the communities. Z is the constant of normalization which make sure that $\sum_{i>j} P_{ij} = 1$.

In the equation definition, if $c_i = c_j$, $\lambda = 1$. While if $c_i \neq c_j$, the definition of λ is $\lambda_{different}$. If $\lambda_{different} = 0$, all the nodes that connected are in the same community. If $\lambda_{different} = 1$, it means there only exists one community.

As we can see from the equation, there is an element d_{ij}, which gives us the information that the networks are influenced by the gravity model. Networks that we want are generated by setting L links. And the links are with the possibility P_{ij}. L is compute in the way below:

$$L = \rho N(N-1)/2$$

The rho in the equation has the meaning that: if $\rho \geq 0$, it means the links' density, and ρ plays the role in controlling the surging of finite size near the $L\rho_{ij}$.

After that, we can build the random model of realization and do the modularity optimization with different $\lambda_{different}$ and ρ. Then compare the different methods by using the normalized variation of information [25].

Meanwhile, the recently paper [14] shows us the way using statistical test to compare different methods. For example, compare the spatial modularity Q_{Spa} and the Girvan Newman modularity Q_{NG}. For the reason that directly compare Q_{NG} and Q_{Spa} is of none sense, thus, in the paper [14], two kinds of random networks are built to test them.

The first type of random networks is the one where weights are generated by random. Specifically, the weights are picked between the two communities i and j, and follows the mean of binomial, which value is: $\rho N_i N_j f(d_{ij})$.

The second type of networks is totally different from the first one, because the second network's weights are not fixed but the nodes are geographically random. In other words, the topology of this network keeps unchanged, the things that have changed are the attributes of nodes. What is more, the Q_{NG} will not be effected for the reason that NG modularity has nothing to do with the geographical information. So, we can change the function $f(d_{ij})$ to see the different results.

For each kind of the networks, at first, are generated 100 networks and then optimize the modularity of both NG and spatial. Then, compare the modularity that generated by spa or NG with the one yielding from the random network by using the z-score.

z-score is described as follows:

$$z = \frac{Q - < Q >_{random}}{\sigma}$$

In the equation, σ is a value of standard deviation getting from the 100 networks. Particularly, if the value of z-score is a negative one, it means that the significant information has been lost.

For sure, we can apply the normalized variation of information to compare the results.

4.2 results of experiment and conclusion

As for the first experiment, when testing on the Read-Keeling network, the normal modularity, that is, Newman-Girvan modularity will yield the result from 0 to 1 according to the experiments. And for the spatial modularity, the result is 0, sometime less than 0. For example, the Read-Keeling network with 89 nodes, and the max node id is 99, from 0 to 99 and 95 edges, along with that the average degree is 5.0. When apply the network into Newman-Girvan modularity calculation algorithm the modularity value is 0.102. And the spatial modularity yields the result 0.

Speaking of the second experiment, after applying the real spatial network that described in chapter 3.1, the partition that get from the spatial modularity maximization CNM algorithm has larger modularity than the that using the normal CNM version.

From the experiments and the results that yielded, the conclusion for the spatial networks may come as follows:

- The spatial networks normally are naturally clustered, using the standard method may not find any underlining information.

- When taking the spatial effects in account, it will affect the nodes' connectivity.

- Adding the non-structural information into the definition of the modularity may discover some unknown attributes

5. Future works

Firstly, the result are not showing in the figure or using the software like matlab to illustrate and visualize the network. In the future, the results from the network should be clearly shows the difference between the different algorithms.

Secondly, the algorithms' time complex or the code can be more optimized. And the tests may applied on different algorithms and get a better look at the differences, the comparison may include the time costs for different networks and different algorithms.

Finally, there are some new researches on calculation the spatial modularity, some new null model is proposed, thus, these new approaches may apply to the experiments as to see the results.

6. References

[1] M.E.J.Newman,A.-L. Barabasi and D.J.Watts, The Structure and Dynamics of Networks (Princeton University Press, Princeton, 2006).

[2] N.E.J.Newman, The structure and function of complex networks, 2003.

[3] Wasserman, S. & Faust, K. (1994) Social Network Analysis (Cambridge University Press, Cambridge, U.K.).

[4] D.L.Nelson, C.L. McEvoy, T.A. Schreiber, The university of south florida word association, rhyme, and word fragment norms (1998).

[5] R. Albert, H. Jeong, A.-L. Barabási, Internet: Diameter of the world-wide web, Nature 401 (1999) 130–131.

[6] M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E 69 (2) (2004) 026113.

[7] E. Bullmore and O. Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. Nature Reviews Neuroscience, 10:186 – 198, 2009.

[8] M. Barthelemy (2010) Spatial Networks, arXiv:1010.0302.

[9] J. Clark and D. A. Holton. A first look at graph theory. World Scientific, 1991.

[10] P. Erdös, A. Rényi, On random graphs. I., Publ. Math. Debrecen 6 (1959) 290–297.

[11] C. Jensen-Butler (1972) Gravity Models as Planning Tools: A Review of Theoretical and Operational Problems, Geografiska Annaler. Series B, Human Geography 54, pp. 68-78.

[12] R. Lambiotte, V.D. Blondel, C. de Kerchove, E. Huens, C. Prieur, Z. Smoreda and P. Van Dooren (2008) Geographical dispersal of mobile communication networks, Physica A 387, 5317-5325.

[13] M. Girvan, M.E.J. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. USA 99 (12) (2002) 7821–7826.

[14] Paul Expert, Tim Evans, Vincent D. Blondel and Renaud Lambiotte (2010). Beyond Space For Spatial Networks. Arxiv preprint arXiv10123409, 1-8.

[15] G. Palla, I. Derényi, I. Farkas, T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society, Nature 435 (2005) 814–818.

[16] M. E. J. Newman (2006) Finding community structure in networks using the eigenvectors of matrices, Phys. Rev. E 74, 036104.

[17] A. Clauset, M.E.J. Newman, C. Moore, Finding community structure in very large networks, Phys. Rev. E 70 (6) (2004) 066111.

[18] Santo Fortunato, community detection in graphs, Phys. Rep. 486 (3-5): 75–174 doi:10.1016/j.physrep.2009.11.002.

[19] Béla Bollobás, Random Graphs, 2nd Edition, 2001, Cambridge University Press.

[20] Newman, M. E. J. (2010). Networks: An Introduction. Oxford.

[21] M. E. J. Newman and M. Girvan, in Statistical Mechanics of Complex Networks, R. Pastor-Satorras, J. Rubi, and A. Diaz-Guilera (eds.), Mixing patterns and community structure in networks, Springer, Berlin (2003).

[22] G. Palla, A.-L. Barabási, T. Vicsek, Nature 446:7136, 664-667 (2007)

[23] L.C. Freeman, A set of measures of centrality based on betweenness, Sociometry 40 (1977) 35–41.

[24] S. Gregory, An algorithm to find overlapping community structure in networks, in: Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD 2007, Springer-Verlag, Berlin, Germany, 2007, pp. 91–102.

[25] Usha Nandini Raghavan, Reka Albert, Soundar Kumara, Near linear time algorithm to detect community structures in large-scale networks, Physical Review E 76, 036106 (2007).

[26] Modularity and community structure in networks, M. E. J. Newman, Proc. Natl. Acad. Sci. USA 103, 8577–8582 (2006).

[27] J.R. Tyler, D.M. Wilkinson, B.A. Huberman, Email as spectroscopy: Automated discovery of community structure within organizations, in: Communities and Technologies, Kluwer, B.V., Deventer, The Netherlands, 2003, pp. 81–96.

[28] Steve Gregory, Finding overlapping communities in networks by label propagation, New J. Phys. 12 103018 (2010).

[29] V.D. Blondel, G. Krings and I. Thomas (2010) Regions and borders of mobile telephony in Belgium and in the Brussels metropolitan zone, Brussels Studies 42, 4.

[30] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikolski, D. Wagner, On modularity — np-completeness and beyond. URL http://digbib. ubka.uni- karlsruhe.de/volltexte/documents/3255.

[31] M.E.J. Newman, Fast algorithm for detecting community structure in networks, Phys. Rev. E 69 (6) (2004) 066133.

[32] L. Danon, A. Díaz-Guilera, A. Arenas, The effect of size heterogeneity on community identification in complex networks, J. Stat. Mech. 11 (2006) 10.

[33] S. Boettcher, A.G. Percus, Optimization with extremal dynamics, Phys. Rev. Lett. 86 (2001) 5211–5214.

[34] S. Fortunato, M. Barthélemy, Resolution limit in community detection, Proc. Natl. Acad. Sci. USA 104 (2007) 36–41.

[35]J.W.Berry,B.Hendrickson, R.A.LaViolette, C.A.Phillips, Tolerating the community detection resolution limit with edge weighting, eprintarXiv:0903. 1072.

[36] B.H. Good, Y. de Montjoye, A. Clauset, The performance of modularity maximization in practical contexts, eprint arXiv:0910.0165.

[37] Read and Keeling (2003) Disease evolution on networks: the role of contact structure Proc. Roy. Soc. Lond. B 270 699-708

7. Appendix: essential code

Part of the code of calculation of the NG modularity:

```
private static double modularity(int nClusters, List<Integer> vertexCluster,
                                 List<Pair> edgesI)
  {
   int i, j, c1, c2, a;
   double d;
   int m = 0;   // # edges
   Pair edge;
   int[][] e = new int[nClusters][nClusters];
   for (i=0; i<nClusters; i++) {
     for (j=0; j<nClusters; j++) {
       e[i][j] = 0;
     }
   }
   System.out.println(edgesI);
   Iterator<Pair> it = edgesI.iterator();
   while (it.hasNext()) {
     edge = it.next();
     c1 = vertexCluster.get(edge.value1);
     c2 = vertexCluster.get(edge.value2);
     e[c1][c2]++;
     //System.out.println(c1+" "+c2);
     e[c2][c1]++;
     m++;
   }

   double m2 = m + m;
   double mod = 0.0;
   for (i=0; i<nClusters; i++) {
     a = 0;
     for (j=0; j<nClusters; j++) {
       a += e[i][j];
       //System.out.println(e[i][j]);
     }
     //System.out.println(a);
     d = (double)a;
     mod += ((double)e[i][i] - d*d/m2)/m2;
   }
   return mod;
  }
```

Part of the code of calculation of spatial modularity:

```
public class SpatialModularity
{
 static int maxVertex = 0;
 static double position[][]; //store the position of i, j nodes
```

```
public static double modularity(String graphFile, String clustersFile)
{
 int i, j;
 String u, v;
 Iterator<String> it, it1;
 ArrayList<Pair> edgesI = new ArrayList<Pair>();
 HashMap<String,Integer> vertexNum = new HashMap<String,Integer>();
 HashMap<String,HashSet<String>> graph =
  CONGA.readGraphEdges(graphFile, null, false, null);
 it = graph.keySet().iterator();
 while (it.hasNext()) {
  u = it.next();
  i = numVertex(u, vertexNum);
  it1 = graph.get(u).iterator();
  while (it1.hasNext()) {
   v = it1.next();
   j = numVertex(v, vertexNum);
   if (i < j) {
    edgesI.add(new Pair(i, j));
   }
  }
 }

 ArrayList<Integer> vertexCluster = new ArrayList<Integer>();
 int k = readClusters(vertexCluster, clustersFile, vertexNum);

 return modularity(k, vertexCluster, edgesI);
}

private static double modularity(int nClusters, List<Integer> vertexCluster,
                                 List<Pair> edgesI)
{
 int i, j, c1, c2, a;
 double d;
 int m = 0;   // # edges
 Pair edge;
 int[][] e = new int[nClusters][nClusters];
 for (i=0; i<nClusters; i++) {
  for (j=0; j<nClusters; j++) {
   e[i][j] = 0;
  }
 }
 Iterator<Pair> it = edgesI.iterator();
 while (it.hasNext()) {
  edge = it.next();
  //System.out.println(edgesI);
  c1 = vertexCluster.get(edge.value1);
  c2 = vertexCluster.get(edge.value2);
  //System.out.println(c1+"  "+c2);
  e[c1][c2]++;
  e[c2][c1]++;
  m++;
 }
```

```
  double m2 = m + m;
  double mod = 0.0;
          double dist = 0.0;
  for (i=0; i<nClusters; i++) {
   a = 0;
   for (j=0; j<nClusters; j++) {
    a += e[i][j];
    //System.out.println(e[i][j]);
   }
   //System.out.println(i+" "+j);
   d = (double)a;
   mod += ((double)e[i][i] - d*d/m2)/m2;
   System.out.println(mod);
  }
  return mod;
}

private static int readClusters(List<Integer> vertexCluster,
                        String filename,
                        HashMap<String,Integer> vertexNum)
{
  int i, j, len, start;
  int k=0;
  String line, v;
  String[] words = null;
  HashSet<Integer> cluster;
  try {
   BufferedReader in = new BufferedReader(new FileReader(filename));
   while ((line = in.readLine()) != null && !line.equals("")) {
    words = line.split(" ");
    len = Array.getLength(words);
    if (words[0].endsWith(":")) {
     start = 1;
    }
    else {
     start = 0;
    }
    for (i=start; i<len; i++) {
     v = words[i];
     j = numVertex(v, vertexNum);
     while (vertexCluster.size() <= j) {
      vertexCluster.add(-1);
     }
     vertexCluster.set(j, k);
    }
    k++;
   }
  } catch (Exception e) {
   System.out.println("Clusters/groups file error: "+e.toString());
   System.exit(1);
  }
  //System.out.println(vertexCluster);
  //System.out.println(vertexNum);
  return k;
```

```
 }

 private static int numVertex(String v, HashMap<String,Integer> vertexNum)
 {
  int j;
  if (vertexNum.containsKey(v)) {
   j = vertexNum.get(v);
  }
  else {
   j = maxVertex++;
   vertexNum.put(v, j);
  }

  return j;
 }

 static void readAttributes(String filename) throws FileNotFoundException{
                        int size = 0;
                        ArrayList<HashSet<Integer>> pos = new
ArrayList<HashSet<Integer>>();
                        File filepath = new File(filename);
                        if (filepath == null){
                                throw new IllegalArgumentException("File should not be
null.");
                        }
                        if (!filepath.exists()){
                           throw new IllegalArgumentException("File does not exist:
"+filepath);
                        }

                        Scanner scanFile = new Scanner(filepath);
                        try{
                                while (scanFile.hasNextLine()){
                           //processLine(scanFile.nextLine());
                              String oneLine = scanFile.nextLine();
                              Scanner scanLine = new Scanner(oneLine);
                              if (scanLine.hasNext()){
                                      size++;
                              }
                              else{
                        }
                        scanLine.close();
                     }
                }
                finally {
                        scanFile.close();
                }

                for (int i=0; i<size; i++) {
                                pos.add(new HashSet<Integer>());   //store in the
hashset
                           }

                        position = new double[size][2];
```

```
                    for (int i=0; i<size; i++) {
                     for (int j=0; j<2; j++) {
                       position[i][j] = 0;
                     }
                    }

                        Scanner scanFile2 = new Scanner(filepath);
                        try{
                                int i = 0;
                                int nodes1 = 0;
                                double nodes2 = 0.0;
                                double nodes3 = 0;
                                String del = "";
                          while (scanFile2.hasNextLine()){
                          //processLine(scanFile.nextLine());
                            String oneLine = scanFile2.nextLine();
                            Scanner scanLine = new Scanner(oneLine);
                            int j = 0;
                            if (scanLine.hasNext()){
                                    nodes1 = Integer.parseInt(scanLine.next());
                                    nodes2 =
Double.parseDouble((scanLine.useDelimiter(",")).next());
                                    nodes3 = Double.parseDouble(scanLine.next());
                              i = nodes1;
                              position[i][j] = nodes2;
                              j++;
                                        position[i][j] = nodes3;
                                        //System.out.println(position[i][j]);
                            }
                            else{
                          }
                          scanLine.close();
                        }
                       }
                    finally{
                      scanFile.close();
                    }
 }

 private static double distance(int i, int j){
                double x1 = position[i][0];
            double y1 = position[i][1];
            double x2 = position[j][0];
            double y2 = position[j][1];
            double result = Math.sqrt((x1-x2)*(x1-x2) + (y1-y2)*(y1-y2));
            return result;
 }
}
```