Community Structure and Detection in Complex Networks: Detailed Annotated Bibliography and Classification of the Results

Guoliang Liu

CSC8350 Literature Review (Due day: 31/03/2012)

Instructor: Dr. Sushil Prasad

This paper is a fundamental one for community detection with respect to the detection principles. Modularity as one of the most used principles to divide a network to communities was first introduced and formalized. In this paper, at first, an introduction of community detection was introduced, which mentioned that the study of community structure in networks has a long history. From a wide perspective of understanding, Community detection is partitioning in graph theory or clustering in sociology, which has been studied for quit a long time and are still interesting topics during current years. Two kinds of common methods including agglomerative and divisive ones are introduced. This paper focuses on the divisive methods. The algorithm is as below.

1. Calculate betweenness scores for all edges in the network.
2. Find the edge with the highest score and remove it from the network.
3. Recalculate betweenness for all remaining edges.
4. Repeat from step 2.

In fact, this algorithm is later used, modified and improved in later works related with community detection and community structure analysis. I.e., this algorithm introduces the basic idea of divisive methods considering betweenness. In the following parts, implementation is talked about. In the above algorithm, the operation of removing the edge with highest score is not hard and not computationally demanding. Instead, calculating betweenness scores is more crucial. Three betweenness measures are talked about.

A. Shortest-path betweenness.
B. Resistor networks.
C. Random walks.

In the next part, the most famous principle called modularity to qualify the strength of
community structure is introduced and formalized.

\[ Q = \sum_i (e_{ii} - a_i^2) = Tr e - ||e^2|| \]

The trace of this matrix \( Tr e = \sum_i e_{ii} \) gives the fraction of edges in the network that connect vertices in the same community, and clearly a good division into communities should have a high value of this trace. The trace on its own, however, is not a good indicator of the quality of the division since, for example, placing all vertices in a single community would give the maximal value of \( Tr e = 1 \) while giving no information about community structure at all.

In the end, the algorithm using the modularity as the metrics is applied on computer-generated networks, famous Zachary’s karate club networks, collaboration networks and some other online social networks.

In this paper, the author mentions Pareto principle, e.g., 80 percent of a country’s land is owned by 20 percent of the population. 80 percent of a company’s income is achieved by 20 percent of its clients. Less than 1 percent of the Twitter users (e.g. entertainers, politicians, writers) produce 50 percent of its content, while the others (e.g. fans, followers, readers) have much less influence and completely different social behavior. As a assumption, In many social networks, there exist two types of users that exhibit different influence and different behavior.

At first, the authors give the definitions of Community Kernel, Auxiliary Community and unbalanced Weakly-Bipartite Structure
This paper proposes a simple heuristic method to extract the community structure of large networks. And the quality of the communities detected is measured by modularity in [7].

The algorithm is divided into two phases which are repeated iteratively. In the first phase, each node is treated as a community. Then, for each node i we consider the neighbors j of i and we evaluate the gain of modularity that would take place by removing i from its community and by placing it in the community of j. The node i is then placed in the community for which this gain is maximum (in case of a tie we use a breaking rule), but only if this gain is positive. The gain in modularity obtained by moving an isolated node i into a community C can easily be computed by:

$$\Delta Q = \left[ \frac{\sum \text{in} + k_{i,in}}{2m} - \left( \frac{\sum \text{tot} + k_i}{2m} \right)^2 \right] - \left[ \frac{\sum \text{in}}{2m} - \left( \frac{\sum \text{tot}}{2m} \right)^2 - \left( \frac{k_i}{2m} \right)^2 \right]$$

The second phase of the algorithm consists in building a new network whose nodes are now the communities found during the first phase. Once this second phase is completed, it is then possible to reapply the first phase of the algorithm to the resulting weighted network and to iterate.

The algorithm is applied on a number of test-case networks.
The algorithm is based on the idea of modularity. Given any network, the GN community structure algorithm always produces some division of the vertices into communities, regardless of whether the network has any natural such division. To test whether a particular division is meaningful we define a quality function or modularity $Q$ as introduced in section 1.2.2.

If a high value of $Q$ represents a good community division, simply optimizing $Q$ over all possible divisions to find the best one should be a way to the goal, we can avoid the iterative removal of edges and cut straight to the chase. The problem is that true optimization of $Q$ is very costly. The number of ways to divide $n$ vertices into $g$ non-empty groups is given by the Stirling number of the second kind $S_n^{(g)}$, and hence the number of distinct community divisions $\sum_{g=1}^{n} S_n^{(g)}$. This sum is not known in closed form, but we observe that $S_n^{(1)} + S_n^{(2)} = 2^{n-1}$ for all $n > 1$, so that the sum must increase at least exponentially in $n$. To carry out an exhaustive search of all possible divisions for the optimal value of $Q$ would therefore take at least an exponential amount of time, and is in practice infeasible for systems larger than twenty or thirty vertices. Various approximate optimization methods are available: simulated annealing, genetic algorithms, and so forth. Here Newman et al, consider a scheme based on a standard greedy optimization algorithm, which appears to perform well.
Modularity has been used to compare the quality of the partitions obtained by different methods, but also as an objective function to optimize. Unfortunately, exact modularity optimization is a problem that is computationally hard and so approximation algorithms are necessary when dealing with large networks. The fastest approximation algorithm for optimizing modularity on large networks was proposed by Clauset, Newman et al., 2004. That method consists in recurrently merging communities that optimize the production of modularity. Unfortunately, this greedy algorithm may produce values of modularity that are significantly lower than what can be found by using. Moreover, the method proposed by Clauset, Newman et al., 2004 has a tendency to produce super-communities that contain a large fraction of the nodes, even on synthetic networks that have no significant community structure. This artefact also has the disadvantage to slow down the algorithm considerably and makes it inapplicable to networks of more than a million nodes. This undesired effect has been circumvented by introducing tricks in order to balance the size of the communities being merged, thereby speeding up the running time and making it possible to deal with networks that have a few million nodes.

The algorithm is divided in two phases that are repeated iteratively. Assume that we start with a weighted network of N nodes. First, we assign a different community to each node of the network. So, in this initial partition there are as many communities as there are nodes. Then, for each node i we consider the neighbors j of i and we evaluate the gain of modularity that would take place by removing i from its community and by placing it in the community of j. The node i is then placed in the community for which this gain is maximum (in case of a tie we use a breaking rule), but only if this gain is positive. If no positive gain is possible, i stays in its original community. This process is applied repeatedly and sequentially for all
nodes until no further improvement can be achieved and the first phase is then complete. Let us insist on the fact that a node may be, and often is, considered several times. This first phase stops when a local maxima of the modularity is attained, i.e., when no individual move can improve the modularity. One should also note that the output of the algorithm depends on the order in which the nodes are considered. Preliminary results on several test cases seem to indicate that the ordering of the nodes does not have a significant influence on the modularity that is obtained. However, the ordering can influence the computation time. The problem of choosing an order is thus worth studying since it could give good heuristics to enhance the computation time. The second phase of the algorithm consists in building a new network whose nodes are now the communities found during the first phase. To do so, the weights of the links between the new nodes are given by the sum of the weight of the links between nodes in the corresponding two communities. Links between nodes of the same community lead to self-loops for this community in the new network. Once this second phase is completed, it is then possible to reapply the first phase of the algorithm to the resulting weighted network and to iterate. Let us denote by pass a combination of these two phases. By construction, the number of meta-communities decreases at each pass, and as a consequence most of the computing time is used in the first pass. The passes are iterated (see Fig.4) until there are no more changes and a maximum of modularity is attained. The algorithm is reminiscent of the self-similar nature of complex networks and naturally incorporates a notion of hierarchy, as communities of communities are built during the process. The height of the hierarchy that is constructed is determined by the number of passes and is generally a small number.
These are relatively new approaches according to which some heuristics are applied in order to build communities directly without following a hierarchical approach as both agglomerative and separative approaches do. One example is the work presented in K. Steinhaeuser et al., 2010, where authors propose an algorithm that weights links in a complex network based on some node similarity metrics allowing then to identify communities by removing links whose weights are under a given threshold. Algorithms proposed in R. R. Khorasgani et al., 2010 and D. Shah and T. Zaman, 2010, where communities are built around special nodes called also leaders. However in both algorithms each community is centered around one leader only and a node can follow one community at once. In addition the first algorithm requires the user to provide the number k of communities to identify. Next we will talk about a more representative paper (Rushed kanawati, 2011) using constructive method and which is the modifications of the previous work that has been mentioned above.

The basic idea underlaying the proposed algorithm is that a community is composed of two types of nodes: Leaders and Followers. Roughly speaking, leaders form a subset of nodes (eventually one node) whose removal form the network implies community collapse. The algorithm functions are as follows. First it searches for nodes in the network that are likely to be leaders in a community. Different node ranking metrics can be used in order to estimate the role of a node. These include the classical centrality metrics. Let L be the set of identified leaders. The list L is then reduced by grouping leaders that are estimated to be in the same community. Let C be the set of identified communities. Each node in the network (a leader or a follower) computes its membership degree to each community in C. A ranked list of communities can then be obtained, for each node, where communities with highest membership degree are ranked first. Next, each node will adjust its community
membership preference list by merging this with preference lists of its direct neighbors in the network. Different strategies borrowed from the social choice theory can be applied here to merge the different preference lists. This step is iterated until stabilization of obtained ranked lists at each node. The convergence towards a stable state is a function of the applied voting scheme. Lastly, each node will be assigned to top ranked communities in its final obtained membership preference list.
To mathematically link the power-law topology and the community structure property, they need to select among the wide variety of definitions for communities a quantifiable measurement. The well-known Newmans modularity shall be selected as the only qualification for finding community structure in this paper. Since the introduction of modularity, maximizing modularity has become primal approaches to detect community structure. Despite that existing methods can find sub-optimal solutions in a reasonably fast time, they do not come with any provable solution quality, e.g. the solution is optimal up to a constant factor. Thus, it is desirable to design approximation algorithms to find community structure in the network. Their contribution in this paper is two-fold. They show that the community structure property can be seen as the consequence of the power-law topology in scale-free networks. In addition, we provide an approximation algorithm to find community structure. The algorithm is optimal up to a constant factor when the parameter $\beta > 2$ and a log n factor when $1 < \beta \leq 2$ in term of maximizing the modularity. It is the first approximation algorithm for the modularity maximization problem. The algorithm encourages further explorations to design both theoretically and empirically justified methods for finding community structure.

First, My T.Thai et al. proposed a following algorithm which belongs to constructive algorithm which is introduced in the previous part. Then a series of analysis is given. They make a detour to focus on the problem of modularity maximization in division of the network into just two communities. The maximum modularity value of the division into two communities are shown to be close to the best possible modularity. Thus, an approximation algorithm for the division into two communities problem also yields an approximation algorithm for the
modularity maximization problem.
Andrea Lancichinetti and Santo Fortunato, "Community detection algorithms: a comparative analysis", physics, 2009. Compared with GN benchmark, there is another benchmark which takes self-similarity, which is one property of complex networks, into account. The LFR benchmark is a special case of the planted \(l\)-partition model, in which groups are of different sizes and nodes have different degrees. The node degrees are distributed according to a power law with exponent \(\tau_1\); the community sizes also obey a power law distribution, with exponent \(\tau_2\). In the following, \(N\) indicates the number of nodes of the network. In the construction of the benchmark graphs, each node receives its degree once and for all, and keeps it fixed until the end. In this way, the two parameters \(p_{in}\) and \(p_{out}\) of the planted \(l\)-partition model in this case are not independent. Once the value of \(p_{in}\) is set one obtains the value of \(p_{out}\) and viceversa. It is more practical to choose as independent parameter the mixing parameter \(\mu\), which expresses the ratio between the external degree of a node with respect to its community and the total degree of the node. Of course, in general one may take different values for the mixing parameter for different nodes, but we will assume, for simplicity, that \(\mu\) is the same for all nodes, consistently with the standard hypotheses of the planted \(l\)-partition model. By construction, the groups are communities when \(p_{in} > p_{out}\). This condition can be translated into a condition on the mixing parameter \(\mu\). Let us label \(k_{in} i\) and \(k_{out} i\) the internal and external degree of node \(i\) with respect to its community (which we denote with \(c\)). By definition, \(k_{in} i\) is the number of neighbors of \(i\) that belong to its community \(c\) and \(k_{out} i\) the number of neighbors of \(i\) that belong to the other communities. The number of available connections \(k_{out}{c} (k_{in}{c})\) outside (inside) \(c\) is given by the sum of the degrees of the nodes outside (inside) the community. If the numbers of nodes inside and outside \(c\) are not too small, the sum of their degrees can be approximated by the product of the average degree \(\langle k \rangle\) by the number of nodes. We indicate with \(n_c\) the number of nodes of the community \(c\) of node \(i\), so we have that \(k_{out}{c} \sim (N - n_c)\langle k \rangle\) and \(k_{in}{c} \sim (n_c)\langle k \rangle\). By
definition of the linking probabilities $p_{in}$ and $p_{out}$ we deduce that

$$P_{out} = \frac{k_{out}^{i}}{k_{out}} = \frac{k_{out}^{i}}{(N - n_{c}) \langle k \rangle}$$

and

$$P_{in} = \frac{k_{in}^{i}}{k_{in}} = \frac{k_{in}^{i}}{n_{c} \langle k \rangle}$$

In this way, the condition for the existence of communities $p_{in}$ i. $p_{out}$ becomes

$$\frac{k_{in}^{i}}{n_{c} \langle k \rangle} > \frac{k_{out}^{i}}{(N - n_{c}) \langle k \rangle}$$

At last, we can get $\mu$ by

$$\mu < \frac{N - n_{c}}{N}$$