The Theory and Implementation of Infomap

Linzi Xing
Computer Science Department
University of Colorado at Boulder

I. INTRODUCTION

In 2007, Rosvall, Bergstrom and some other researchers proposed a brand new way to solve community detection problem[1]. Before their method, tasks of community detection always just focused on the links and edges of a network separately. It worked well, but no one treated the whole network as a system or map and had eye on the connection of interaction between nodes in network. Rosvall and Bergstrom tried to transfer the community discover problem into how to compress and encode the topological information of a network in the most efficient way to lose the least information and can get the most similar network compared to the original one after decoding the compression of network[1]. This process is vague if just described by words, so i made a figure named **FIG 1** to show the process more clear. **FIG 1** shows in the next page.

FIG 1 shows a process of compressing and sending information of network[1], from the figure[1], we can see clearly that here we have a signaler. As Rosvall, Bergstrom described in [1], "this person have the whole network structure and topological information. His purpose is to send the network information he knows to a receiver. However, because of limited capacity of the sending channel, he has to compress the network in an efficient way."[1] In here, "to maximize information which is sent, the signaler compresses the network into 2 modules which are grey and black in **FIG 1**"[1]. We can see the encoded network only contains some basic information like links and nodes inside each modules and links between pair of modules. However, some detailed information like which pair of nodes has link is lost. Then, according to Rosvall, Bergstrom, "receiver gets the signal from sending channel and tries to decode it. Finally, we can see he will get a set of candidates of networks."[1] they have the same number of links and nodes, but the structures are different. Smaller the size of candidates means more information has been compressed[1]. But for all possible module divisions, how we evaluate the similarity between them and the true network and pick the most similar one? We can calculate the mutual information[1] between decoded networks and real one. Then we choose the one maximizes mutual information. Like in **FIG 1**, we separate network into 2 modules, but we can also separate it in other number of modules. If we separate the true network into different number of modules, we will get totally different decoded networks and they can be described like this[1]:

$$B = \{\mathbf{m}, \mathbf{L}\}$$

where m is a vector contains the separation of modules and L represents the links information of the decoded network. For example, if for nodes 1,2, nodes 1,3 and nodes 2,3 there are links between them, L will contain l_{12},l_{13},l_{23}.

After calculation of all the possible separations of modules, we can get the one division with most mutual information. This is the best way we can use to compress the network into modules and keep the most structure information. This process shows a concept that the structure of community of a network can be seen through the information flow or connection inside the network and we can get the detection result as our community prediction as a way of mining data and getting some rules for networks[1].

The process above is just a very basic concept of Infomap[3] and after reading some papers[1–4] related to the whole concept and process of Infomap algorithm, the structure and system of Infomap becomes much clear to me. The whole algorithm can be separated into three steps. First, we should simulate random walker to get the flow of network map[2]. One important thing is to obtain the visit frequency for each node in network and some node ergodic algorithms like PageRank[15] will be very efficient for the process, also, to avoid the scenario that random walker is trapped in one community because not all networks are strongly connected, for some of them, they may have some components do not have any link between them[2]. We should also add tricks like teleportation probability[2] for all the nodes. So, in this way, we can make sure the visit frequency we get after random walk can best describe the whole network structure. There are several different teleportation processes[2] can be chosen and some have really great effect on result. The second step, which is also the most significant step, is defining a map equation[4] to compute the average codelength we shall use for each step. We can understand this through encoding in two levels of network with Huffman coding[16], these two levels are module level and node level[3]. To describe random walker's path, we count module codename when leaving the module. but when it comes to detailed computation, entropy[5] will be more efficient. Third, using greedy algorithm[6] to merge nodes in the network and finally heading to the global optimal of minimum of network's

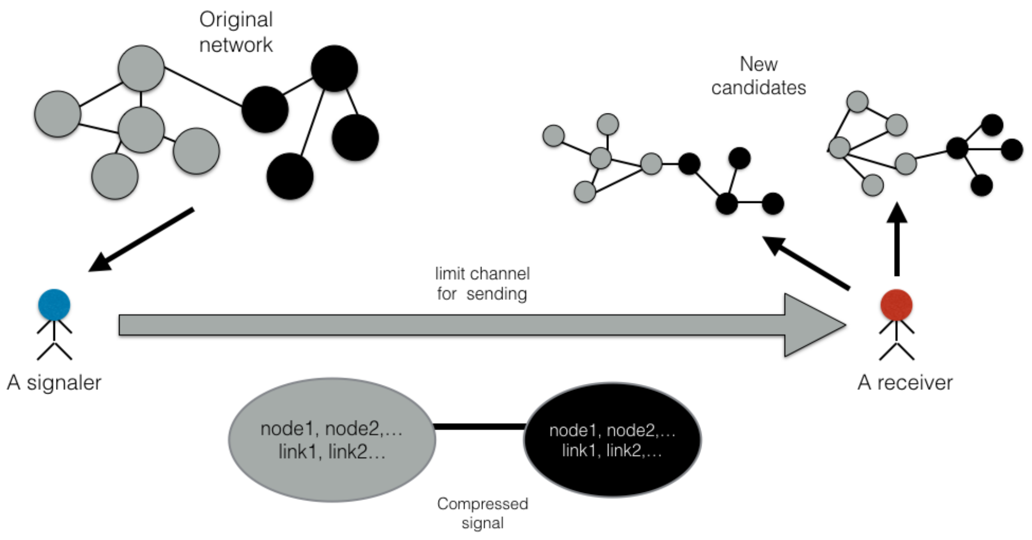
Original network
New candidates
limit channel for sending
A signaler
A receiver
node1, node2,… link1, link2…
node1, node2,… link1, link2,…
Compressed signal

FIG. 1. **The process of network compression** This figure was created based on concept in paper[1](Martin Rosvall, Carl T. Bergstrom, An information-theoretic framework for resolving community structure in complex networks, PNAS, 2007). The person in blue represents a signaler. He knows true network and compresses into signal based on module separation. Then sending through a pipe and receive by a receiver in red head. After receiver decodes signal, he can get several networks with the same number of nodes and edges but different structures[1].

codelength. In the initial condition, we just treat every single node as a community, then compute the codelength for merging each possible communities pair and merge the pair which has the shortest codelength until it finally converges to stable condition[6].

All above is the brief instruction of Infomap community detection algorithm[3]. I also practice Infomap on some datasets and compare with modularity[6] which is proposed by Newman and colleagues, it can be proven that first, my version Infomap can solve community detection problem in some kinds of levels and secondly, in some cases, Infomap can show better performance than modularity. Different from some other teams' works whose point is collecting and processing data in network formation and using exist algorithm to test and find something new, mine focused on reproducing my own version Infomap with MATLAB rather than testing datasets. I checked Infomap source official website (www.mapequation.org/) and found official code sources included C++, Python and R versions. So i implemented with efficient MATLAB code and planned to contact manager to make reference to my code if possible.

II. RANDOM WALK[2]

Random walk[2] is a really significant concept for network theory and it's a really effective way to evaluate the network topology. Random walk can be represented as the frequency for a random walker to visit each node in a network. For Infomap[3], we can use this to represent a map to describe in the dynamical way by considering links and nodes in weighted and directed networks because visit frequencies are strongly based on directions and weights of links. We can imagine there is a person walking randomly in a network based on directions and weights which means probabilities. After each step, he will move from one node to a new one through the link between them. When the step number gets larger and larger, finally, the frequency for random walker to each node will converge to stableness[2]. What we get from random walk algorithm can be used to evaluate importance of nodes in a network as centrality. Also, for some tasks just like community detection, many recent methods like Infomap[3] which we mainly talk about in this report, are based on the random walker's walking flow and we can use the concept because it is reasonable that a random walker should be trapped inside communities for a longer time than walking across communities[3]. This idea implies network structure information has been included in this somehow. To estimate the frequency for random walker to visit a node at stable number of step, the PageRank algorithm[15] is proven to provide a great way, it can also be used to evaluate the centrality of each node to represent the importance of nodes in a network.

Although, random walk process[2] can describe the real topological information of network by nodes' visit frequencies, it still has an significant drawback which cannot be ignored. That is for some networks which are not very strongly connected, like the network in the **FIG**

2, there are three components, if we start from nodes on one of components on the right side, the random walker may always be trapped in one component and cannot reach other even bigger components. So if we apply this to Infomap algorithm[3], the final result will be like we just split one component into several communities and count all the other parts of the network into one community.

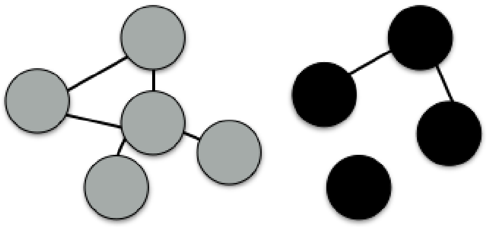

FIG. 2. **Network with more than one component**, this figure's concept is from paper[2](R. Lambiotte, M. Rosvall, Ranking and clustering of nodes in networks with smart teleportation, American Physical Society, 2012)

To solve this problem, we need to add a trick named teleportation process[2] into original random walk process. The main idea is, for the arrival to each node, we decide a probability to hop across the whole network in the next step and thus after the whole process, random walker can be better evaluated as the visit frequency for every node. Teleportation process[2] can be showm as **FIG 3**.

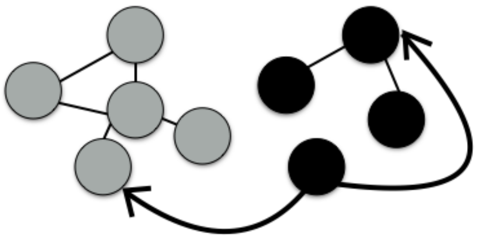

FIG. 3. **Teleportation process**, this figure's concept is from paper[2](R. Lambiotte, M. Rosvall, Ranking and clustering of nodes in networks with smart teleportation, American Physical Society, 2012)

It's clear to see for these pairs of components without any connection between them, random walker can still reach one from others by jumping across the network randomly. However, if we do so, we also need to make sure that all the nodes in network will have the same probability to teleport in the next step and all the nodes will have the same probability to get reached after the teleportation process[2]. In this way, this strategy will barely affect the visit frequency for nodes. This change is just like we plus one on every element in a set of number, the deviation and standard deviation won't change.

Now we consider the general condition, the network is directed and wighted. We can first obtain a transition matrix T_{ij}[2] for random walker, it represents the probability walker reaches node i from node j, we can get by dividing each out link weight with summation of out degree[2]. Then, we can obtain the frequency p_i[2] of random walker to node i as:

$$p_{i;t+1} = \sum_j T_{ij} p_{j;t}$$

The solution above can get convergence only when the network only has one component[2], but as we know, this condition will not show very often in the real world network, many times networks are just sparse. So we do some tricks on this solution and let the walker can jump over the network. We can induce a variance α represents the probability the walker hops to another random node in the network instead of following the edges between nodes[2]. So the new equation[2] for the frequency of node i becomes:

$$p_{i;t+1} = (1-\alpha) \sum_j T_{ij} p_{j;t} + \alpha v_i$$

Where v_i means the probability for random walker to the i^{th} node. In the most networks, we can just assume the probability for each node to be teleported to are all the same[2], and since we assume $\sum v_i = 1$[2], it can be computed as[2]:

$$v_i = \frac{1}{N}$$

Where N is the number of nodes in the network and here we get teleportation vector with same elements.

Then, we can compute the frequency for each node in the final stable stage, it can be denoted by π_i[2]. This also defines nodes' PageRank[15]. It can be represented as[2]:

$$\pi_{i;a} = (1-\alpha) \sum_j (I - \alpha T)^{-1}_{ij} v_j$$

However, for the real world networks, lots of them may contain some nodes which have no link with all the other nodes. For directed network, their out degree are zero and we call these nodes dangling node. If random walker reach to this kind of nodes, the only thing they can do for the next step is teleportation[2]. So, in the transition matrix T_{ij}, we just change the jth column's elements all into v_i and the system will still work well[2].

III. MAP EQUATION[3]

First, we need to make sure we are always following two goals[3], the first is we have to try our best to compress our network, the other one is we should find a good way to encode the network and reflect the network's structure. So what should we do. We can translate this general problem into another one that if we want to describe the paths created by a random walker, which kind of encode method we can use to represent paths and imply the network's topological information? In [3], Rosvall said "If our goal is just to maximize the compression of representation of random walker's path, we can just apply some encoding methods to give all nodes a different codename and then, path can be represented as a set of nodes' codenames. However, if maximizing the network compression is not our only purpose and we also want the encoding method to reflect network structure, this method cannot work well because for all the nodes in network, their codenames are all different."[3] That means this method contains no information of the network structure because if we change these codenames in different order, there is no difference compare with original one. Even codelength relates to this node's visit frequency. If visit frequency is large, codelegth is small. It still cannot imply any community information. So, should we just get rid of this method? Just need to do some changes of this encoding method, we can apply it to show the network structure information well and what's more, network can be compressed in a better way.

To sum up, the best encoding way should always approach two goals which are compressing and reflecting the network structure. For the first one, we can induce Huffman Coding[16]. To do the best compression of path, these nodes can be named by Huffman code[16] based on the visit frequency for every node. In this way, more common nodes will have shorter codenames and rare nodes will have longer codenames[16]. Now, we can look at a detailed example as below:

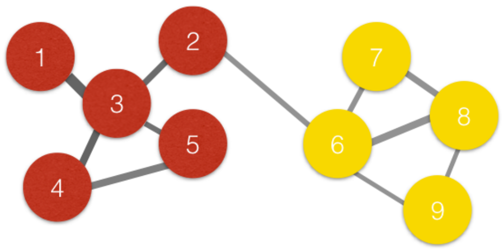

FIG. 4. **Example network**

Now we have a small size network only contains 9 nodes like this. And we also assume we already know the best community separation and i mark them in different color. After the random walk process, we get the visit frequency for each node using PageRank algorithm[15] and these are shown below:

1	2	3	4	5	6	7	8	9
0.35	0.16	0.13	0.10	0.09	0.07	0.05	0.03	0.02

TABLE I. visit frequency for each node

Now, we already gotten flow of the network represented by visit frequency, the next thing we do is creating a Huffman tree[16] based on the visit frequency and for the tree, every leaf is a node from network, like the one is shown below:

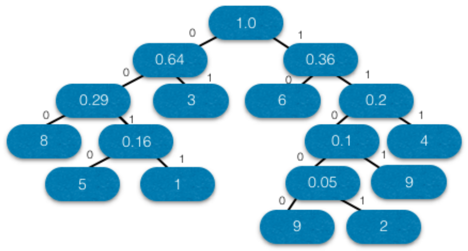

FIG. 5. **Huffman tree[16] for example network**

From the Huffman tree[16] above, we can get the codename for each node and the original network can be encoded as below:

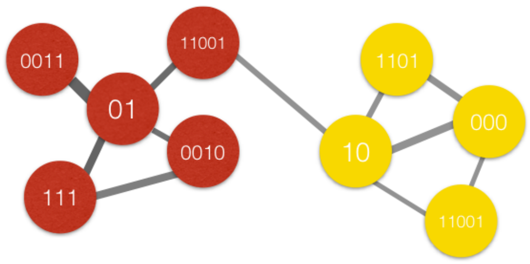

FIG. 6. **Encoded network**

At first, we can try to compress all nodes in the simplest way. We give all nodes the same length codename in the binary formation. There are 9 nodes in the network and because $2^3 < 9 < 2^4$, the code length for each node is 4. So if a path is $[3, 4, 5, 2, 6, 8]$ and we use the equal length encoding way, the total length for the path is $4 * 6 = 24$. Now, we try to use the Huffman codes[16] we got from the above process to represent the same path. In this way, each codelength of nodes are $[2, 3, 4, 5, 2, 3]$ and the total code length is $2 + 3 + 4 + 5 + 2 + 3 = 19$, which is way shorter than 24. It looks like huffman coding[16] can perform great on the information compression task. However, in the real condition, we don't need to focus on giving all nodes different codenames, and also, it's impossible to build a huffman tree[16] every time to compute the average codelength used to describe network because it will be so slow. We can just use the Shannons source coding theorem[5]. The basic explanation is that when assume all nodes have totally different codenames, average codelength of nodes will be larger than the entropy based on nodes' visit frequencies[3]. So in this way, if we get the entropy, that means we know the shortest average length of codelength for random walker's paths[3].

However, for the structure of network, this encoding approach can only imply the PageRank[15] centrality of each node, but we cannot know the relative positions in network for these nodes. What we want is actually the relations among nodes. To focus on the network topological structure, we can compress the network in 2 levels - module level and node level[3]. We separate the nodes into different modules and for each module, we represent it with a codename. Within each module, we encode the nodes belong to this module and in this way, the codename of each node in network don't need to be different from all the others[3]. Just like in different cities of American, there are always some roads share the same name, like Madison rd and Broadway rd. but when we describe them, we add the city name in front of them, the road we mentioned will still be clear[3]. Also, the other advantage of encoding network in 2 levels is that we can use shorter string of binary codenames to represent the path of network[3]. Random walker's paths can be compressed in the next level. Because this natural random walker will be trapped inside a community or module for a long time instead of walking between modules[3]. Using shorter codewords for the nodes belong to the same module can make the total length of path codewords shorter. Following is the instance, still the same small size network, and this time, we encode it depend on modules in different colors.

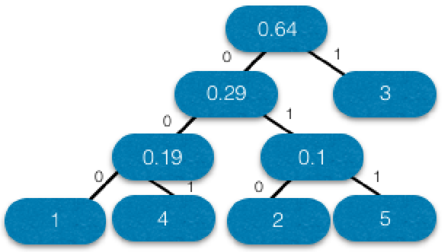

FIG. 7. **Red community Huffman tree[16]**

This Huffman tree[16] is for the red module. Still it is built based on the visit frequency of each node.

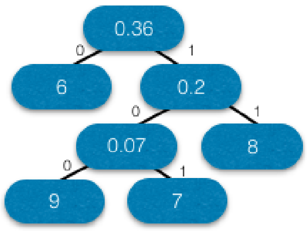

FIG. 8. **Yellow community Huffman tree[16]**

This Huffman tree[16] is for the yellow module.

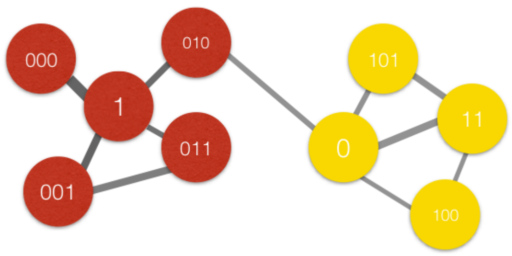

FIG. 9. **Encoded network**

The processes above are in node level, we encode each node separately for each module. Then we need to encode modules. Because here are only 2 modules, we give the red module codename 0 and the yellow module codename 1. Finally, we can get the encoded network like this and because it's encoded in two level, we can see clearly especially for the yellow module, the codelength for each node is much shorter than one level Huffman coding. And we can see for the same path [3,4,5,2,6,8], the code length becomes $1 + 3 + 3 + 3 + 1 + 2 + 1 + 1 = 14$(don't forget each time leave one module, we need to plus codelength of the module we leave), which is much shorter than 19 we got above.

To sum up, the most important concept is we try to compress the representation for the paths in 2 levels of description, and based on the goal to minimize the code length in a network, we can construct a equation to be the standard as community detection. We named it map equation[4] and it represents the average codelength for one step[4]:

$$L(M) = q_{exit}H(Q) + \sum_{i=1}^{m}(q_{exit}^{i} + \sum_{\alpha\in i} p_{\alpha})H(P^{i})$$

There are two terms in this formula. The first part represents the codelength used to describe the steps between modules and the second part represents the codelength used to describe the steps within each module[3]. In this formula, H[3] means entropy and it can be calculated by[3]:

$$H = -\sum p_i log_2 p_i$$

and q_{exit}[3] means the summation of probabilities random walker leaves each module, it is[3]:

$$q_{exit} = -\sum_{i=1}^{m} q_{exit}^{i}$$

so the entropy of Q, $H(Q)$[3] should be[3]:

$$H(Q) = \sum_{i=1}^{m} \frac{q_{exit}^{i}}{\sum_{j=1}^{m} q_{exit}^{j}} log(\frac{q_{exit}^{i}}{\sum_{j=1}^{m} q_{exit}^{j}})$$

Also, for the second term of map equation, we should notice that we also add probability of module exit[3].

To sum up, map equation[4] is used as the standard of community spliting. It will work combined with node merge algorithms which will be mentioned in the next part, like Greedy algorithm[6] and Simulated annealing algorithm[7]. Before merging a pair of communities, we should first test all possible ways, calculate the average code lengths and pick the pair with shortest code length to merge. The whole process will come to converge when we get the shortest code length and the shortest length for the next step going up.

IV. MERGE ALGORITHM

This is the last part for Infomap[3]. Based on map equation[4], we need to list all the possible partitions. Several algorithms can achieve that and in our case, we use greedy search algorithm[6]. Though simulated annealing algorithm[7] has more possibility to get to the global optimal, for convenience of comparing with modularity community detection algorithm[6], we still use greedy search algorithm[6] as our merge algorithm.

Initially, we treat each node as one community and then calculate $L(M)$[3] when merge two communities. After testing all pairs of communities, we choose the pair with the smallest $L(M)$ to merge into one community, then we finish a step. We repeat this until cannot find any pair of communities when they merge can reduce $L(M)$[6]. It means we reach the convergence and this is the best separation of communities for this network[6].

V. EXPERIMENTS

In this section, some results are shown using different datasets and comparing with modularity community detection algorithm[6]. There are some other network datasets were tested by these two algorithm, because results were not typical, so i don't mention here.

For datasets, the first i used is Zachary's karate club network[8]. This network show the relationships between 34 students in karate club[8], the size of network is 34 and number of edges is 77[8]. This is a undirected and unweighted network, dataset file was downloaded from **http://tuvalu.santafe.edu/aaronc/data/zkcc-77.zip**. The second network i used is US contiguous states network[9], which is a network includes 48 contiguous states (with Washington D.C)[9]. This network eliminates the datasets of Alaska and Hawaii because these two are not connected with any other 49 states by land[9]. Node denotes each state and edge means two states are contiguous by land[9]. This dataset file was downloaded from **http://konect.uni-koblenz.de/networks/contiguous-usa**. The third dataset i chose is zebra social network[10]. This dataset has 28 nodes and 111 edges[10]. Every node represents a wild Grevy's zebra and every edge means these two zebras appear together during the observation period[10]. This is a weighted network and there are two values. 1 means this pair of zebras appear together once and 2 means this pair of zebras appear together more than one time[10]. For all zebras, there are three labels based on zebras' sex and breeding condition[10]. This dataset was downloaded from **http://konect.uni-koblenz.de/networks/moreno-zebra**.

For the network visualization part, i used d3.js[11] as visualization tool. Its full name is Documents Data-Driven[11]. Just like what its name implies, it's known as a data driven document JavaScript class library. To put it simply, d3.js is mainly used for operation of data through using HTML, SVG and CSS to give data life, which is converted to a variety of simple and gorgeous graphics which can help people understand data much easier[11].

A. Zachary's karate club[8]

For this experiment, i used force layout[11] of d3.js to visualize the network. The advantage of this technology is that it can reflect the connection strength of members within a network. Between each pair of nodes, there is a repulsive force and as all, there is a gravitational force, finally the nodes in the interaction of various forces will gradually converge to a stable position[11]. In this way, we can recognize which nodes are on the boundary of communities and easy to be classified incorrectly.

The figure below shows the true community splits of karate club.

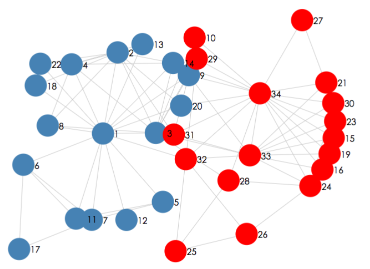

FIG. 10. **Zachary's karate club network[8]**

From this figure, we can see clearly that some nodes like 10, 29, 14, 9, 20, 3, 31 are on the boundary of communities. And then, i used Infomap[3] and modularity algorithm[6] to detect the communities separately. The following two figures are what we got.

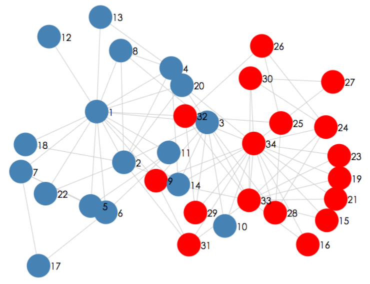

FIG. 11. **Communities of Zachary's karate club network using Infomap**

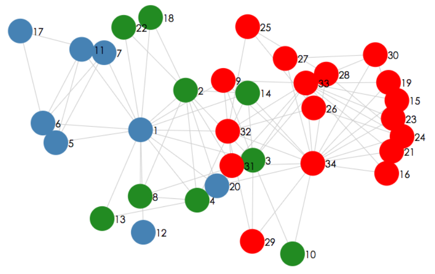

FIG. 12. **Communities of Zachary's karate club network using Modularity**

The first one is the result i got from Infomap[3] and the second one is the result i got from modularity algorithm[6]. It's not hard to tell that with Infomap, there are only two nodes misclassified which are 9 and 10. However, with modularity, this network is splited into three communities. The original blue community is separated into two groups and the green group covers lots of nodes on the boundary of the true network, like 3, 10, 14. Because of this, many nodes even not on boundary but connected with these green nodes turn green too which makes the detection is totally different from the real one. We also calculate the NMI (normalized mutual information)[14] for these two methods, for Infomap, the NMI is 0.677243041103, for modularity, the NMI is 0.564606879094. Now, we can see the better performance of Infomap more clearly and directly.

B. Contiguous States of USA [9]

For this experiment, i used projection function mercator[11] of d3.js to visualize the network. However this time, i didn't express the network as nodes and edges, i create a US map for each result, different communities with different colors. In this way, it will be easier for us to compare the difference of two algorithms and the similarity of detections and original one. But here comes a problem, i lack the real world situation to judge if the community separation gets a good result because i do not have a standard only depends on the location to separate US into different parts. So here i use the Six Regions of the United States (New England, Mid-Atlantic, South, Midwest, Southwest and West)[12] to represent the true situation. Though i inspire different regions, the landscapes and climates are different depend on regions as well as the cultures, all the states in a region are still contiguous which means it depends on the geography in some level[12]. So i can use it to estimate the result we get only based on geography. To produce the map, i also downloaded American json file which includes the location of each state and some basic information about that from https://gist.github.com/mshafrir/2646763.

The original network's figure looks like the following:

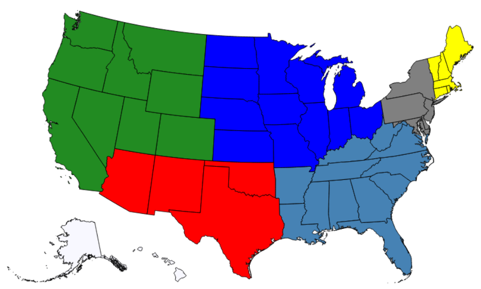

FIG. 13. **Contiguous States network[9]**

I marked each region in different color from others. And then, i used Infomap[3] and modularity algorithm[6] to detect the communities separately. The following two figures are what we got.

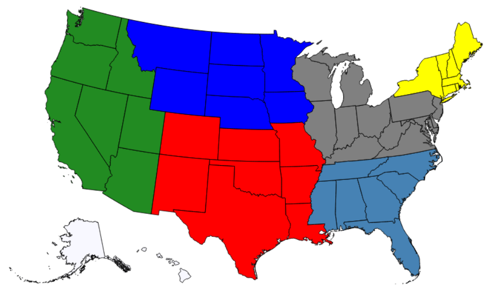

FIG. 14. **Communities of Contiguous States network using Infomap**

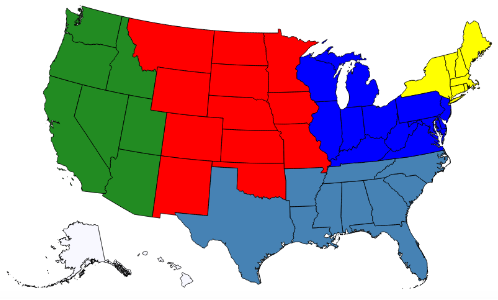

FIG. 15. **Communities of Contiguous States network using Modularity**

On this network, two methods also get different results. The first one is the result i got from Infomap[3] and the second one is the result i got from modularity algorithm[6]. It's clear to see that with Infomap, there are also six communities on map and each region is located in the relative similar part of US compare to the true condition and all regions' sizes are more uniform. However, with modularity, this network is only splited into five communities. Western coast and upper part eastern coast look the same as using Infomap, however, the southern part is merged into Mid-Atlantic and Midwest regions. We also calculate the NMI (normalized mutual information)[14] for these two methods, for Infomap[3], the NMI is 0.616419181496, for modularity[6], the NMI is 0.5841288801824. So in this case, Infomap is also proved better than modularity algorithm. The NMI score for both methods are both not high may because the label of this network is actually not totally based on geography, the region division is formatting gradually and affected by other facts like atmosphere, landscape inside each states and the most important, traditions and cultures[12], like New England region is because of English settlement in the Americas.

C. Zebra Social Network[10]

This dataset is based on zebra social relations and behaviors[10]. This is a weighted network, 1 means a pair of zebras appear together once and 2 means a pair of zebras appear together more than once during observation period[10].

The figure below shows the true community based on sex and breeding condition.

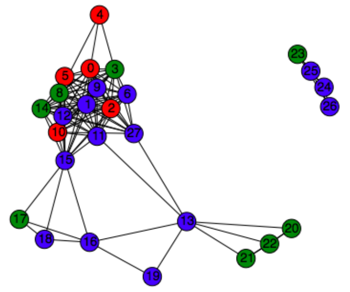

FIG. 16. **Zebra Social Network[10]**

And then, i used Infomap[3] and modularity[6] algorithm to detect the communities separately. The following two figures are what i got.

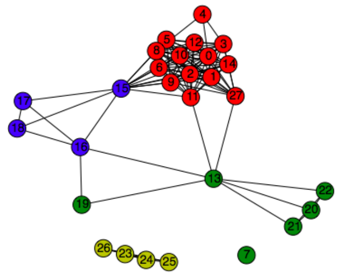

FIG. 17. **Communities of Zebra Social Network using Infomap**

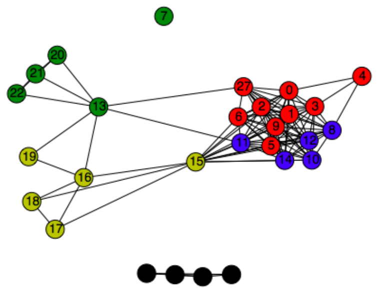

FIG. 18. **Communities of Zebra Social Network using Modularity**

For Infomap[3], the network is separated into 4 communities and for modularity[6], the network is separated into 5 communities. On this dataset, both of these two algorithms didn't work well. However, from figures we can see Infomap and modularity both can categorize nodes which connect tightly into one community. Because nature also has other factors to affect the appearance of zebra, From the true network condition, we can see a component includes all three kinds of nodes, it can mean zebras have slight trend to appear together based on sex, but it doesn't absolutely depend on that. if we only consider some other factors in labels except sex and breeding condition, these two algorithms may both work well.

D. Compare with Modularity[6]

After some experiment, we can find in some cases, Infomap[3] performs better than modularity[6]. So, what's the weakness for modularity compared with Infomap? The usual method we use to detect community structures for directed and weighted networks is just ignored the direction and weight and treat edges in the simplest way. However, for weighted or directed network, these properties are important and contain information should be used for community detection[3]. So, for modularity algorithm[6], there are still some methods consider information about weight and direction and if we want to calculate the modularity[6] of a partition of m communities. It will be [13]:

$$Q = \sum_{i=1}^{m} \frac{w_{ii}}{w} - \frac{w_i^{in}}{w_i^{out}}$$

where w is the summation of all edges' weights and w_{ii} is the summation of all weights of edges attached to community i[3]. What we do is maximizing this[3].

We can compare it with $L(M)$[3] of Infomap and it's not hard to see for Infomap, it's based on visit frequency random walker follow in network, which we can also see as information flow in the network. But for modularity[6], it's just based on the weights of edges attached to communities. We can see an example network in **Fig 19**

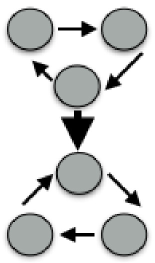

FIG. 19. **A simple network**, this figure's concept is from paper[3](Maps of random walks on complex networks reveal community structure, PNAS, 2008)

If we use Infomap[3], this graph is prefered to be separated into communities like the picture shows in **FIG 20**. Because in the upper community and down community, all nodes interact with one another and finally get into a circle. This is a completed process that information flow can get to all parts of the network:

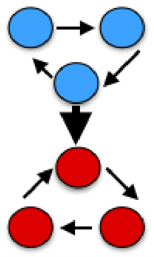

FIG. 20. **Network detected with Infomap[3]**, this figure's concept is from paper[3](Maps of random walks on complex networks reveal community structure, PNAS, 2008)

But, if we use modularity algorithm[6], the graph will be splited into the following condition in **FIG 21**. It's clear that the link in the middle has larger weight.

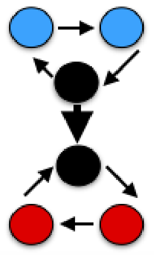

FIG. 21. **Network detected with Modularity[6]**, this figure's concept is from paper[3](Maps of random walks on complex networks reveal community structure, PNAS, 2008)

For this case, the way Infomap[3] does looks better because all the nodes in the same community have interactions with others (have in degree and out degree) and interactions can build a circulate even weights of links between communities are larger[3]. However, it doesn't mean Infomap[3] is better than modularity[6] definitely. It depends on what kind of network we are working on. For some networks which are steady and edges don't mean transfer process, modularity is a better choice. However, for some networks like world trade process, Infomap will be more meaningful.

VI. FUTURE WORK

In this report, we talk a lot about theory and concept of Infomap algorithm[3]. For the record, because my main work is reproducing MATLAB version Infomap, we explain Infomap algorithm based on some original Infomap papers like [1–4]. For the experiment part, we just apply Infomap algorithm on some small size networks which are also simple and the main purpose is to test if my version Infomap can work in efficient way. For the future work, we can test our version on some large scale networks with complex structures like directed and weighted network, even use on mutiviews network, and in this way, improve my version to meet more strict requirements and tasks. We can also analyze efficiency of this version Infomap. About the teleportation part [2], there are some other smart teleportation processes[2] we didn't implement and if we can apply better teleportation depend on what task we face, the accuracy and efficiency will be improved. Last, it's also meaningful to upload implementation as open source on website and if possible, contact manager of official map equation website to link my version Infomap.

[1] Martin Rosvall, Carl T. Bergstrom, *An information-theoretic framework for resolving community structure in complex networks*, PNAS, vol.104 no.18, pp. 73277331, 2007.

[2] R. Lambiotte, M. Rosvall, *Ranking and clustering of nodes in networks with smart teleportation*, American Physical Society, Vol.85, Iss. 5, 2012.

[3] R. Lambiotte, M. Rosvall, *Maps of random walks on complex networks reveal community structure*, PNAS, vol.105 no.4, pp. 11181123, 2008.

[4] M. Rosvall, D. Axelsson and C.T. Bergstrom, *The map equation*, THE EUROPEAN PHYSICAL JOURNAL, Special Topics 178, pp. 1323, 2009.

[5] Shannon CE, *A Mathematical Theory of Communication*, The Bell System Technical Journal, 27:379 423, 1948.

[6] M. E. J. Newman, *Modularity and community structure in networks*, PNAS, vol.103 no.23, pp. 85778582, 2006.

[7] Chii-Ruey Hwang, *Simulated annealing: Theory and applications*, Acta Applicandae Mathematica, Vol.12, Issue 1, pp. 108111, 1988.

[8] W. W. Zachary, *An information flow model for conflict and fission in small groups*, Journal of Anthropological Research, Vol.33, No.4, (Winter, 1977), pp. 452-473, 1977.

[9] D. Knuth, *The Stanford GraphBase: a platform for combinatorial computing*, Addison-Wesley Publishing Company, 1993.

[10] S. R. Sundaresan et al., *Network metrics reveal differences in social organization between two fission-fusion species, Grevy's zebra and onager*, Oecologia 151(1), Vol.151, Issue 1, pp. 140149, 2007.

[11] Michael Bostock, Vadim Ogievetsky, Jeffrey Heer *D Data-Driven Documents*, IEEE Transactions on Visualization and Computer Graphics, vol.17, issue 12, pp. 2301 - 2309, 2011.

[12] *The Regions of the United States*, U.S. Diplomatic Mission to Germany, 2008. http://usa.usembassy.de/travel-regions.htm

[13] E. A. Leicht and M. E. J. Newman, *Community Structure in Directed Networks*, PHYSICAL REVIEW LETTERS, vol.100, issue 11, 2008.

[14] David L. Wallace, *A method for comparing two hierarchical clusterings: Comment*, Journal of the American Statistical Association, vol.78, issue 383, pp. 553-569, 1983.

[15] Taher H. Haveliwala, *Topic-Sensitive PageRank*, WWW '02, pp. 517-526, 2002.

[16] David A. Huffman, *A Method for the Construction of Minimum-Redundancy Codes*, Proceedings of the IRE, 1952.