Exploring the Community Set Space

Jerry Scripps

Computing and Information Systems
Grand Valley State University
Allendale, MI, USA
Email: scrippsj@gvsu.edu

***Abstract*—This paper presents the community set space canvas, a triangular canvas where the results of community finding algorithms can be plotted for comparison. The points of the triangle represent trivial sets, such as the set of one large community, and the edges are populated by well known set types, such as disjoint communities.**

I. Introduction

Community finding algorithms [3] are unsupervised and so there is no obvious method of comparing the results of one algorithm with another. Some metrics have been proposed but none (especially when overlapping communities are considered) are as well accepted as, for example, precision is in classification.

The result of a community finding algorithm is a set of communities (or community set). We refer to the space of all possible community sets as the community set space. Using three simple metrics the community set space can be mapped to a triangular canvas. Areas of the canvas correspond to the general classes of community sets: disjoint communities, cliques and ego-centric communities.

Disjoint communities, cliques and ego-centric communities (every node has at least one community to which it and all of its neighbors belong) have characteristics that allow these sets to fill different needs. Given a social network one might want to separate members into, say, study groups (disjoint), special interest groups (cliques) or personalized groups (ego-centric).

The contributions of this paper are to present the metrics and the canvas as a way to compare and evaluate community finding algorithms. In Section II we introduce the notation used in sections III and IV, which detail our canvas and describe the algorithms to be compared. We present the experimental results in Section V.

II. Notation and Metrics

A network $G = (V, E)$ is a closed systems of *nodes* V and are *linked* to each other by edges $E \subset V \times V$. Nodes can also be grouped into *communities*, $c_i = \{v_j, ...v_m\}$, through a process called community finding. A node v_i can be placed in more than one community, but only one is designated as it's home community. A *community set* $S = (C, h)$ is a pair where $C = \{c_1, ..., c_k\}$ is a collection of k communities and h is a home community function. We use the symbol $\mathcal{S}_k$ to represent the collection of all community sets of k communities. In this paper an ego-centric community for node v_i is one in which all of v_is neighbors are in the community. An ego-centric community set is one in which each node's home community is an ego-centric community for that node.

To define community set space we present the following metrics:

DEFINITION 2.1. *Missing neighbors* are those neighbors of a node that do not appear in the node's home community.

$$M_{mn}(S) = \sum_{v_i \in V} \left[deg(v_i) - \sum_{v_j \in h(v_i)} I((v_i, v_j) \in E) \right]$$

where $deg(v_i)$ is the degree of v_i and $I(\cdot)$ is an indicator function which is 1 when the argument is true and zero otherwise.

DEFINITION 2.2. *Extraneous nodes* are nodes not directly linked to a node within its home community.

$$M_{en}(S) = \sum_{v_i \in V} \left[\sum_{v_j \in h(v_i)} I((v_i, v_j) \notin E) \right]$$

DEFINITION 2.3. *Overlap* is the number of communities that a node is placed in besides its home community.

$$M_{ol}(S) = \sum_{v_i \in V} \left[\sum_{c_j \in C} I(v_i \in c_j) \right] - |V|$$

Generally there is a trade-off between the metrics; a lower value of one of the metrics will normally result in a larger value for one or both of the other.

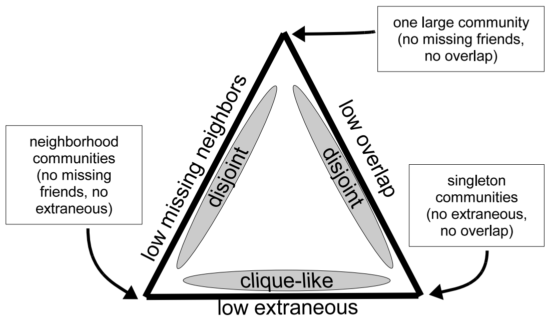
one large community (no missing friends, no overlap)
neighborhood communities (no missing friends, no extraneous)
singleton communities (no extraneous, no overlap)
low missing neighbors
low overlap
disjoint
disjoint
clique-like
low extraneous

Figure 1. Community Set Space Canvas

III. Community Set Space

In this section we introduce the concept of *community set space*. This space refers to the set of all possible community sets with respect to the three metrics: missing neighbors, extraneous nodes and overlap.

A. Canvas: The community set space canvas is a two dimensional chart for plotting community sets. It provides a visual means to compare community sets and to see where a community set fits into the space of all possible community sets.

Figure 1 shows the canvas as an equilateral triangle in which each side corresponds to a low or zero measurement of one of the metrics. The lower edge corresponds to low extraneous values, the left edge corresponds to low missing neighbors and the right edge corresponds to low overlap. Moving away from an edge towards the other side of the triangle, the value of the metric gets increasingly larger.

The three points of the triangle represent special cases and are labeled as such. The top point, where there is zero overlap and zero missing neighbors would be the community set defined by one large community. It would be every node's home community, would contain all neighbors of all nodes and would have no overlap. Note however that there would be many extraneous nodes which is why it is opposite from that edge.

The point in the lower left is where there is zero missing neighbors and zero extraneous nodes. This is defined as the set of all neighborhood communities – that is, for each node, create a home community consisting of that node and its neighbors. Every node's home community would contain all of its neighbors and have no extraneous nodes. There would however, be much overlap.

The point in the lower right has zero extraneous nodes and zero overlap. This point is the set of singleton communities – that is where each node is in its own community. Again, two of the metrics, overlap and extraneous nodes are zero while the third, missing neighbors is very high.

This chart can be used to compare community sets for the same network or more generally for different networks. To better understand the community sets, labeled bars are placed below the chart to indicate the total number of violations ($M_{tot} = M_{mn} + M_{en} + M_{ol}$) in Section V.

B. Community Finding Problems: Using the community set space we can define community finding problems more narrowly. First we consider the problem of restricting the community set to zero M_{ol} and then attempting to minimize either M_{mn} or M_{en}. These community sets would fall along the edge of zero overlap and are labeled disjoint communities. Second is the problem of community sets where M_{mn} is zero and either M_{en} or M_{ol} is minimized which fall along the edge of zero missing neighbors and are labeled as ego-centric communities. A third problem is that of having zero M_{en} and minimizing one of the other two metrics. These community sets appear along the bottom edge and are labeled clique-like.

For all of the above problems we can choose to either supply k, the number of communities desired or to allow the algorithm to choose the best k that optimizes the requirements. If k is specified then the value chosen for k will also have an impact on the metrics. In general, we expect that as k decreases there will be in increase in M_{en}. This will be demonstrated in the experiments section.

IV. Algorithms

A. Selected algorithms: The first algorithm considered is spectral clustering [4], or equivalently, normalized cut. In normalized cut, each cut (potential split of two groups) is weighted by the size of the groups. Balanced cuts are weighted higher and are more likely to be used. Another splitting method, called fast modularity [1], uses modularity to locate and remove high traffic edges.

An agglomerative approach [3] is also considered, where nodes are assigned to their own community (called singletons in this paper) and the communities stepwise joined. Another method has recently been proposed [5] where, instead of starting with singletons, it starts by forming neighborhood communities around each node and then joining communities as before.

Another community finding algorithm [2] merges similar cliques to form communities. Communities are formed of k-cliques (cliques of size k) for as many values of k as possible. A k-community is the union of all k-cliques that are connected by k-1 common nodes.

B. Algorithm characteristics: The algorithms are listed in Table I. The spectral clustering, fast modularity and community finder algorithms are the actual algorithms that were available from the authors. The two agglomerative methods were specifically coded for this paper. For the ego-centric version, we begin with node-neighborhoods, as in [5], but instead of using the Jaccard index, we merged communities based on which two would result in the largest reduction in M_{en}. Both M_{en} and M_{ol} were tested but in general, using M_{en} resulted in the best performance. For singletons, we also used M_{en} as the criteria for merging communities.

Table I
ALGORITHMS AND CHARACTERISTICS

algorithm	**abbr.**	$\mathbf{M_{ol}}$	$\mathbf{M_{mn}}$	$\mathbf{M_{en}}$
agglom. Ego	agglE	some	zero	min
agglom. Sng	agglS	zero	some	min
spectral clust.	NCut	zero	some	some
Fast Modularity	Mod	zero	some	some
Comm. Finder	CFind	some	some	low

The table lists some of the characteristics that we expect to observe in the experiments. The agglE algorithm starts with node neighborhoods, which means there should be zero M_{mn}. As the algorithm merges communities, every node's home community will always contain all of its neighbors so we expect M_{mn} to always be zero. And since it attempts to minimize M_{en} that value is expected to be low. On the other hand, the agglS begins with singletons which have no overlap. Merging should not create any overlap so we expect M_{ol} to be zero with minimum M_{en}.

NCut and Mod both produce disjoint communities and neither is specifically tuned to reduce any of the three metrics. So while they do not specifically attempt to minimize M_{mn} or M_{en} we expect these values to be low since both algorithms attempt to find communities with many intra-community links and few inter-community links. Since the CFind algorithm is based on cliques, we expect to find low values of M_{en}.

The proportion of M_{mn}, M_{en} and M_{ol} is influenced not only by the choice of algorithm. The number of communities is also a factor. It seems logical that as the number of communities is reduced the value of M_{en} rises and the values of M_{mn} and M_{ol} fall. The experiments will bear out this general supposition.

Note that with a simple operation we can also easily change M_{mn} and M_{ol}. Given a community set, we can add a node to non-home community where it does not exist. We can also remove it from a non-home community where it does exist. Adding nodes will increase M_{ol} (and possibly M_{en}) and decrease M_{mn}. Removing nodes will have the opposite effect. If this is done carefully, it can actually decrease M_{tot}. We propose this as a post-processing operation for NCut, Mod and CFind and we will integrate it into the two agglomerative algorithms.

V. Experiments

A. Data Sets and Setup: Two data sets were used that are non-directional networks and have binary link and attribute data. The American college *foot*ball (http://www-personal.umich.edu/ mejn/netdata/) network represents the schedules of teams in the NCAA college football, division 1A division. There are 115 nodes, representing the schools and 613 links representing the games. The *jazz* musician's dataset (http://deim.urv.cat/ aarenas/data/welcome.htm) has 198 nodes representing musicians and 2,742 links (presumably representing their collaboration).

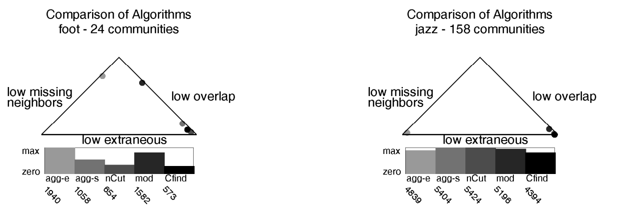

Figure 2. Plots CFinder communities and other algorithms using same number of communities

The experiments a) compare algorithms or b) compare different numbers of communities for the same algorithm. Each algorithm produces a community set, metrics are calculated and then results are plotted on the canvas. For Mod and CFind algorithms where where some nodes are not assigned to a community, we assigned the orphan nodes to the communities to which they had the most in common. Another post processing step for NCut, Mod and CFind was to assign a home community for each node.

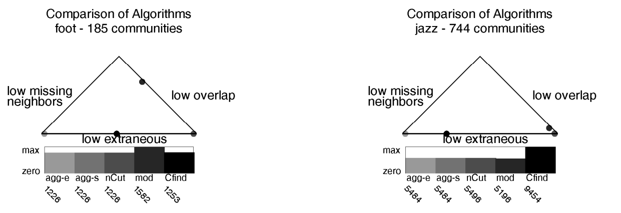

Figure 3. Plots CFinder cliques and other algorithms using same number of communities

B. Comparison of Algorithms: The experimental results start out with a simple comparison of the algorithms. The charts will show that the different algorithms will produce community sets with different characteristics as predicted in the previous section. All of the algorithms allow the user to specify k, the number of communities, except CFind which creates communities from cliques. CFind presents communities made from cliques of 3, 4, etc. So in order to make a comparison, the CFind algorithm was run first (using the clique size that gave the best results) and the number of communities it created was used for the input k to the other algorithms. The the other algorithms perform better at other values of k.

The charts in Figure 2 show community sets plotted on the canvas using 24 communities for foot and 158 for jazz. Looking at the foot chart, the CFind set appears along the bottom edge having zero M_{en}. The sets for NCut, Mod and agglS are all aligned along the edge with zero M_{ol} and agglE is on the edge with zero M_{mn}. All of the algorithms appear to have small numbers of M_{en} which is not surprising given the relatively high value of k. In the jazz canvas, the CFind algorithm returned 158 communities. This value of k is relatively close to the number of nodes so it is not surprising that the sets all have low values of M_{en}.

Another set of experiments was run using the *cliques* that are produced by CFind instead of the communities. Figure 3 shows the results. Notice that the cliques do in fact fall along the edge of zero M_{en} as expected. Because the jazz data set has a higher average degree we would expect it to form larger cliques so that most nodes belong to a community with many of its neighbors. This is borne out by the placement of the community set being closer to the ego-centric corner than to the singletons. In both cases the resulting number of cliques is higher than the number of nodes, so the other algorithms are quite close to either the singleton or ego-centric corner. The only exception is Mod which apparently still finds a large component even when k is large.

C. Trace of algorithms: We plotted a community set trace for each algorithm varying the value of k. Two representative plots (agglE and Mod) can be viewed in the canvases of Figure 4 for the foot data set. Results for the other algorithms and data set are quite similar and omitted to save space. The numbers are difficult to read on these charts but the importance of the charts is the trace of the community sets.

Mod starts in the singleton corner for large values of k and then move up the edge of zero M_{en} as the size of k is reduced (agglS and NCut behave the same). The sets for agglE start in the ego-centric corner and move up the edge of zero M_{mn} as k is reduced. All of the algorithms appear to have monotonically increasing M_{en} as k is reduced.

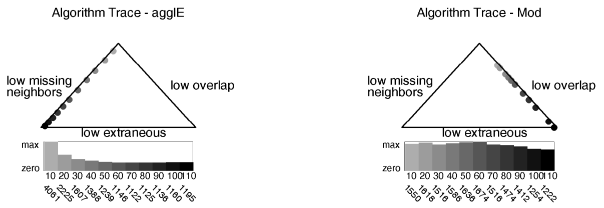

Figure 4. Community Set Space Trace for Foot

D. Augmented agglomerative algorithms: At the end of Section IV we suggested a simple set of operations to add or remove nodes to a given community set. In this subsection the results of *integrating* the operations into the algorithm itself will be tested. The two algorithms agglS and agglE are step-wise iterative methods. Each time two communities are merged it is possible that the M_{tot} can be improved. The augmented algorithms called agglSaug and agglEaug add and remove nodes that improve M_{tot} each time two communities are merged.

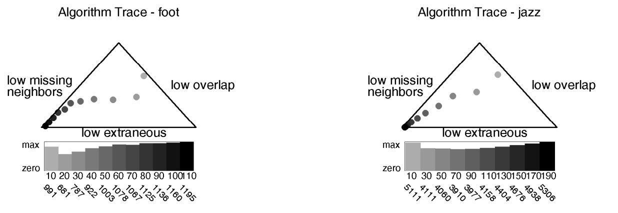

Figure 5. Community Set Space Trace for agglE

The results for agglEaug are shown in Figure 5. In both data sets, the community sets begin as expected in the corner with the neighborhood communities but instead of following the edge of ego-centric communities, the sets have reduced overlap at the expense of a little more M_{mn}.

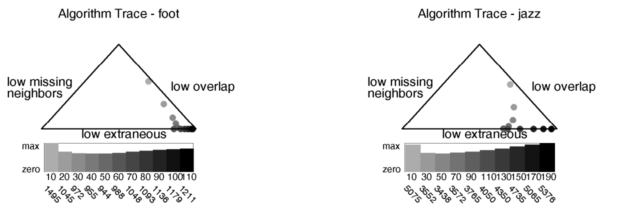

Figure 6. Community Set Space Trace for agglS

Similarly, the results for agglSaug can be seen in Figure 6. Again the community sets are drawn towards the center as k is decreased, however with agglSaug, the M_{tot} can be reduced more suddenly than with agglEaug but appears to draw back toward the disjoint edge.

References

[1] A. Clauset, M. E. J.Newman, and C. Moore. Finding community structure in very large networks. In *Statistical Mechanics*, 2004.

[2] G. Palla, I. Dernyi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435, Jun 2005.

[3] M. Porter, J. Onnela, and P. Mucha. Communities in networks. *Notices of the American Mathematical Society*, 56, Feb 2009.

[4] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 22(8), August 2000.

[5] Lei Tang, Xufei Wang, Huan Liu, and Lei Wang. A multi-resolution approach to learning with overlapping communities. In *KDD Workshop on Social Media Analytics*, 2010.