Community Finding within the Community Set Space

Jerry Scripps
Grand Valley State University
Allendale, MI 49506, USA
scrippsj@gvsu.edu

Christian Trefftz
Grand Valley State University
Allendale, MI 49506, USA
trefftzc@gvsu.edu

ABSTRACT

Community finding algorithms strive to find communities that have a higher connectivity within the communities than between them. Recently a framework called the community set space was introduced which provided a way to measure the quality of community sets. We present a new community finding algorithm, CHI, designed to minimize the violations defined by this framework. It will be shown that the CHI algorithm has similarities to kmeans. It is flexible and fast and can also be tuned to find certain types of communities. It is optimized for the community set framework and results so that it performs better than other algorithms within that framework.

Categories and Subject Descriptors

Data: 07 Social Networks [**Community Finding**]

General Terms

Community Finding, Networks, Link Analysis

Keywords

Community Finding, Networks, Link Analysis

1. INTRODUCTION

Increasingly, networks are being used in programs to represent the complex relationships that naturally occur in social, biological, computer and other networks. Often, practitioners are interested in grouping the nodes into communities. These community sets can be helpful in describing and analyzing the network.

There is general agreement that "good" communities are ones that have many links (or edges) within the communities and fewer of them between the communities. However, an exact definition of the "best" set of communities for a network is elusive. One issue is that the search space is large. The Bell number [1] tells us the exact number of partitions for a network of size n. For a very small network such as Zachary's karate club [17], where $n = 34$, the Bell number is 2.1×10^{28}. The solution space grows exponentially with the number of nodes.

Another issue is that community sets have different characteristics that appeal to different analysts. Overlapping communities are appropriate for some uses and for others it is better to use disjoint communities. Some communities will be similar to cliques while others are not. Finally, communities can be ego-centric (where nodes are in with a high proportion of their neighbors) or not.

A number of algorithms have been proposed [9, 5, 16] that automatically detect communities by using the link structure. There are many of them for several reasons:

- the algorithms are designed to detect communities with specific characteristics
- because there is no agreed-upon metric for measuring quality, it is difficult to judge between them
- there are many approaches to the problem, such as optimizing an error function, minimizing a graph theory metric or clustering techniques

In addition to the issues above, the efficiency of algorithms varies. Because the solution space is so large, making an efficient algorithm is difficult. Many of the proposed solutions can produce good results but are very slow.

The algorithm that we present in this paper makes use of the community set space framework [10]. It contains the concept of communities and home communities. A node can be placed in more than one community but it can only be placed in one home community. This framework also provides a way to measure and compare community sets using three metrics (or violations): missing neighbors M_{mn}, extraneous nodes M_{en} and overlap M_{ol}. These metrics will be formally defined in a later section, but briefly, a missing neighbor is a linked node that does not appear in the node's home community, an extraneous node is a non-neighbor that appears in a node's home community and overlap is the number of times a node appears in communities other than its home.

The concepts can be illustrated by looking at Figure 1. One can see nodes a through n are placed in communities comm1, comm2 or comm3, with only node d being placed in both comm1 and comm2. Assume that the home community for d is comm1 and for the others, it is simply the community they are already in. Notice that, among other violations, d has two missing neighbors in comm2, that k is

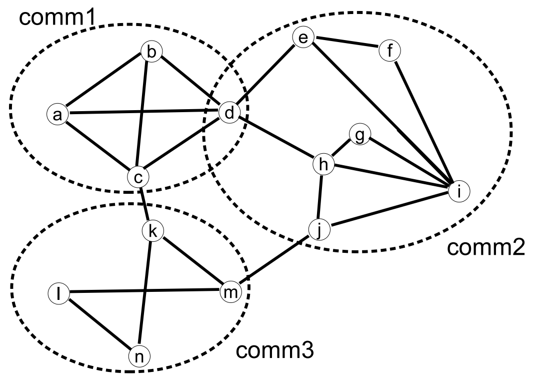

Figure 1: Network to describe the metrics of the community set space

in a community with one node extraneous to it and there is only one overlap, that of node d.

In this paper we introduce the algorithm CHI in which we use the total of the metrics $M_{tot} = M_{mn} + M_{en} + M_{ol}$, as our objective function to be minimized. We do this by alternately changing the communities and then the home communities. In the first step we hold the home communities constant and re-assign nodes to communities so that we reduce M_{tot}. Next, we hold the communities constant and re-assign nodes to home communities, again reducing M_{tot}.

This is illustrated by the very small network in Figure 2. In (a), we see the network with two cliques of 3 nodes each. We are going to attempt to discover two communities. Assume that the initial community assignment in (b) where a, b and d are assigned to one community and c, e and f are assigned to the other. Also assume that the home communities are the same as the communities.

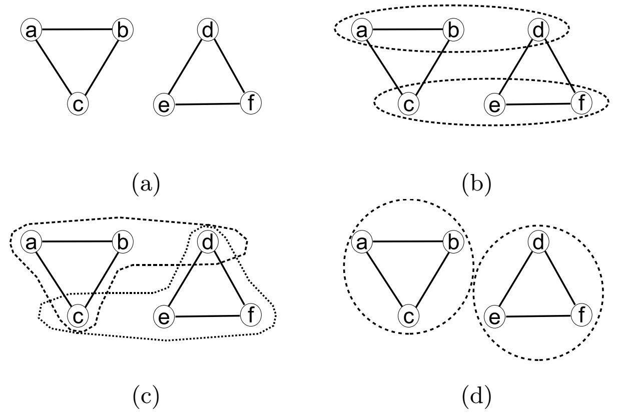

Figure 2: Community Set Space Plot for Teen

First, we fix the home communities and re-assign nodes to communities. Notice that we can put node c into the community with a and b which will increase the overlap by one but decrease missing neighbors by 2. We can do the same for d. The result can be seen in Figure 2 (c). Then we fix the communities and re-assign the home communities. Again we can reduce M_{tot} by changing the home community of both c and d. Now, we fix the home communities and remove c and d from the communities they were originally in (to reduce M_{tot} and we are left with the communities in Figure 2 (d).

CHI will be shown to have similarities to and many of the advantages of the kmeans clustering algorithm. First of all it is efficient. Second, within the framework of the community set space, it has fewer violations than any of the other methods tested. This is not surprising as it is designed to optimize M_{tot}. Third, like kmeans, it starts with an input community set. This is a disadvantage because it is sensitive to the initial set. However, this is also an advantage because an analyst can use the result from another algorithm as input to CHI and it will improve its M_{tot}. Unlike kmeans, it is flexible in that it can produce community sets with many different characteristics.

After this introduction, necessary terms will be defined. Following that section, the community set space is defined and analyzed. In the next section the new problems and algorithms are introduced. Then the experiments section shows the results of tests run on the new algorithm. After the experiments, we discuss related work and then finally present conclusions.

2. NOTATION AND METRICS

A network $G = (V, E)$ is a closed systems of *nodes* V which 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 community is designated as its home community. A *community set* $S = \{G, C, h\}$ is a triplet 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.

For describing the algorithm it will be convenient to represent the network G by an adjacency matrix $A = [a_{ij}]_{n\times n}$ where $a_{ij} = 1$ if there is a link between nodes v_i and v_j and $a_{ij} = 0$ if there is not a link. Furthermore, we shall also represent both the communities and home communities by 0/1 matrices. For communities, $C = [c_{ik}]_{n\times K}$ where $c_{ik} = 1$ if v_i is in community k and zero otherwise. Likewise, for home communities, $H = [h_{ik}]_{n\times K}$. Later we will represent the network as a neighbor list and the home communities as a vector to make the algorithm more efficient.

2.1 Metrics

The community set space is defined by the following metrics:

Definition 1. Missing neighbors are those neighbors of a node that do not appear in the node's home community.

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

where $deg(v_i)$ is the degree of v_i and $I(\cdot)$ is an indicator function which is one when the argument is true and zero otherwise. We also use $c(h(v_i))$ to represent the set of nodes in v_i's home community.

Definition 2. Extraneous nodes are nodes not directly linked

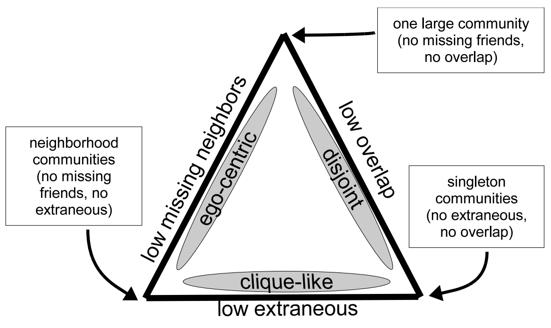
one large community (no missing friends, no overlap)
neighborhood communities (no missing friends, no extraneous)
singleton communities (no extraneous, no overlap)
low missing neighbors
low overlap
ego-centric
disjoint
clique-like
low extraneous

Figure 3: Community Set Space Canvas

to a node within its home community.

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

Definition 3. Overlap is the number of communities that a node is placed in besides its home community.

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

Using these metrics we can quantitatively judge a community set. Generally, an analyst would probably choose lower values for all three of these metrics however, there is a trade-off. A lower value of one of the metrics will normally result in a larger value for one or both of the others. What constitutes an ideal community set cannot be objectively defined but is specific to a user's needs.

2.2 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 3 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 are zero missing neighbors and zero extraneous nodes. This is defined as the set of all neighborhood communities – that is, for each node, a home community is created, 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.

Community sets that are mapped near to the edges also have distinctive characteristics. Disjoint communities – those with no overlap – are appropriate for partitioning nodes. An example would be placing students into study groups. Clique-like communities, which have low values of extraneous nodes, are those where nearly all nodes in a community are connected to nearly all others. These types of communities occur naturally when people form small, special interest groups. Ego-centric communities – where every node has at least one community, to which it and all of its neighbors belong – have zero missing neighbors. Terrorism experts may be interested in forming ego-centric communities of suspected terrorists with known connections. Each suspect would have at least one community with all of his known connections with the other nodes in the community being possible accomplices. In the middle of the triangle, community sets take on a mixture of the characteristics described above.

3. METHOD

The CHI algorithm was conceived to operate within the community set space as described in Section 2. This section provides the theory behind it and the details of its operation.

3.1 CHI Algorithm

CHI was designed to optimize the objective function:

$$\mathcal{L} = M_{mn} + M_{en} + M_{ol}$$

The input to the algorithm is an initial community set $S = (G, C, H)$ and the output is the (locally) optimal community set $\hat{S} = (G, \hat{C}, \hat{H})$. We begin by restating definitions 1, 2 and 3 in terms of the adjacency, community and home matrices and for particular nodes v_i and v_j:

$$M_{mn}(v_i, v_j) = \left(1 - \sum_{k=1}^{K} (h_{ik} c_{jk})\right) \cdot a_{ij}$$

$$M_{en}(v_i, v_j) = \sum_{k=1}^{K} (h_{ik} c_{jk}) \cdot (1 - a_{ij})$$

$$M_{ol}(v_i) = \sum_{k=1}^{K} c_{ik} - 1$$

The objective function can be rewritten as:

$$\begin{aligned} \mathcal{L} \quad = \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} \left(1 - \sum_{k=1}^{K} h_{ik} c_{jk}\right) a_{ij} + \\ & \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{K} h_{ik} c_{jk} (1 - a_{ij}) + \\ & \sum_{i=1}^{n} \sum_{k=1}^{K} c_{ik} - 1 \end{aligned} \qquad (1)$$

Our approach to optimization is to alternate between improving H and C. In step 1, the C values are held fixed and the H values are changed. Step 2, changes the C values while the H values are fixed.

3.1.1 Step 1

Each node, v_i is placed in one and only one home community, that is $h_{ik} = 1$ for some k and $h_{ix} = 0$ for $x \neq k$. For each node v_i and each community k, we isolate the terms in $\mathcal{L}$ with h_{ik} in them:

$$\sum_{j=1}^{n} (1 - h_{ik}c_{jk})\, a_{ij} + (1 - a_{ij})h_{ik}c_{jk} \tag{2}$$

For v_i we need to set $h_{ik} = 1$ for exact one k and the rest must be zero. With C fixed, to minimize 1 we set:

$$h_{ik} = \begin{cases} 1 & \text{for } \arg\min\limits_{k} \sum_{j=1}^{n} -a_{ij}c_{jk} + (1 - a_{ij})c_{jk} \\ 0 & \text{otherwise} \end{cases}$$

This process moves each node to the community that minimizes the objective function given the current values of C. It should be noted that changing the values of H for v_i will not effect the decision of home community for any other node because we are not changing the value of any terms that contain H values other than for v_i.

3.1.2 Step 2

In the next step we change C while holding H fixed. Recall that C allows for overlapping communities so that there is not just one c_{ik} that can to be set to 1 (but at least one needs to be 1). Like before, we isolate the terms with c_{ik}:

$$\sum_{j=1}^{n} (1 - h_{ik}c_{jk})\, a_{ij} + (1 - a_{ij})h_{ik}c_{jk} + c_{ik} \tag{3}$$

We consider each c_k in C for v_i. Setting $c_{ik} = 1$ can cause Formula 3 can be positive or negative. Since negative values reduce the objective function we set all values of $c_{ik} = 1$ where it is negative. For the case when none of the values of Formula 3 are negative, we set $c_{ik} = 1$ for the minimum value. :

$$c_{ik} = \begin{cases} 1 & \text{for } \sum_{j=1}^{n} a_{ij}h_{jk} - (1 - a_{ij})h_{jk} - 1 > 0 \\ 1 & \text{for } \arg\max\limits_{k} \sum_{j=1}^{n} a_{ij}h_{jk} - (1 - a_{ij})h_{jk} - 1 \\ 0 & \text{otherwise} \end{cases}$$

This process puts node v_i into any community that makes the objective function smaller given the current values of H. Again, changes can be made in C to any node v_i without affecting the other nodes.

Since the decision to change one node will not affect the decisions for the others, the changes can be made to nodes in any arbitrary order. It follows that in each step, the total of the objective function will either decrease or stay the same (if no changes are made).

CHI starts with a random or given initial community set $S = (G, C, H)$ and then to loop through step 1 and step 2 until no more changes are possible. As stated above, after each step either the objective function is reduced or no changes are made so that we are guaranteed to find a local minimum. The details of CHI can be seen in Algorithm 1.

```
input : Initial community set $S = (G, C, H)$
output: Optimum community set $\hat{S} = (G, \hat{C}, \hat{H})$
$\hat{C} = C$;
$\hat{H} = H$;
while no more changes do
    foreach $v_i \in G$ do
        $\hat{h}_i = k$ for $\arg\min_k \sum_{j=1}^{n} -a_{ij}\hat{c}_{jk} + (1 - a_{ij})\hat{c}_{jk}$;
    end
    foreach $v_i \in G$ do
        $\hat{c}_{ik} = 0$;
        for $k \leftarrow 1$ to $|C|$ do
            if $\sum_{j=1}^{n} a_{ij}\hat{h}_{jk} - (1 - a_{ij})\hat{h}_{jk} + 1 > 0$ then
                $\hat{c}_{ik} = 1$;
            end
            $\hat{c}_{ik} = 1$ for
            $\arg\min_k \sum_{j=1}^{n} a_{ij}\hat{h}_{jk} - (1 - a_{ij})\hat{h}_{jk} + 1$;
        end
    end
end
```

Algorithm 1: CHI algorithm

Notice that in the two inner loops in which the values of H and C are reassigned, the order in which the program exams the nodes is not important. In the first loop, the values of H are reassigned using only the network A and the communities C. In the second loop, the C are reassigned using only A again and H. This means that changing one node's home community will not influence another's. The same applies to C.

3.2 Similarity to Kmeans

The Kmeans algorithm [13] separates n samples into k clusters. Each sample x_i is a vector of d data values. Typically, the Euclidean distance is used to compute the distance between the samples and the cluster centers c_j. The algorithm is designed to minimize the objective or error function:

$$E = \sum_{i=1}^{n} \sum_{j=1}^{k} (c_j - x_i)^2$$

The algorithm proceeds by alternating between assigning samples to the nearest center and recalculating the centers until convergence.

The CHI algorithm introduced in this paper has many similarities to the Kmeans algorithm. They both are designed to minimize an objective function by a converging, alternating process. In the CHI algorithm, the H values are the assignments of the nodes to communities, similar to the assignment table used by Kmeans. Each column of the H matrix represents the assignments for one of the k communities. The adjacency matrix A corresponds to the data samples $X = \{x_1 ... x_n\}$, where the neighbors of a node provide evidence of which nodes should be grouped together.

The C matrix corresponds to the data centers. Each column vector of n elements lists the nodes that belong to that community. While this is not really an average of the nodes that are home to that community, it provides evidence to which nodes should be considered to be home to that community. A node that is home to community k_1 but is also assigned in C to k_3 may later be assigned a home community of k_3.

Like Kmeans, CHI must also begin with an initial community configuration. This is a weakness for both algorithms. A poorly chosen initial community set for CHI results in an suboptimal local minimum.

3.3 Complexity

The complexity of the CHI algorithm as described above, is bound first by the number of iterations I, necessary for convergence. Within that loop we alternate between step 1 and step 2 for each of the n nodes. Both of the steps involve summing data for each of the k communities for each of the n possible neighbors. The complexity is thus $O(Ikn^2)$.

For the actual implementation, we chose different structures for the network graph and home communities H. Rather than using the adjacency matrix, we selected the neighbor list format where, for each node, there is a list of its neighbors. Not only does this require less memory but also speeds up the algorithm. Notice in Algorithm 1, the summaries inside the loop must examine all n nodes. With the neighbor list it need only iterate over the nodes neighbors.

For the home communities, instead of using an n by k matrix where each row contains only one 1 value, we chose a vector of n numbers $0...n-1$ representing the community to which it belongs. Using these choices, allows the algorithm to be written more efficiently, specifically in $O(Ikna)$, where a is the average number of neighbors for a node.

3.4 Generalization

One potential weakness of the CHI algorithm is that the choice of objective function is somewhat arbitrary, that is, it weights M_{mn}, M_{en} and M_{ol} equally. To make the algorithm more general we offer the following objective function:

$$\mathcal{L} = \lambda_1 M_{mn} + \lambda_2 M_{en} + \lambda_3 M_{ol}$$

where the lambda values are parameters that the user can enter to shape the communities to their specific needs. For example, if overlap is something one wants to avoid, one could use the values $\lambda_1 = \lambda_2 = 1$ and $\lambda_3 = 10$. In the section above where the algorithm is developed, one can easily substitute the lambda values in the appropriate places.

In Section 4, we provide experiments to demonstrate the effectiveness of using the parameters. In running our experiments, it became clear that even for the small data sets, there were many solutions (local minimums) using the CHI algorithm. For a user who is looking for communities in a network it may be that it is more important to get the right type (ego-centric, disjoint, etc.) of community than the community set with the lowest number of violations.

4. EXPERIMENTS

The intention of the experiments section is to demonstrate the usefulness of the CHI algorithm. It will be shown that CHI

- compares favorably against other algorithms using the metrics of the community set space
- is flexible, by creating overlapping or disjoint communities
- is versatile, creating community sets of differing characteristics
- is fast

Table 1: Data Sets

data set	nodes	links	density
karate	34	78	0.1390
teen	50	77	0.0629
foot	115	613	0.0935
jazz	198	2742	0.1406
webkb	877	1388	0.0036
cora	2708	5278	0.0014
wiki	8297	103689	0.0030
slash	82168	870161	0.0003

4.1 Set up

Many data sets were used in the experiments to provide a variety of small and large sets as well as a low and high density of links. The sets with their attributes are listed in Table 1. All of the sets are non-directional networks. The *karate* [17] set is the famous Zachary karate set from the study of relationships within a karate studio. The *teen* [15] set, a study of the relationships of 50 school girls in Scotland, comes from a collection on the Siena website. 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. The *jazz* musician's dataset [8] represents musicians and their collaboration. Both the *webkb* (web page network) and *cora* (citation network) data sets are from the Linqs [6] web site. The *wiki* (a network built from the votes from wikipedia) and *slash* (taken from the slashdot social network) are from the SNAP website [12].

A number of the experiments compare the CHI algorithm to two other algorithms, *cmn* (or FastCommunity) by Clauset, et al. [2] and spectral clustering, also known as normalized cut (*ncut*), by Shi and Malik [11]. These algorithms were chosen because they are well accepted algorithms and the code was readily available. While there are many overlapping algorithms published, finding the code proved to be more difficult. We did some early tests using the agglomerative method by Tang et al. [14] but the results were not very good and the algorithm was very slow.

For CHI, unless specified, $\lambda_1 = \lambda_2 = \lambda_3 = 1$. Both cmn and ncut take the network as input and deterministically return a set of communities. CHI on the other hand, starts with the network and an initial community set $S = \{C, h\}$ and from that produces an improved community set. Since the results of the algorithm are dependent on the initial (random) community set chosen, the tests that compare it to the other algorithms average the results of 10 runs. In multiple comparisons, the results of 10 runs were fairly consistent.

4.2 Comparison of Algorithms

The experiments in this section compare the results of the different algorithms against the others using the data sets and three different levels of grouping, i.e. different values of k. The results will be presented in two sets of charts. The first set compares the algorithms using the total number of violations ($M_{tot} = M_{mn} + M_{en} + M_{ol}$). In these charts we used the entire community set of overlapping communities from CHI, $S = \{G, C, h\}$. The second set of charts uses modularity for comparison. Because modularity is a measure for only disjoint communities, we used only the home communities, h from CHI.

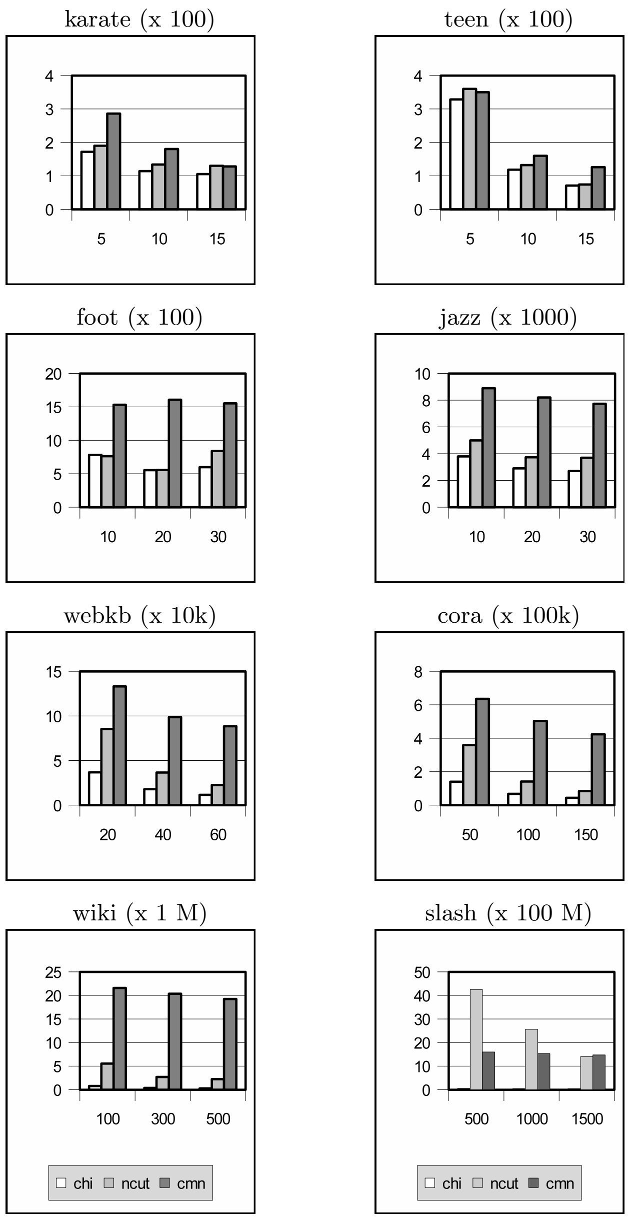

Figure 4: Comparison of CHI to ncut and cmn. The horizontal axis groups the algorithms by k (nbr of communities). The vertical axis is violations. Notice the violations are scaled for each data set, e.g. hundreds for karate.

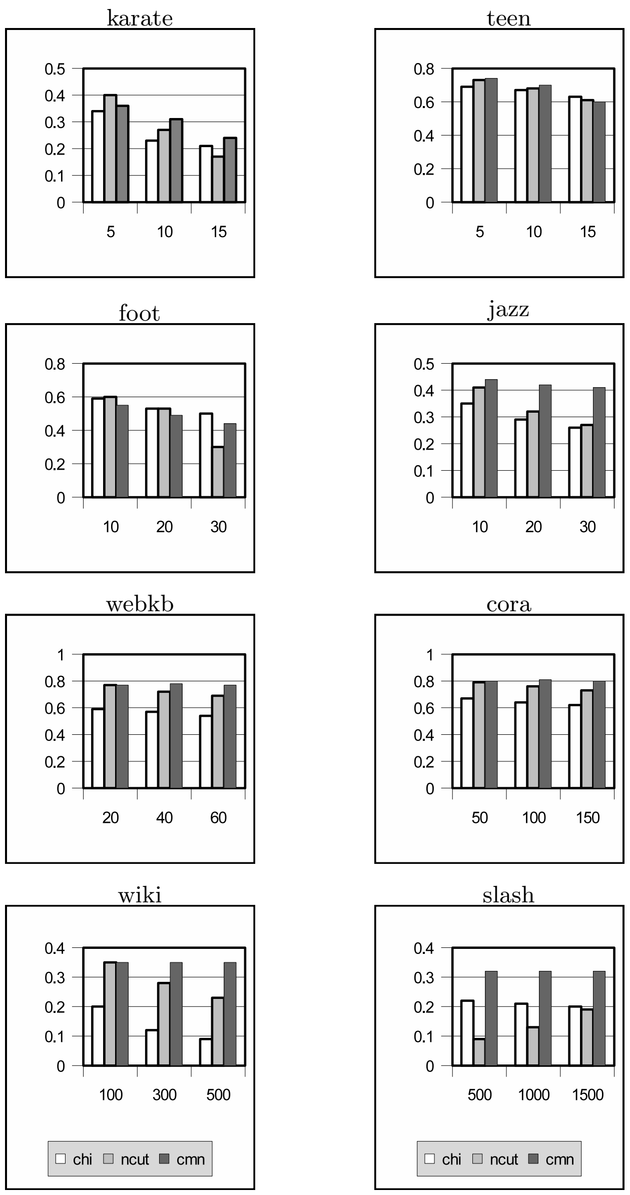

Figure 5: Comparison of disjoint CHI to other algorithms using modularity.

The first charts, in Figure 4, show a chart for each data set. The legends in the bottom two charts (wiki and slash) apply to all of the charts. The bars represent the various algorithms, grouped by value of k. The height of the bars is the number of violations. The violations have been scaled to make them easier to read - thus the number of violations for karate, using cmn with 5 communities is close to 300.

In all of the tests, the only case where CHI did not have the lowest number of violations was with foot, with $k = 10$. The CHI algorithm does not guarantee to find the community set with the absolute lowest number of violations, but it does find a local minimum. Given the large number of possible community sets there are probably many, many local minima. In this particular case, the ncut algorithm found a community set that had fewer violations than many of the local minimums found by CHI (the experiment was repeated several times with similar results).

Outside of this one case, CHI consistently finds community sets with fewer violations than ncut or cmn. With the larger sets, the results are particularly striking. The results of the experiments were also compared using just the disjoint community sets (h) of CHI with nearly identical results. These results are omitted due to space constraints.

The second set of charts, in Figure 5, compares the results of the disjoint communities of CHI against the results of ncut and cmn. These charts look similar to the previous

Table 2: Mean and std. dev. of size of communities

			std dev		
dataset	k	mean	chi	ncut	cmn
karate	5	6.8	1.2	3.1	5.6
	10	3.4	1.2	1.4	3.3
	15	2.3	1.6	0.9	1.7
teen	5	10.0	1.6	3.5	3.2
	10	5.0	1.2	1.7	2.7
	15	3.3	1.5	1.3	2.3
foot	10	11.5	1.5	2.1	9.2
	20	5.8	3.8	3.6	8.4
	30	3.8	3.8	2.1	6.9
jazz	10	19.8	7.2	16.0	28.5
	20	9.9	6.7	8	21
	30	6.6	5.8	6	17
webkb	20	43.9	1.1	50	70
	40	21.9	0.9	22	45
	60	14.6	0.8	14	36
cora	50	52.2	0.5	67	101
	100	27.1	0.7	28	66
	150	18.1	0.8	17	51
wiki	100	83.0	7.1	220	458
	300	27.7	4.7	90	260
	500	16.6	3.1	64	196
slash	500	164.3	3.9	2911	1782
	1000	82.2	2.8	1598	1235
	1500	54.8	3.9	967	990

ones, except that they measure modularity. Modularity is the fraction of edges that fall within communities minus the expected fraction. It has a maximum of 1 and can be negative. Sets that are closer to 1 are considered better than ones that are lower.

In these charts, the taller bars are the ones that are better. With only a few exceptions, the CHI sets score lower than either ncut or cmn and sometimes both. This is not surprising as the cmn algorithm is designed to maximize the modularity metric. Since modularity is a measure only of the fraction of edges within a community, higher scores are given to sets with lower M_{mn}. It does not consider either M_{en} or M_{ol}.

While CHI did not do as well as the other two algorithms it did better than chance (zero). This is not surprising as it is not designed to optimize the modularity. In a later section, it will be shown that the algorithm can be tuned by changing the values of λ_1,λ_2 and λ_3. By tuning the lambdas, community sets can be generated that have much better modularity scores.

4.3 Distribution of nodes across communities

The way in which nodes are distributed amongst the communities is another characteristic of community sets that may interest analysts. One analyst may desire a balanced spread of nodes across the communities, while another may not care about the spread as long as M_{mn} is minimized.

Much depends on the network. For example, given a network with a fifty node clique and 10 singleton nodes, we would expect that with $k = 11$ an algorithm would place the clique in one community and each singleton in its own community. For a regular network, we would expect an algorithm to find a more balance set of communities.

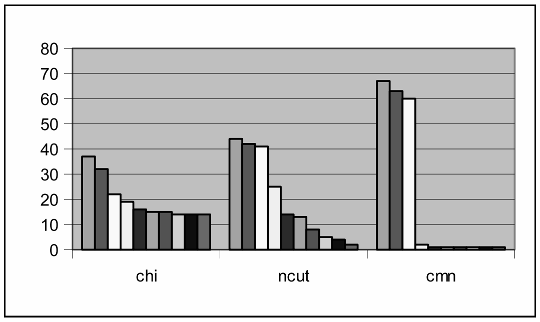

Figure 6: Spread of communities for jazz with $k = 10$. The bars within each algorithm group represent communities with the vertical axis measuring the number of nodes in a community. Bars are ordered from largest community to smallest within algorithm.

Because CHI minimizes the three violation metrics (relative to the λ values), with $\lambda_1 = 1$, $\lambda_2 = 1$ and $\lambda_3 = 1$, the algorithm produces fairly balanced communities. Table 2, shows the results of calculating the standard deviation of the sizes of the communities. With a few exceptions, CHI has a smaller standard deviation. With the larger data sets, the difference is profound.

Since most algorithms attempt to find communities by grouping linked nodes together, large communities of densely linked nodes are inevitable. CHI, too, will create large communities if the nodes are clique-like. However, because it is designed to balance the three metrics M_{mn}, M_{en} and M_{ol}, it produces more balanced communities. Figure 6 shows the communities for jazz and $k = 10$. Clearly the ones for CHI are much more balanced than the other two, which have about 3 larger communities and several smaller ones.

Of course, by changing the lambda values, one will also change the distribution of nodes. For instance, if λ_1 is much greater than the other two, the community distribution will be less uniform. Then, one would expect the communities to look more like the ones produced by an algorithm like cmn.

4.4 Tuning

One of the advantages to CHI is the flexibility to find community sets with desired characteristics by tuning the lambda parameters. For these experiments we chose to show the results from the jazz dataset as they were a little more expressive than the others.

In Table 3, one can see the results from CHI using seven different sets of values for the lambdas. We chose these values as they represent a balanced approach (1,1,1) and vectors in six different directions of the triangle. Figure 7 shows where these community sets would be plotted on the community set space canvas.

Note that the points are not placed as ideally as one would hope, e.g. (10,10,1) is not in the lower left corner. The way the sets appear on the canvas has to do with the network and the value of k. For any given data set, as k gets smaller, the possible solutions become closer to the upper corner. The type of network will influence the community sets that CHI produces, placing them in different locations on the canvas.

Changing the lambda values can produce sets more like

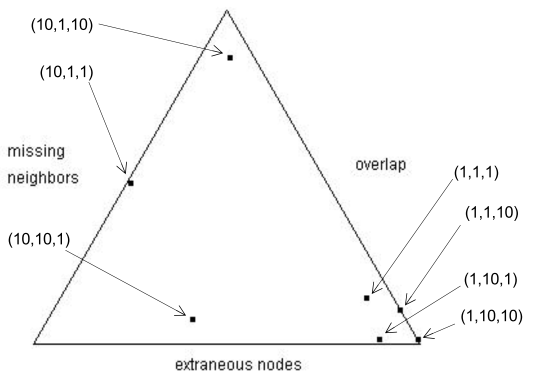

Figure 7: Plot of community sets for jazz with $k = 10$ and different lambda values

Table 3: Community set violations for different lambda values

λ_1	λ_2	λ_3	M_{mn}	M_{en}	M_{ol}
1	1	1	2131	350	190
10	1	1	5	2871	3051
1	10	1	2585	1	298
1	1	10	2856	290	8
10	10	1	1050	184	1594
10	1	10	428	5519	378
1	10	10	3556	2	0

cmn and improve the modularity score. Since cmn produces zero overlap and does not consider extraneous nodes, we set $\lambda_1 = 1000$, $\lambda_2 = 1$ and $\lambda_3 = 10000$. We do not present all of the results here but for the webkb set, it improved the modularity score for $k = 20$ from 0.59 to 0.73.

4.5 Complexity

As stated in the previous section, the complexity of CHI is $O(Ikna)$. In tests, the number of iterations, I ranges from 3 to 5 for the smaller datasets and from about 4 to 20 for the larger ones. The average number of neighbors a depends upon the network. So for sparse networks, the algorithm is closer to $O(kn)$.

For the four small networks, each of the algorithms could find communities in less than a second. Figure 8 summarizes the run times for the other data sets. The chart is scaled at a maximum of 1800 seconds, so the results for ncut on the slash data set run off of the top of the chart. The time $k = 500$ was about 2000 seconds and for $k = 1500$ it was over 14 hours. Under many conditions CHI will not be as fast as cmn. Notice in the chart that cmn has nearly constant time for the slash set. As k is increased, CHI will get progressively slower, so that cmn has a speed advantage for large values of k. There have also been some improvements to cmn [cite] that have made it even faster.

5. RELATED WORK

There have been many community finding algorithms proposed; it is not our intention to review each one here. The reader is directed to one of the recent reviews [9, 5, 16].

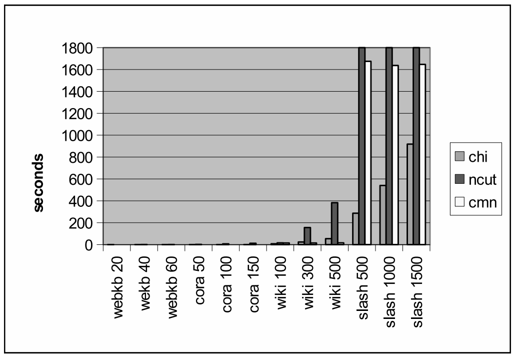

Figure 8: Comparison of runtimes

There are many ways in which the algorithms can be organized: overlapping vs. disjoint, local vs. global, approach (agglomerative, iterative, divisive, etc). They are organized here in how they fit into the community set space, that is, the amount of M_{mn}, M_{en} and M_{ol} their communities produce. It should be noted that a simple way to control M_{mn} and M_{en} is to vary k. Our implied intention is that we are comparing the algorithms using the same number of communities.

Algorithms that find disjoint communities implicitly hold M_{ol} to zero. Some then attempt to minimize M_{mn}. One of the first is the algorithm by Girvan and Newman [4] which uses the betweenness metric to remove edges to reveal communities. Starting with a single large community (top corner of the canvas), it separates the graph into communities, moving down the left edge until it has reduce the network to singletons (bottom right corner). Any divisive algorithm that creates disjoint community sets will follow the same path.

The improved algorithm by Clauset, et al. [3], starts with singletons and merges them based on the modularity metric until they are all merged into one community. This creates community sets that follow the same disjoint edge in the opposite direction from the divisive algorithms. This will be true for any agglomerative approach that starts with singletons.

Other disjoint algorithms are not designed to minimize M_{mn} but appear to detect communities with low values of both M_{mn} and M_{en}. Spectral [11] methods cluster the eigenvector components of nodes. As a result, it is more likely to group connected node pairs while separating unlinked pairs. There are many other disjoint algorithms [5] that appear to have a similar, balanced approach. It should be noted that while we did not come across any disjoint algorithms that minimized M_{en}, it is easy to imagine an agglomerative algorithm that starts with singletons and merges them based on minimizing M_{en}.

The divisive and agglomerative approaches can also be applied to ego-centric communities. In particular, the approach by Tang, et al. [14], starts with the neighborhood communities (in the lower left corner of the canvas) and merges the communities based on the Jaccard index (using overlap) until it reaches a single community. This is essentially holding $M_{mn} = 0$ while minimizing M_{ol}. It can be shown [10] that a community set that is ego-centric (no missing neighbors) can have two communities merged and the resulting community set will also be ego-centric. So similarly to the disjoint agglomerative methods, this one creates sets that move along an edge of the canvas but the ego-centric edge instead of the disjoint one.

Any approach that creates cliques will be placed on the bottom edge of the canvas. The CFinder [7], algorithm starts with cliques of a certain size and then merges them. The resulting communities are not cliques but are very close. We are not aware of any algorithms that start with singletons and copy nodes into communities until they become neighborhood communities (or the other direction) but such an algorithm is not inconceivable.

Other algorithms detect community sets that would be plotted in the interior of the canvas. Where exactly the sets will appear depends on the emphasis that the algorithm places on the three metrics and the number of communities found. There are many such algorithms listed in [16]. While all of the community finding algorithms that we reviewed, in one way or another, minimized one or a combination of two or all three of the metrics, none of them explicitly minimized them all.

One last note should be made about fuzzy clustering. Overlapping sets can be crisp (where every node is either in or not in a community) or fuzzy (where nodes belong to a community in proportion to a weight). The metrics defined in Section 2 cannot be applied to fuzzy sets. One could get around this problem by applying a threshold to convert them to crisp sets. Another way would be to modify the metrics but it is not clear how one would determine overlap for fuzzy sets.

6. CONCLUSIONS

The CHI algorithm was proposed in this paper to optimize the metrics that define the community set space. The algorithm starts with a random community set and makes changes to optimize the metrics until it reaches a local minimum. It was shown that CHI is very similar to the Kmeans clustering algorithm except in the way that it represents the characteristics (averages in the case of Kmeans) of the communities. It was demonstrated to be fast and performed favorably against other well-known algorithms.

CHI produces two community structures, one overlapping and the other disjoint while other community finding algorithms return only overlapping or disjoint. This leads to another advantage of CHI. Since it starts with a seed community set, one can either choose to have it use a randomly chosen set or use the result of another algorithm as input. This allows one to use CHI to further tune the result of another algorithm. In addition, it can also be used to turn a disjoint algorithm into an overlapping one and vice versa. One could, in theory take the output of say, spectral clustering, and use it in CHI for one iteration and then use the overlapping community output.

Many algorithms allow the user to decide on the number of communities (k), as CHI does. In addition, CHI has parameters (λ_1, λ_2 and λ_3) that coerce the algorithm to find community sets with certain characteristics. For example, an analyst may be interested in disjoint communities or ego-centric communities.

The CHI algorithm is based on our contention that finding the best community set is a combination of low violations and type of set. An analyst should first of all determine what characteristics they are interested in (overlap, missing neighbors and extraneous nodes) and find the community set that best fits their interests.

7. REFERENCES

[1] H. Becker and John Riordan. The arithmetic of bell and stirling numbers. *American Journal of Math*, 70:385–394, 1934.

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

[3] A. Clauset, C. Moore, and M. E. J.Newman. Structural inference of hierarchies in networks. In *Proceedings of the 23rd International Conference on Machine Learning (ICML), Workshop on Social Network Analysis*, 2006.

[4] M. Girvan and M. Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci*, 99:7821–7826, 2002.

[5] A. Lancichinetti and S. Fortunato. Community detection algorithms: a comparative analysis. *Physical Review E*, 80, Sep 2009.

[6] Statistical relational learning group. http://www.cs.umd.edu/linqs/.

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

[8] P.Gleiser and L. Danon. Community structure in jazz. *Adv. Complex Syst*, 6:565, 2003. http://deim.urv.cat/ aarenas/data/welcome.htm.

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

[10] J. Scripps. Exploring the community set space. In *IEEE/WIC/ACM International Conference on Web Intelligence*, 2011.

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

[12] Stanford large network dataset collection. http://snap.stanford.edu/data/.

[13] P. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining.* Addison Wesley, Inc., 2005.

[14] 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.

[15] Siena network statistical analysis program. http://stat.gamma.rug.nl/snijders/siena.html.

[16] J. Xie, S. Kelly, and B. Szymanski. Overlapping community detection in networks: the state of the art and comparative study. *CoRR*, abs/1110.5813, 2011.

[17] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.