6-2018

The Difference between Optimal and Germane Communities

Jerry Scripps
Grand Valley State University, leidigp@gvsu.edu

Christian Trefftz
Grand Valley State University, trefftzc@gvsu.edu

Dr. Zachary Kurmas
Grand Valley State University, kurmasz@gvsu.edu

Social Network Analysis and Mining (2018) 8:44
https://doi.org/10.1007/s13278-018-0522-1

ORIGINAL ARTICLE

The difference between optimal and germane communities

Jerry Scripps[1] · Christian Trefftz[1] · Zachary Kurmas[1]

Received: 3 May 2017 / Revised: 4 June 2018 / Accepted: 7 June 2018

Abstract
Networks often exhibit community structure and there are many algorithms that have been proposed to detect the communities. Different sets of communities have different characteristics. Community finding algorithms that are designed to optimize a single statistic tend to detect communities with a narrow set of characteristics. In this paper, we present evidence for the differences in community characteristics. In addition, we present two new community finding algorithms that allow analysts to find community sets that are not only high quality but also germane to the characteristics that are desired.

Keywords Community finding · Networks · Link analysis

1 Introduction

Networks are used in programs to represent the complex relationships that occur in social, biological, computer and other networks. These networks often exhibit community structure. A community set (commSet) refers to a particular set of communities for a network. There is general agreement that high quality communities are ones that have many links (or edges) within the communities and fewer of them between the communities. While there are many community finding algorithms (Porter et al. 2009; Lancichinetti and Fortunato 2009; Xie et al. 2011), an exact definition of a high quality set of communities for a network is elusive.

Existing community finding algorithms typically are designed to optimize a specific function. While these algorithms find high quality commSets, this paper presents an argument for searching for commSets that are not only of a high quality but also have characteristics that are germane (i.e. appropriate) to the network and the purposes of the user. Considering all of the possible commSets for a given network, some different characteristics will emerge. For a small social network, one could place nodes into sets to approximate bipartite sets, maximimal cliques, min-cut partitions or something altogether different.

In this paper, three simple statistics—collectively named NEO—are used to map commSets onto a triangular canvas that distinguish their characteristics. In addition, objective functions using NEO will be used to formulate two algorithms for detecting commSets.

The three statistics of NEO are missing neighbors (M_{mn}), extraneous nodes (M_{en}) and overlap (M_{ol}). A missing neighbor is counted when a node in one community is linked to a node in a different community—in Fig. 1, k is a missing neighbor of c. An extraneous node is counted for any node in a community that is not linked to another node in the same community. n is an extraneous node with respect to m. Overlap is counted for each additional community that a node is assigned to (beyond its first or home community). d is has an overlap of 1 since it is in both comm1 and comm2. This description is simplified; the actual definitions are presented in Sect. 3. For the remainder of this paper, a NEO score will be an ordered triplet (e.g. {2, 10, 0}—2 missing neighbors, 10 extraneous nodes and 0 overlap).

NEO is the foundation that will be used to find germane communities:

1. It is simple to understand. Given a data set with two commSets with NEO scores of {75, 254, 0} and {57, 241, 20} and the same number of communities, one can tell that the first is a set of disjoint and the second is overlapping. Further, the first has more missing neighbors than the second—thus the second has communities that are more tightly connected. There is often a tradeoff with the metrics; allowing a higher value in one can result in lower values of the others.

✉ Jerry Scripps
scrippsj@gvsu.edu

[1] Grand Valley State University, Allendale, USA

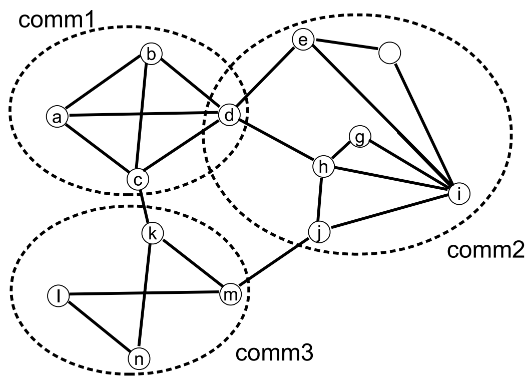


Fig. 1 Example network

2. NEO will be used to define the commSet space canvas (described in Sect. 3.3). In the previous example, plotting the two commSets on the canvas provides a convenient visual map for comparing the two commSets.
3. Using the two algorithms described later, which use objective functions incorporating NEO, analysts can tune the algorithms so that they will find good commSets with the desired characteristics.

To illustrate, one statistic for purchasing a new computer would be the ratio of RAM to price. It does not make sense to simply maximize this statistic if one is looking for a specific kind of computer (laptop, server, etc.). commSets can also have different characteristics. Community finding algorithms tend to find sets of a specific type, i.e. with different characteristics. While it is important to find good quality communities, priority should also be placed on finding the kind of communities that are desired. The motivation for finding communities with specific characteristics is discussed in Sect. 3.3.

The algorithms proposed used two different methods to find high quality communities with the desired characteristics. The first, 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 commSet space, it has fewer violations than any of the other methods tested. Third, like kmeans, CHI starts with an input commSet. Unlike kmeans, it is flexible in that it has parameters that can be tuned to produce commSets with many different characteristics.

As stated above, CHI uses a seed or random input commSet and then finds a local optimum according to the parameters given. While it is effective at finding high quality commSets they may not have exactly the desired characteristics. The second algorithm, Gamit, is designed to find high quality commSets with characteristics very close to those desired. Gamit has similarities to the agglomerative clustering method. One can use Gamit to find a seed commSet for CHI, which will then find the local optimum.

Portions of this paper were published in Scripps (2011), Scripps and Trefftz (2013). For this journal paper, the initial experiments in Sect. 5.2 were added as were many of the experiments. Gamit with its derivation and experiments is also new. The authors have also posted a Java version of CHI and a stand-alone tool for analyzing networks that incorporates both Gamit and CHI at http://www.cis.gvsu.edu/~scrippsj/pubs/software.htm.

After this introduction, related work is presented in Sect. 2. Necessary terms, metrics and the commSetSpace canvas will be defined in Sect. 3. The algorithms are defined in Sect. 4 and experiments are in Sect. 5. The paper ends with a section for conclusions.

2 Related work

Networks are often given characterizations based on statistics (such as clustering coefficient) which describe a growth model. In particular, there are models for random, small world, and scale-free, among others. We know that networks having one of these designations will have certain characteristics that can be helpful in analysis. While there have been studies to examine the characteristics of individual communities (see Traud et al. 2011), we are not aware of any attempt to characterize an entire set of communities.

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 (Porter et al. 2009; Lancichinetti and Fortunato 2009; Xie et al. 2011). 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 commSet 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.

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 (2002) 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 commSets will follow the same path.

The improved algorithm by Clauset et al. (2006), starts with singletons and merges them based on the modularity metric until they are all merged into one community. This creates commSets 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 necessarily designed to minimize M_{mn} but appear to detect communities with a balance of M_{mn} and M_{en}. Spectral (Shi and Malik 2000) 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 (Lancichinetti and Fortunato 2009) 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. (2010), 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 $M_{\text{mn}} = 0$ while minimizing M_{ol}. It can be shown (Scripps 2011) that a commSet that is ego-centric (no missing neighbors) can have two communities merged and the resulting commSet 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.

There are some algorithms that detect algorithms on the bottom edge of the canvas. Simply finding cliques would result in commSets placed there. However, the CFinder (Palla et al. 2005), algorithm starts with cliques of a certain size and then merges them. The resulting communities often have very low M_{ol} so they are close to the right edge of the triangle. 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. The algorithm by Ahn et al. (2010) partitions the links instead of the nodes—the nodes are then added to the communities to which their associated links belong. This algorithm tends to generate many communities with a high level of overlap.

3 Notation and metrics

Nodes in networks can be grouped together into sets and are often referred to as communities. Intuitively, they are typically grouped in a way that agrees with the link structure—that is, there should be many links between nodes within the same community and fewer between nodes in different communities. The commSetSpace is a framework that is useful for measuring the quality and characteristics

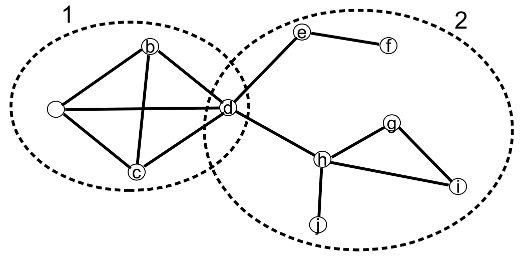


Fig. 2 Network to describe the metrics of the commSetSpace

of a specific set of communities. It will be defined below, after defining the structures and statistics necessary for the commSetSpace definition.

3.1 Notation and structures

Nodes in networks can be placed into disjoint or overlapping communities. Disjoint communities are ones where each node is placed in one, and only one, community. Overlapping communities allow nodes to be placed in one or more communities. The commSetSpace represents all possible commSets, which is the same as all possible overlapping commSets. This set of all possible commSets, has as a subset, the set of all possible disjoint commSets.

As will be seen later, the commSetSpace maintains both disjoint (referred to as home communities) and overlapping communities (communities). This is necessary for the algorithms that will be described later. We begin with a formal description of networks and community structures.

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, $c_i = \{v_j, \dots v_m\}$, 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 *commSet* $S = \{G, C, h\}$ is a triplet where $C = \{c_1, \dots, c_k\}$ is a collection of k communities and h is a home community function. 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. Furthermore, we shall also represent both the communities and home communities by 0/1 matrices. For communities, $C = [c_{ij}]_{n \times k}$ where $c_{ij} = 1$ if v_i is in community j. Likewise, for home communities, $H = [h_{ij}]_{n \times k}$.

Figure 2 will be used to illustrate the structures and NEO. The two community structures are shown in Table 1. Note that the home (disjoint) communities place nodes –d in community 1 and nodes e–j in community$a$ 2. To the right of that is the (overlapping) community structure which looks the same except that node d is in both communities 1 and 2.

Table 1 Community structures

Node	Home		Comm	
	1	2	1	2
a	1	0	1	0
b	1	0	1	0
c	1	0	1	0
d	1	0	1	1
e	0	1	0	1
f	0	1	0	1
g	0	1	0	1
h	0	1	0	1
i	0	1	0	1
j	0	1	0	1

3.2 NEO

The statistics chosen for the commSetSpace are considered violations, so that a smaller number is better than a larger number. The choice of the statistics was driven by a desire for high quality communities and for a system that can visually represent the spectrum of commSet types. Generally, we consider high quality communities as those with few between-community links, few within-community non-links and low overlap.

While the commSetSpace, the algorithms and this whole paper, considers only undirected networks, it is important to understand that the metrics are calculated for both directions of the link. Even though there is an undirected link between two arbitrary nodes, v_i and v_j, there might be a violation in the direction of v_i to, v_j but not from v_j to v_i.

The commSetSpace is defined by the following statistics for the nodes v_i, v_j,

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

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

In the example network, the link from *a* to *b* is not a missing neighbor because *b* is in *a*'s home community. From *b* to *a* is also not a missing neighbor as *a* is in *b*'s home community. Consider node *d*, though. It's home community is 1 but it is in both communities 1 and 2. Like the link from *a* to *b*, the link from *b* to *d* is not a missing neighbor because both *b* and *d* are in the same home community. It is different for the link from *d* to *e*. Since *d*'s home community is 1 and *e* is not in community 1 that is considered a missing neighbor. In the other direction, node *d* is in community 2 which is *e*'s home community so that is not considered a missing neighbor.

Definition 2 Extraneous nodes are nodes not directly linked to a node within its home community.

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

There are no extraneous nodes in community 1 of the example network, because all nodes are linked to all other nodes in that community. In community 2, though, there are many extraneous nodes. Since there is no link between node *f* and node *h*, and they are both in the home community of 2, node *h* is an extraneous node with respect to *f* and *f* is extraneous with respect to *h*. Again, consider node *d*. There is no link between *f* and *d*, node *f*'s home community is 2 and *d* is in 2, so *d* is extraneous with respect to *f*. However, since *d*'s home community is 1, node *f* is not extraneous with respect to *d*.

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

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

The only overlap in the example network is node *d*, so there is a total overlap of 1.

Using these metrics we can quantitatively judge a commSet. Generally, an analyst would probably choose lower values for all three of these metrics however, there is a tradeoff. 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 commSet cannot be objectively defined but is specific to a user's needs. Later, in describing the CHI algorithm, weights will be incorporated to prioritize the violations.

While there are other statistics for measuring the quality of communities it is not the intention here to show that NEO is a better statistic. It is valuable because it allows the communities to be charted according to their characteristics and because algorithms that optimize it can be tuned to find sets with specific characteristics.

3.3 commSetSpace canvas

The commSetSpace canvas, as shown in Fig. 3, is a two dimensional chart for plotting commSets. It is 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.

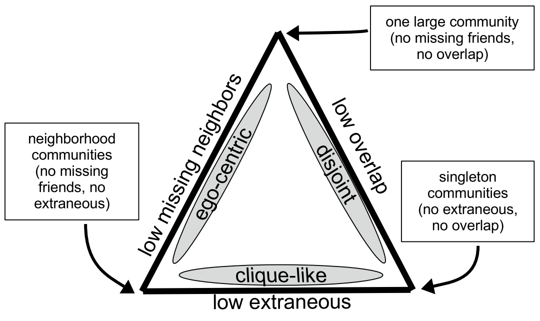

one large community (no missing friends, no overlap)
neighborhood communities (no missing friends, no extraneous)
singleton communities (no extraneous, no overlap)
low missing neighbors
ego-centric
low overlap
disjoint
clique-like
low extraneous

Fig. 3 commSetSpace canvas

The top point, where there is zero overlap and zero missing neighbors would be the commSet defined by one large community. The point in the lower left is the set of all neighborhood communities—that is, each node has a home community consisting of it and its neighbors. The point in the lower right is the set of singleton communities.

The commSets that are mapped near to the edges also have distinctive characteristics. Disjoint communities—those with no overlap—are appropriate for partitioning nodes. Clique-like communities are those where nearly all nodes in a community are connected to nearly all others. Ego-centric communities are ones every node has at least one community, to which it and all of its neighbors belong.

Here we provide some examples of situations where commSets with specific characteristics are desired. Disjoint commSets are applicable in cases where nodes cannot be physically separated into more than a single community. For example, when considering a network of computer equipment one might wish to have the devices assigned to a particular community for purposes of oversight and maintenance. The clique-like communities at the bottom of the canvas occur naturally when people form small groups within social networks. Finding these communities in a network such as Facebook would reveal the many groups that form around special interests. 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.

4 Method

As stated in the previous section, high quality communities should have low values for all three NEO metrics. Towards this goal, the following objective function is proposed:

$$\mathcal{L} = M_{\mathrm{mn}} + M_{\mathrm{en}} + M_{\mathrm{ol}} \tag{1}$$

In Sect. 4.3 it will be generalized to allow the user to weight the metrics according to the kind of communities desired.

Using the concept of home and overlapping communities, two algorithms will be proposed to optimize the objective function. The first, CHI, is an EM-like algorithm that alternates between improving the home and overlapping communities. The second, Gamit, is an agglomerative approach. Each has its merits and they work well together as discussed in Sect. 4.3.

4.1 CHI algorithm

The input to the CHI algorithm is an initial commSet $S = (G, C, H)$ and the output is the (locally) optimal commSet $\hat{S} = (G, \hat{C}, \hat{H})$. CHI was designed to optimize the objective function $\mathcal{L} = M_{\mathrm{mn}} + M_{\mathrm{en}} + M_{\mathrm{ol}}$ which can be rewritten as:

$$\begin{aligned}\mathcal{L} = &\sum_{i=1}^{n}\sum_{j=1}^{n}\left(1 - \sum_{k=1}^{K} h_{ik}c_{jk}\right)a_{ij} \\ &+ \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{K} h_{ik}c_{jk}\left(1 - a_{ij}\right) \\ &+ \sum_{i=1}^{n}\sum_{k=1}^{K} c_{ik} - 1\end{aligned} \tag{2}$$

The 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.

4.1.1 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:

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

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 2 we set:

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

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.

4.1.2 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}:

$$\sum_{j=1}^{n} (1 - h_{ik}c_{jk})a_{ij} + (1 - a_{ij})h_{ik}c_{jk} + c_{ik} \tag{4}$$

CHI starts with a random or given initial commSet $S = (G, C, H)$ 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 commSet $S = (G, C, H)$
output: Optimum commSet $\hat{S} = (G, \hat{C}, \hat{H})$
$\hat{C} = C$;
$\hat{H} = H$;
while no more changes do
    foreach $v_i \in G$ do
        $\hat{h}_{ix} = 0, \forall x$;
        $\hat{h}_{ik} = 1$ 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
        for $k \leftarrow 1$ to $|C|$ do
            $\hat{c}_{ik} = 0$;
            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
```

We consider each c_k in C for v_i. Setting $c_{ik} = 1$ can cause Formula 4 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 4 are negative, we set $c_{ik} = 1$ for the minimum value:

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

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).

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.

4.1.3 Similarity to Kmeans

The Kmeans algorithm (Tan et al. 2005) 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 $X = \{x_1 \dots x_n\}$, 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.

4.1.4 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 to use the neighbor list format rather than the adjacency matrix. This require less memory and 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 we chose a vector of n numbers $0 \dots n-1$ 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.

4.2 Gamit

Gamit is an agglomerative algorithm (see Tan et al. 2005; Jain and Dubes 1988), which starts with every node in a community by itself. It then merges communities base on minimizing the objective function in Eq. 2. Unlike a typical agglomerative algorithm, it merges two sets of communities, the home and overlapping communities. Since membership in the home communities is unique (a node is placed in one and only one community), merging them is straightforward. Rather than merge the overlapping communities, it is convenient to simply perform a single iteration of step 2 of the CHI algorithm.

4.2.1 Algorithm

Details for Gamit are in Algorithm 2. The input to Gamit is the graph, $G = (V, E)$ and optionally k, the desired number of communities. H is initialized to an $n \times n$ matrix with 1s in the diagonal (each node in its own community). The main while loop merges communities until it is left with k communities. The algorithm then finds the two best communities, i and j to merge. Then it calls the merge function that combines column i and j of $\hat{H}$ (or's the values).

In the second part of the loop, it creates a new, empty matrix for C. The remainder of the loop, is the second half of the CHI algorithm, adding nodes to the overlapping communities where appropriate.

```
input  : Graph $G = (V, E)$, number of communities $k$
output: Optimum commSet $\hat{S} = (G, \hat{C}, \hat{H})$
$n = |V|$;
$\hat{H} = empty(n, n)$;
for $i = 1; i < n; i++$ do
    $h_{ii} = 1$
end
$\hat{k} = n$;
while $\hat{k} > k$ do
    $i, j = \arg\min_{i,j} \sum_{u=1}^{n} \sum_{v=1}^{n} -a_{uv}\hat{h}_{ui}\hat{c}_{vj} + (1 - a_{uv})\hat{h}_{ui}\hat{c}_{vj}$;
    $\hat{H} = merge(\hat{H}, i, j)$;
    $\hat{C} = empty(n, \hat{k})$;
    foreach $v_i \in G$ do
        for $k \leftarrow 1$ to $|C|$ do
            $\hat{c}_{ik} = 0$;
            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
    $\hat{k} = \hat{k} - 1$;
end
```

Note that the algorithm can be modified so that instead of stopping at the input k, it can evaluate each step, to find the best value of k. In our tests we used the lowest value of NEO as an optimum. In Sect. 4.3, another suggestion will be made for an optimal number of communities.

4.2.2 Complexity

Implemented like the agglomerative clustering algorithm, Gamit has a complexity of $O(n^2 \log n)$.

4.3 Generalization

The algorithm weight M_{mn}, M_{en} and M_{ol} equally. To make the algorithms more general the following objective function

$$\mathcal{L} = \lambda_1 M_{\mathrm{mn}} + \lambda_2 M_{\mathrm{en}} + \lambda_3 M_{\mathrm{ol}} \tag{5}$$

can be used where the lambda values are parameters that the user can enter to shape the communities to their specific needs. As an example, to find communities with little or no overlap and an emphasis on low missing neighbors, one could use $\lambda_1 = 0.9, \lambda_2 = 0.1, \lambda_3 = 1.0$.

Looking at the two algorithms, the λ values can be inserted where there are expressions involving a and h or c. For example, in the CHI algorithm, replace

$$-a_{ij}\hat{c}_{jk} + (1 - a_{ij})\hat{c}_{jk}$$

with

$$-a_{ij}\hat{c}_{jk}\lambda_1 + (1 - a_{ij})\hat{c}_{jk}\lambda_2$$

CHI and Gamit behave differently and can be used for different purposes. CHI always finds a local minimum for a given starting set of communities. Gamit finds a good commSet but not usually the local minimum. However, when it is generalized according to the suggestions above, it find a good solution with NEO metrics close to the ones desired (according to the input λs)—in other words, it is in the right part of the triangle. CHI is not as good at finding the communities with the desired properties because it depends on the input communities which could be random. If a user is interested in finding the best (lowest NEO) commSet with the properties near to the input λs, good results can be obtained by using Gamit to find the initial commSet as input to CHI.

5 Experiments

The central theme of this paper is summarized by two claims:

1. Different commSets have different characteristics and in comparing commSets, analysts should be concerned not only with the quality of the sets but also the characteristics.
2. The algorithms Gamit and CHI are effective at finding commSets that are both germane (having the desired characteristics) and of a sufficiently high quality.

The intention of the experiments section is to demonstrate evidence for the two claims.

The rest of this section is separated into a subsection to describe the algorithms and data sets, and two other subsections to show the need for Gamit and CHI and to show the effectiveness of the algorithms. The section concludes with some tests on larger networks and a discussion of the scalability of the algorithms.

5.1 Data sets and algorithms

Many data sets were used in the experiments to provide a variety of small and medium-large sets as well as link structures. The sets with their attributes are listed in Table 2. All

Table 2 Datasets with relevant metrics

	Dataset	n	Edges	Degree		Clust. coef.	Path length	Power law	References
				Avg	Max				
1	Tina	11	32	5.82	8	0.652	1.322	1.00	Pajek datasets (2018)
2	Ragusa	24	58	4.83	14	0.433	2.007	2.742	Pajek datasets (2018)
3	Karate	34	78	4.59	17	0.588	2.337	2.524	Zachary (1977)
4	Risk	42	81	3.86	6	0.542	4.381	1.203	
5	Teen	50	77	3.08	7	0.523	2.44	1.76	West and Sweeting (1995)
6	Lesmis	77	254	6.6	36	0.736	2.607	2.123	Knuth (1993)
7	Copper	112	425	7.59	49	0.19	2.513	1.815	Newman (2006)
8	Football	115	613	10.66	12	0.403	2.486	1.185	Girvan and Newman (2002)
9	Jazz	198	2.7k	27.7	100	0.633	2.224	2.142	Gleiser and Danon (2003)
10	Dating	288	284	1.98	9	0	16.075	1.508	Bearman et al. (2004)
11	SlashDot	82k	504k	12.27	2552	0.0603		3.147	snap (2018)
12	Stanford	281k	1M	8.2	255	0.5976		4.489	snap (2018)

of the sets are non-directional, unweighted networks. The sets tina, ragusa, karate teen, jazz, lesMis, dating and slash-Dot are social networks extracted from books, music albums, studies or historical documents. The Parker Brothers game, Risk was transposed using the countries linked by borders. football is the network of college teams linked by matches and stanford is the web graph from Stanford University.

The table has columns for average clustering coefficient, average path length and power law coefficient. These allow the reader to identify networks as small world (high clustering coefficient and low average path length), scale free (power law coefficient between 2 and 3) or other. For example, football network is small world but not scale free, wikiElec is scale free but not small world and lesmis is both small world and scale free. In addition to the list of the data sets, there are diagrams of the smaller ones in Fig. 2.

In some of the experiments it is necessary to evaluate the quality of the commSets. There are some established statistics designed to measure the quality but it is not the intention of this paper to compare the results using every available statistic. While NEO's merits can be debated, it will be used here as a measure of quality. When comparing algorithms, modularity (Newman and Girvan 2004), a well known statistic, will also be used to provide an additional level of confidence for readers.

In the experiments, besides Gamit and CHI, five other community finding algorithms are used to support the claims above. They were chosen to represent a range of different approaches to community finding. The first algorithm is the agglomerative (*agglom*) by (Clauset et al. 2004). The algorithm begins with singleton communities and joins them to maximize modularity. The algorithm can be stopped at any threshold in the joining process to produce the commSet. The threshold chosen was the one that maximized modularity. While there have been improvements made to this algorithm they have been mainly to improve the complexity leaving the basic approach intact. This is the only algorithm that specifically creates disjoint communities.

The second algorithm, CFinder by (Palla et al. 2005), uses the clique percolation method. In this approach, k-cliques are found and joined if they share $k-1$ nodes. Overlap is possible when a node is in more than one k-clique. In most experiments the best results were used, with $k = 3$, 4 or 5. There were some networks where CFinder did not find any communities.

Another approach (Ahn), by Ahn et al. (2010), partitions links hierarchically using edge similarity. Since a node can have many links, that node belongs to every community that each of its links are assigned to. Link similarity is based on the Jaccard index using the neighborhoods on the adjacent nodes. Links that are part of a tight community would have a high Jaccard index.

The agent-based algorithm SLPA by (Xie et al. 2011) is an extension of the label propagation method. It spreads labels between nodes according to pairwise rules. The nodes can retain a memory of past transactions which allows it to place a node in more than one community.

The last algorithm, OSLOM by Lancichinetti et al. (2011) uses a local expansion and optimization approach. It grows communities by adding neighboring nodes whose probability of having internal connections greater than a random model. If a node has significant connections to two growing communities it can be placed in both.

5.2 Variability of commSet characteristics

For a given network there are many possible commSets. It is not enough to show that their characteristics vary over the entire range of commSets. The quality of the commSets can also vary and as a general rule, higher quality commSets are desired. Recall that high quality commSets are those that have many links within the communities and fewer ones between. It is defined here as low values of the function in Eq. 1 (NEO). NEO was chosen because it satisfies the general concept of quality and it does not limit high quality commSets to a limited region of the canvas. The experiments will show that:

- there is a large number of possible commSets for even small networks, but only a very small number of high quality ones
- the high quality commSets are not limited to a concentrated area of the commSetSpace
- existing algorithms tend to localize their solutions
- different data sets tend to have good commSets with varied characteristics

The first step is to show the distribution of quality over the range of commSets. Finding communities in a network is a difficult task because the search space is so large. For example, it can be calculated that the number of different commSets for karate (Zachary 1977) using $k = 2$, is 2.9×10^{20}. Attempting to do an exhaustive search is prohibitive for even small networks like karate. However, to show the distribution of commSet quality it will be necessary to do exhaustive search. A Monte Carlo approach would be inappropriate because, as it will be shown, there are very few high quality commSets and even if a large number of samples were chosen it is likely that it would select only mediocre commSets.

5.2.1 The small number of quality commSets

The experiments here are designed to show the need for Gamit and CHI so those algorithms are not used. The

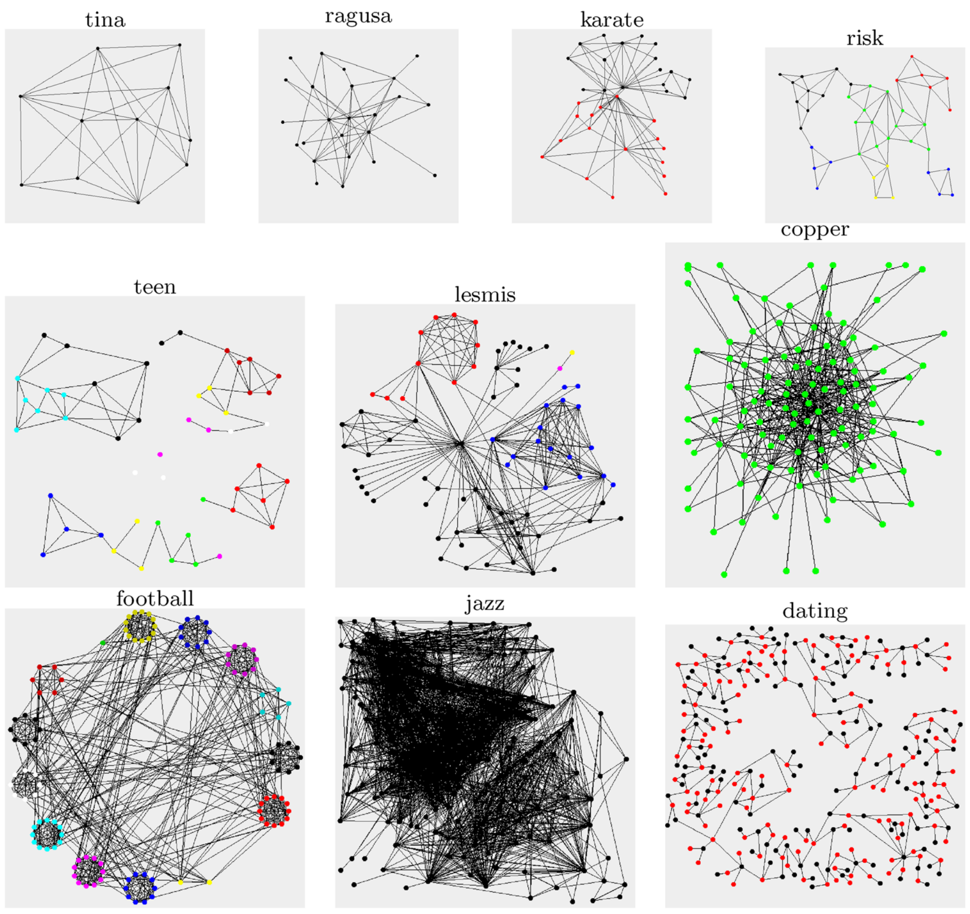


Fig. 4 Images of the small networks used in the experiments

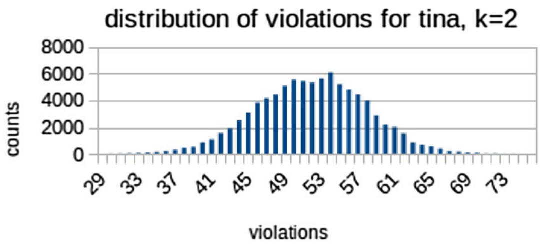


Fig. 5 Histogram of commSets for tina

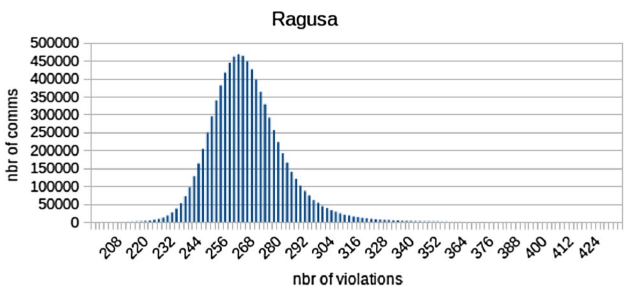


Fig. 6 Histogram of commSets for ragusa

experiments also operate on tiny (7–9 nodes) and small networks. These networks are used for three reasons. First, because some of the experiments are exhaustive and can only be done on tiny graphs. Second, because some of the algorithms used for comparison had restrictive memory or time constraints and which limited the size of the networks. Third, the small networks can be visually displayed which will be helpful in understanding the results of some of the experiments.

To begin, two small networks will be used to visually show the distribution of commSet solutions. The set tina Hlebec (1993), Pajek datasets (2018) is an 11 node graph based on a study of 11 members of student government. The total number of possible combination of nodes grouped into 2 communities is 88,572. The ragusa Pajek datasets (2018) set is a 24 node graph based on the ruling families of ragusa (now Dubrovnik). There are 8,388,607 possible combinations of 2 communities for the ragusa set Fig. 4.

Using the technique described in Kurmas et al. (2014), an exhaustive search of all possible 2-commSets were evaluated on the two graphs. For each commSet the number of NEO violations were computed. The results have

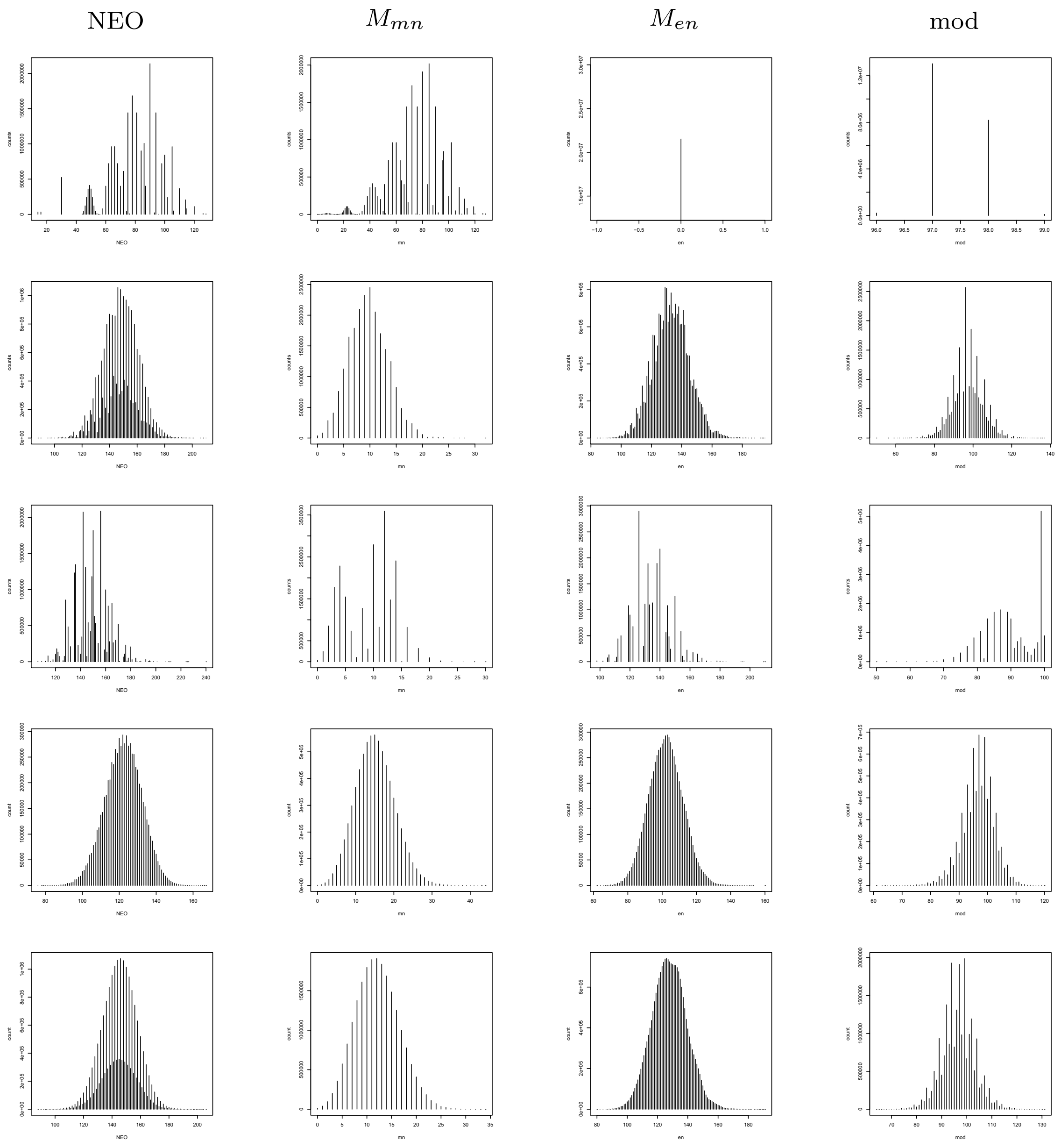


Fig. 7 Distribution of violations for the clique (row 1), cycle (row 2), star (row 3), kapfer (row 4) and padgett (row 5) networks

been plotted as histograms in Figs. 5 and 6. The total NEO violations are measured on the horizontal axis with the counts on the vertical axis. In both plots, NEO appears to follow a well behaved distribution.

It should be noted that NEO is the sum of 3 statistics and each of the statistics may have its own distribution. While the sum might appear to be normal it is probably more complex than that. Figure 7 shows histograms of the NEO statistics for five different networks. The networks include tina and ragusa plus three synthetic networks that represent extremes. The first row is a 16 node clique, the second row is for a 16 node ring network (each node is connected to the 2 nearest ones) and row three is a 16 node star. The fourth and fifth rows show the charts for tina and ragusa.

For each network, there is a chart for the total NEO, another for M_{mn}, another for M_{en} and finally one for modularity. The chart for M_{ol} is not shown as it does not depend on the network and always looks like a binomial distribution. Even though the analysis uses NEO, modularity charts were added to show that it too, appears to behave somewhat like a distribution. Notice that the distributions look jagged for the clique and star. M_{en} for clique is a single bar—it will always be zero since every node is attached

Table 3 Percent of graphs that have less than x% of commSets in the top 10% of commSet distribution

Nodes	$x = 1\%$	$x = 0.5\%$
7	45.4	18.6
8	81.6	54.3
9	91.3	73.8

to every other. The total NEO in these extreme networks is not a normal distribution but there clearly are many more mediocre commSets and very few high quality ones.

As Fig. 7 demonstrates, the commSet distribution for graphs will not always follow a well defined distributions. The next experiment will provide evidence that many graphs will have very few commSets on the low NEO end of the distribution. The experiment was run to calculate the commSet distribution for every possible graph of n nodes. This sort of analysis is not possible for even small values of n such as 34 (size of karate set). The number of graphs grows exponentially with n as does the number of possible commSets. Using $n = 7, 8$ and 9 was feasible for the equipment available while still yielding some interesting results.

The experiment used the graphing program (McKay and Piperno 2013) to generate the graphs. For each graph, the NEO score for all possible commSets were calculated. The experiment determines whether a graph has less than x% of the commSets within the lowest 10% of the range of NEO scores. For example (using $x = 0.01$, supposing that the range of NEO scores for a commSet went from 101 to 200, this experiment determines whether less than 1% of the commSets have NEO scores in the range of 101–110. Results are tallied for all of the graphs ($n = 7, 8$ and 9, to display what percent of them have less x% of the commSets within the lowest 10% of scores. The experiment was done for only $k = 2$ but it is assumed that other values of k would lead to similar results.

Table 3 summarizes the results. For $n = 7$, only about 45% of the graphs have less than 1% of the commSets in the first 10% and only about 19% have less than 0.5%. As n increases so do the percentages. With larger n there should be many more commSets and the distributions should be better defined. When searching for near optimum solutions then, we can expect there to be many, many good and fair solutions, but only a tiny few near-optimum solutions.

5.2.2 Variety in quality commSets

The intention now is to show that these few near-optimum solutions can have different characteristics. The commSet-Space canvas provides a convenient way to describe the characteristics of commSets. After finding a community,

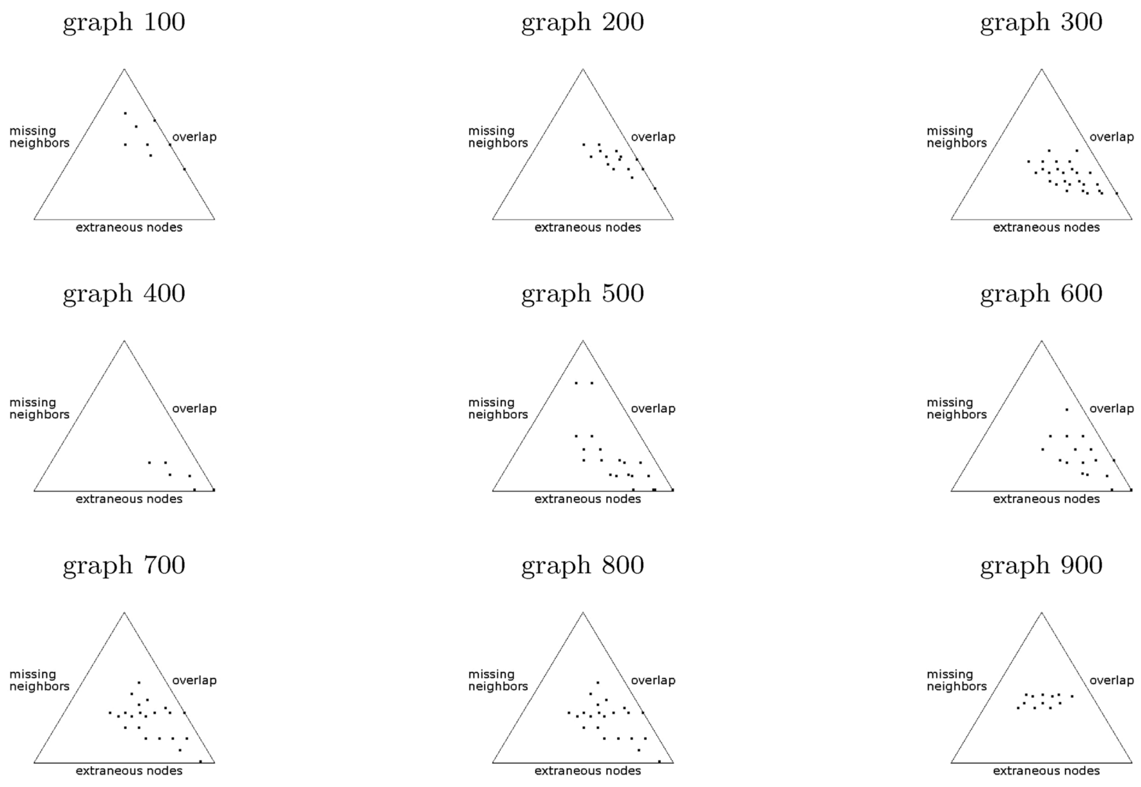


Fig. 8 Top commSets for a selection of graphs

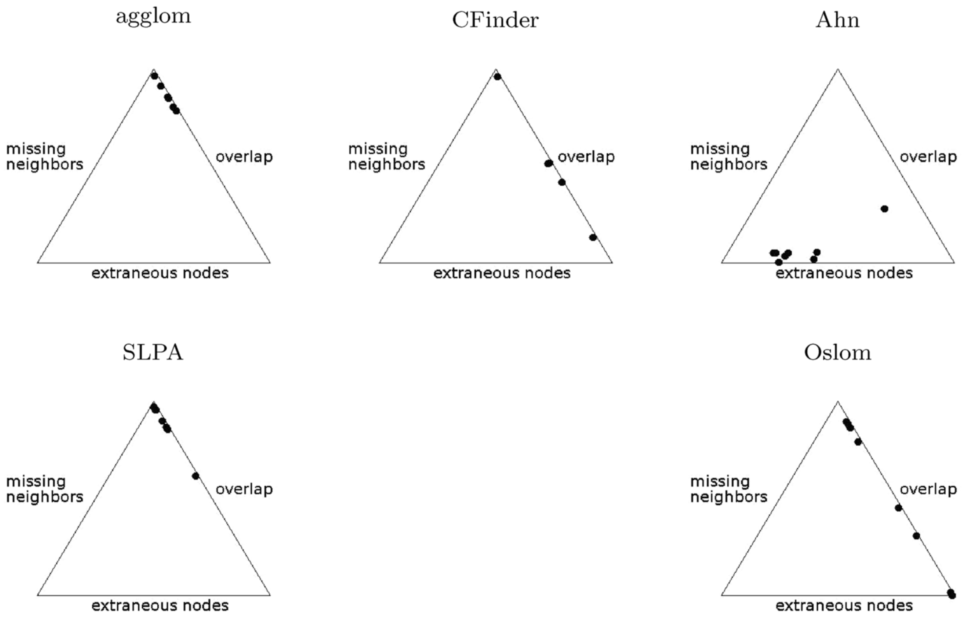


Fig. 9 NEO placement for different algorithms

one can calculate the NEO metrics and place a dot on the canvas to convey the characteristics of the set. Then at a glance, one can tell if it is disjoint, ego-centric, clique-like or somewhere in between.

In the first experiment, all of the best (lowest NEO score) commSets for a given 7 node graph are plotted on the canvas to show the diversity of the sets. The NEO scores ranged from 9 ± 3 to 38 ± 2. The sets with a NEO score of less than the minimum plus 3 were chosen. Of the 135,072 possible commSets, this represented less than 0.3% in all cases. Rather than show the results for all 1044 graphs, a pseudo random selection of graphs were chosen.

In Fig. 8, the commSets have been plotted according to their NEO metrics for the graphs 100, 200, etc. The graphs are ordered by the sequence that they are extracted from Nauty. While for some graphs the best solutions are located in a somewhat narrow region of the chart, in all cases they are spread out within that region.

It is not claimed here that these results conclusively prove that for all networks good commSets can be found with differing characteristics. However, doing the experiment on a large graph is not feasible due to the exponentially large number of possible sets. Even with small networks like Tina and Ragusa it is prohibitive.

5.2.3 Algorithms are localized and data sets are not

There are many different community finding algorithms that optimize different criteria. It can be claimed that they would tend to find the same kind of community. To test this claim, five different algorithms were tested on a number of different network data sets and the results plotted on the canvas. Results of this experiment are shown in Fig. 9. The agglom and SLPA algorithms appear to consistently find commSets at the top right of the triangle where there is zero or little overlap and low values of M_{mn}. Ahn is also fairly consistent with results in the lower left portion of the triangle where the communities are like cliques or neighborhoods. CFinder and Oslom are less consistent, finding communities along the right edge. These commSets have low overlap and range from a very small number of communities with low M_{mn} down to singletons.

These results show that a particular algorithm will find commSets with specific characteristics or a range of characteristics but that the decision is part of the algorithm and not adjustable by the analyst.

The results of the previous experiments are organized by data set in Fig. 10. A canvas is shown for each data set with dots being plotted for the commSets that each of the different algorithms found. Notice that the plots for karate, risk and teen have many commSets near the top of the canvas. In Fig. 4, one can see that these networks appear to have a small number of non-overlapping communities. Thus it is not surprising that the commSets would be found near the top where M_{mn} and M_{ol} are low. lesMis, copper, football and jazz are more dense and thus one would expect that the high quality commSets would have more communities of smaller number of nodes and possibly more overlap. Accordingly the solutions for these networks are more spread out towards the corner with singletons. The dating network is very sparse and very localized clustering (only between pairs of nodes). For this network, the algorithms either found a few (15–25) large communities or many (more than 280) communities of singletons or two-node cliques.

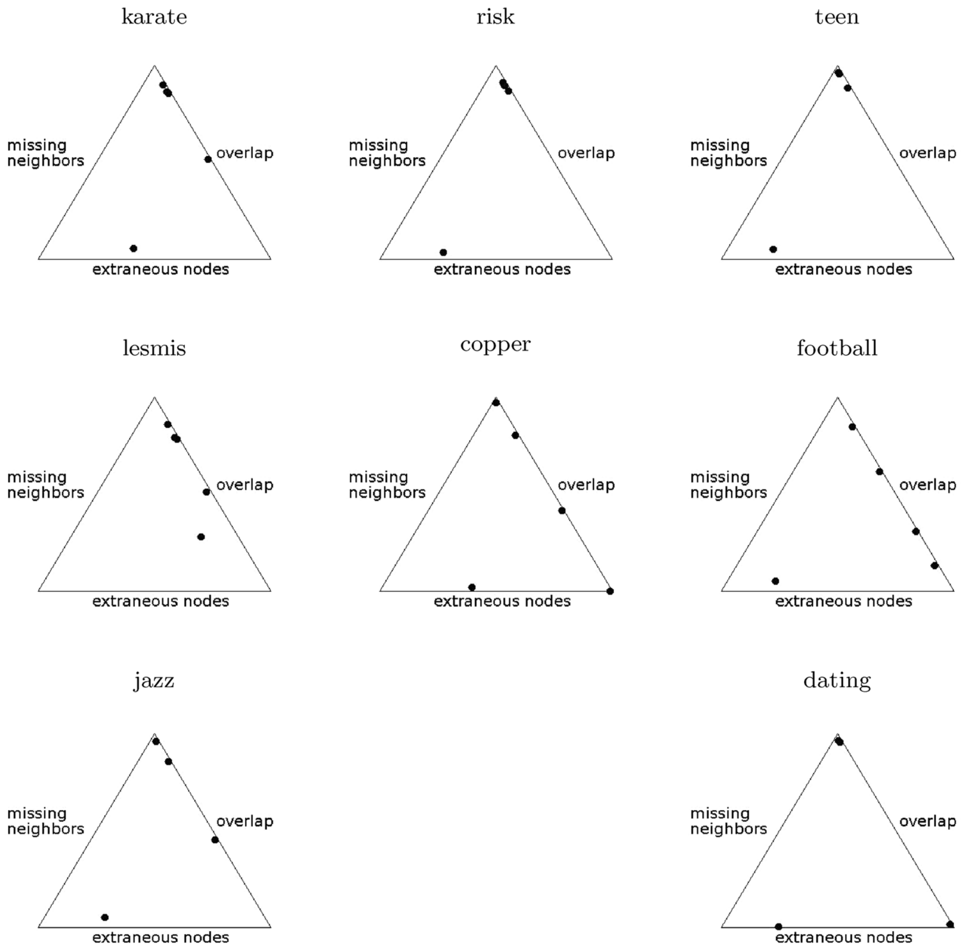


Fig. 10 NEO placement for different data sets

5.3 Effectiveness of Gamit and CHI

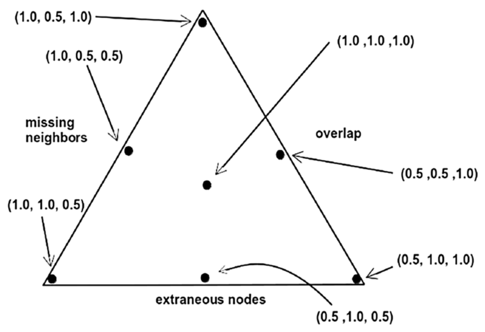
The second group of experiments will show the effectiveness of Gamit and CHI. In particular, it will be shown that:

- the λ parameter allows the user to find germane commSets—those with specific characteristics
- Gamit and CHI can be tuned to find solutions that are similar to solutions found by the other algorithms in both characteristics and quality
- CHI is efficient

5.3.1 Finding germane communities

By setting the λ parameters the analyst can encourage both Gamit and CHI to focus on commSets with particular characteristics. The experiments will show that while the algorithms often do find commSets with the desired characteristics there are some characteristics that are ellusive. With most sets, the algorithms have trouble with the area of the triangle in the middle, stretching to the edge with missing neighbors.

Recall that Gamit is an agglomerative method which merges communities based on optimizing Function 5. CHI, is an iterative method like KMeans, which starts with a



description	λ_1	λ_2	λ_3
top	1.0	0.5	1.0
left middle	1.0	0.5	0.5
left bottom	1.0	1.0	0.5
bottom middle	0.5	1.0	0.5
right bottom	0.5	1.0	1.0
right middle	0.5	0.5	1.0
center	1.0	1.0	1.0

Fig. 11 λ Values for experiment to find specific commSets

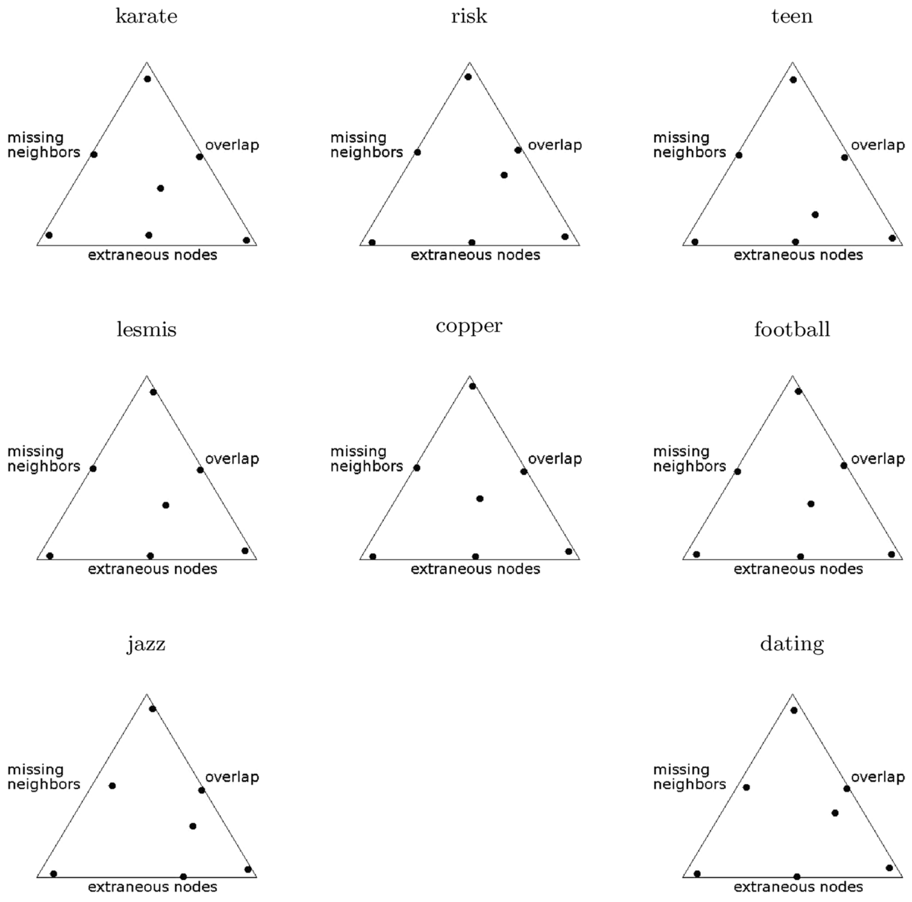


Fig. 12 Gamit placements for different networks using different λ values

randomly chosen commSet. CHI typically finds commSets with very low NEO values but can drift from the characteristics specified by the λ values. Gamit finds commSets that are very close to the characteristics desired but often has higher NEO values. Analysts could effectively use a combination of the two algorithms; use Gamit to find a set with the desired characteristics and then apply CHI to improve the results, lowering the NEO values.

For the experiments in Fig. 12 we used Gamit only so that the results would more accurately reflect the desired characteristics as specified the the λ values. For this experiment we ran Gamit on eight different data sets using 7 different λ values. The λ values chosen—shown in the table of Fig. 11—represent the extreme positions of the triangle. It should suffice to show that if the algorithm can consistently find commSets with the characteristics of these extreme points, it can find the sets with any desired characteristics.

The results of the experiments can be seen in Fig. 12. As can be seen in all 8 data sets Gamit nearly always finds commSets with the desired characteristics. The major difficulty happens with $\lambda = (1, 1, 1)$—the set in the center of the triangle. The result from Gamit had characteristics quite different from the λ parameters for the sets risk, teen, jazz and dating. During the process of the algorithm, when communities are merged, early decisions could effect the results later. We suspect that the data sets that stray from the center position have some structural qualities that lead to these situations. Of course, if one is unhappy with the results of the algorithm, the λ values can be modified and new communities generated. With a few exceptions, our experiments show that Gamit is effective in finding communities with specific characteristics.

5.3.2 Comparison to other algorithms on real datasets

We ran a number of experiments on the data sets to compare Gamit and CHI[1] to the other algorithms selected for this paper. To compare the algorithms we used two metrics, NEO and modularity. One would expect CHI to do well at reducing NEO since it is specifically designed to do just that. We include modularity—a popular metric—as an additional comparison.

To do a fair comparison between gCHI and the other algorithms, it is important to insure that gCHI is finding

[1] For the remainder of this section, we will refer to the combination of Gamit and CHI just as gCHI.

Table 4 Comparison of Gamit and CHI (gCHI) to other algorithms

Data set	NEO			Mod		
	Alg	gCHI		Alg	gCHI	
Agglom						
Karate	320	295	•	0.381	0.200	
Risk	258	200	•	0.633	0.620	
Teen	302	150	•	0.736	0.714	
LesMis	1578	576	•	0.135	0.371	•
Copper	2466	1648	•	0.292	0.146	
Football	2064	1303	•	0.566	0.567	•
Jazz	9780	9713	•	0.439	0.229	
Dating	4748	660	•	0.878	0.736	
CFinder						
Karate	232	216	•	0.063	0.334	•
LesMis	908	490	•	0.068	0.389	•
Copper	1212	1040	•	0.009	0.164	•
Football	616	1004		0.489	0.576	•
Jazz	24594	7117	•	0.016	0.230	•
Ahn						
Karate	278	151	•	0.010	0.036	•
Risk	271	88	•	0.057	0.499	•
Teen	244	65	•	0.070	0.582	•
LesMis	735	255	•	0.039	0.452	•
Copper	1718	851	•	0.009	0.017	•
Football	2135	1216	•	0.008	0.035	•
Jazz	9484	5171	•	0.002	0.011	•
Dating	1128	581	•	0.114	0.184	•
SLPA						
Karate	468	295	•	0.244	0.200	
Risk	226	202	•	0.627	0.609	
Teen	278	146	•	0.741	0.702	
LesMis	2000	576	•	0.163	0.371	•
Copper	11582	1637	•	0.000	0.147	•
Football	966	1303		0.592	0.567	
Jazz	9398	9713		0.441	0.229	
Dating	3188	660	•	0.856	0.736	
Oslom						
Karate	447	117	•	0.176	0.302	•
Risk	542	94	•	0.395	0.600	•
Teen	263	62	•	0.612	0.635	•
LesMis	1796	302	•	0.083	0.467	•
Copper	850	1088		– 0.016	0.227	•
Football	598	738		0.580	0.603	•
Jazz	4257	4113	•	0.364	0.331	
Dating	584	452	•	0.015	0.648	•

the commSets with similar characteristics of the set found by the other algorithm being compared. This was done by running the other algorithms first and then, using the NEO values to assign λ's that would guide CHI to find a set with similar NEO values.

The results are summarized in Table 4. The rows are grouped by the algorithms agglom, CFinder, Ahn, SLPA and Oslom. Within each algorithm all of the data sets are listed. Since CFinder does not always find a set of communities, only the data sets that had found communities are listed. The second two columns show the NEO values for both the

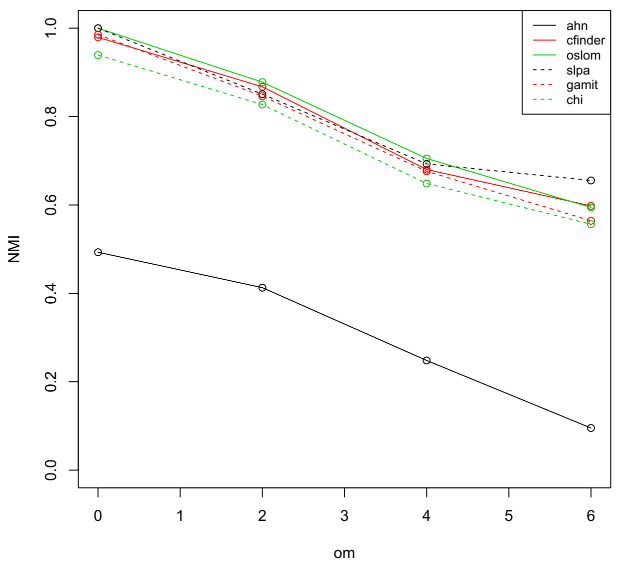


Fig. 13 Comparison of algorithms using NMI to ground truth communities of benchmark networks

algorithm listed to left and gCHI. Columns 5 and 6 show the modularity for the two algorithms. Column 4 contains a bullet if gCHI has lower NEO score than the other algorithm and column 7 contains a bullet if CHI has a higher modularity. Recall that NEO is a measure of violations so lower values are better but for modularity higher values signify a better commSet.

Considering NEO, it is not surprising that gCHI does better than all of the other algorithms on all data sets except for 5 exceptions. In many cases the differences are dramatic. Looking at modularity the results are more mixed. gCHI has better results on all data sets for both CFinder and Ahn and all but one set for Oslom. SLPA and agglom have higher modularity scores than gCHI in 6 out of the 8 data sets. Respecting agglom, this is not surprising since it specifically optimizes modularity. While SLPA does not specifically optimize modularity it's message passing algorithm puts the emphasis on keeping linked nodes together which would result in higher scores for modularity. Even though these algorithms had higher modularity than gCHI in many circumstances the differences are not very large.

5.3.3 Comparison to other algorithms on benchmark, (ground truth) networks

Real networks often have a natural community structure that is compatible with the link structure. Consider faculty at a university; academic department communities form naturally because individuals are more likely to be linked to others in their department than outside of it. Since there are not many real data sets available with these ground truth

Table 5 Comparison of CHI to other algorithms on benchmark networks (NMI and seconds)

Alg.	om = 0	om = 2	om = 4	om = 6	Time
Ahn	0.9793	0.8674	0.6802	0.5979	0.4
cFinder	0.4931	0.4129	0.2481	0.0953	0.7
Oslom	1	0.8779	0.7052	0.5944	1.8
Spla	1	0.8507	0.6929	0.6555	2.7
Gamit	0.9853	0.8463	0.6762	0.5643	12.8
Chi	0.9393	0.8273	0.6486	0.5565	0.02

communities we used the LFR benchmark (Lancichinetti et al. 2008). This allows networks to be generated with different characteristics. For our experiments we generated networks of 1000 nodes, with average and maximum degrees of 10 and 50 respectively, minimum and maximum community sizes of 20 and 50. We set the number of nodes to have overlap at $O_n = 100$ with the overlap (O_m) set to 0, 2, 4, and 6.

For the experiments, the networks were generated and then for each algorithm, communities were detected and then compared to the ground truth communities using the extended normalized mutual information (NMI) proposed in Lancichinetti and Fortunato (2009). This metric compares the similarity of two sets and ranges between 0 and 1 with 1 being a perfect match. For Gamit and CHI lambda values were set to $\lambda = (1, 1, 1)$ and $\lambda = (1, 0.1, 1)$ with the best results reported.

The result of the experiments can be seen in both Table 5 and Fig. 13. SLPA and Oslom were the best at recovering the ground truth communities with CFinder, Gamit and CHI having slightly lower values of NMI. It should be noted that as the amount overlap increases all algorithms do worse at detecting the communities. This is probably because the additional overlap obscures the ground truth community structure. In the face of this, algorithms that use the link structure to find communties will have a greater variance in the communities that are found. That being the case, Gamit and CHI will find communities with the characteristics desired by the analyst according to the lambda values submitted.

5.3.4 Scalability

In Sect. 4 it was shown that the complexity for CHI is $O(Ikna)$ and for Gamit is $O(n^2 \log n)$. In practice the algorithm typically converges in 3–10 loops so we can consider I to be a constant. Also, a—the average number of neighbors—is often fairly small in most sparse networks. This means that CHI is really bounded by kn. Obviously CHI scales much better than Gamit. There may be a more

Table 6 Comparison of CHI to agglomerative for large sets (metrics and time)

	NEO		Mod		Seconds	
Data set	Alg	CHI	Alg	CHI	Alg	CHI
SlashDot	1441M	4M	0.324	0.165	900	157
Stanford	3911M	1457M	0.894	0.823	1947	1434

efficient way to implement Gamit but at this point it is left as future work.

To demonstrate the scalability the running times (in seconds) were recorded on the benchmark tests described above. The numbers can be seen in Table 5 in the last column. While Gamit is typically more accurate than CHI in finding communities that reflect the given lambda values, it is also much less efficient. It is several times slower than the other algorithms. On the other hand CHI by itself is much more efficient than the other algorithms.

For another demonstration of the scalibility of CHI, tests were run on two larger networks: slashDot and stanford. Due to their size, we chose to run just CHI without Gamit, which still produced good results. It suffices to compare to the results to agglom since problems were encountered using the other algorithms. The results are listed in Table 6. For both sets, we set λ to values that would find commSets like agglom. Not surprisingly, CHI had better NEO results than agglom. For modularity it was lower than agglom but not terribly far below, especially for stanford. CHI ran faster than agglom for both sets.

6 Conclusions

This paper used the commSetSpace canvas to reason through the characteristics that different commSets might take on. For example, communities belonging to a set with a small number of communities and zero overlap will have different characteristics than, say, a set with many overlapping communities. Each may have advantages for different analyses.

It was shown that while there are a very large number of possible commSets for any given network, there are relatively few that have low values of the NEO statistic. In the cases we studied, the few sets with low NEO could have different characteristics. It is important then, that analysts are able to find good sets (those with good statistics) as well as sets with the desired characteristics.

Through experiments, it was shown that specific algorithms tend to find sets with specific characteristics (or a specific range of characteristics). It is important, when using an algorithm to know the type of communities it produces. Running the algorithms on data sets singly showed that a specific data set might have good commSets with a specific range of characteristics. So while an analyst may wish to find a commSet with specific characteristic requirements, the data set may not have good sets with those requirements.

Two algorithms are presented to find communties with different characteristics. CHI is a fast, EM-like algorithm that finds a local minimum for a seed set of communities. It step-wise improves the solution to optimize the objection function in Eq. (2). If a seed is not provided it will start with a random commSet. The other algorithm, Gamit, is an agglomerative algorithm that merges communities based again on the Eq. (2). With both algorithms the analyst can tune the solution using the λ parameters.

The central theme of this paper is the importance of finding commSets of high quality and the right characteristics. The algorithms can be used in series, using Gamit to find a commSet that is close to the desired characteristics and then using CHI to improve the quality of the results. Since CHI scales much better than Gamit, with large networks, using CHI by itself will be much faster. The experiments showed that the algorithms are effective when compared to other proposed algorithms.

6.1 Final thoughts

Although the efficiency of CHI was demonstrated in the experiments it is also possible to speed up the algorithm through parallelization. From one iteration to the next, changing the community or home community assignments for one node do not impact those of another, so the process can be done in parallel.

For those wishing to test or make use of the algorithms, the Java version of CHI has been posted to http://www.cis.gvsu.edu/~scrippsj/pubs/software.htm. Also, a tool for analyzing small networks, Netzer, is also posted on the same page. Netzer is a GUI tool, written specifically for small networks only because of the visualization. The community finding portion of Netzer uses both CHI and Gamit.

References

Ahn Y-Y, Bagrow JP, Lehmann S (2010) Link communities reveal multiscale complexity in networks. Nature 466:761–764

Bearman P, Moody J, Stovel K (2004) Chains of affection: the structure of adolescent romantic and sexual networks. Am J Sociol 110:44–91

Clauset A, Moore C, Newman MEJ (2006) Structural inference of hierarchies in networks. In: Airoldi E, Blei DM, Fienberg SE, Goldenberg A, Xing EP, Zheng AX (eds) Statistical network analysis: models, issues, and new directions, vol 4503. Springer, Berlin, Heidelberg

Clauset A, Newman MEJ, Moore C (2004) Finding community structure in very large networks. Phys Rev E 70:066111

Girvan M, Newman M (2002) Community structure in social and biological networks. Proc Natl Acad Sci 99:7821–7826

Gleiser P, Danon L (2003) Community structure in jazz. Adv. Complex Syst 6:565. http://deim.urv.cat/~aarenas/data/welcome.htm

Hlebec V (1993) Recall versus recognition: comparison of the two alternative procedures for collecting social network data. Dev Stat Methodol 9:121–129

Jain A, Dubes R (1988) Algorithms for clustering data. Prentice-Hall Inc, Upper Saddle River

Knuth DE (1993) The Stanford GraphBase: a platform for combinatorial computing. Addison-Wesley, Reading

Kurmas Z, McGuire H, Scripps J, Trefftz C (2014) Enumerating communities for a deeper understanding of community finding. In: Proceedings of the 2014 IEEE/WIC/ACM international conference on web intelligence

Lancichinetti A, Fortunato S (2009) Community detection algorithms: a comparative analysis. Phys Rev E 80:056117

Lancichinetti A, Fortunato S, Radicchi F (2008) Benchmark graphs for testing community detection algorithms. Phys Rev E 78:046110

Lancichinetti A, Radicchi F, Ramasco JJ, Fortunato S (2011) Finding statistically significant communities in networks. PLoS One 6(4):e18961

McKay BD, Piperno A (2013) Practical graph isomorphism. Symb Comput 60:94–112

Newman MEJ (2006) Finding community structure in networks using the eigenvectors of matrices. Phys Rev E 74(3):036104

Newman M, Girvan M (2004) Finding and evaluating community structure in networks. Phys Rev E 69:026113

Pajek datasets (2006) http://vlado.fmf.uni-lj.si/pub/networks/pajek/default.htm

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

Porter M, Onnela J, Mucha P (2009) Communities in networks. Not Am Math Soc 56:1082–1097

Scripps J (2011) Exploring the community set space. In: IEEE/WIC/ACM international conference on web intelligence, pp 316–319

Scripps J, Trefftz C (2013) Community finding within the community set space. In: ACM workshop on social network mining and analysis (SNAKDD13)

Shi J, Malik J (2000) Normalized cuts and image segmentation. IEEE Trans Pattern Anal Mach Intell 22(8):888–905

SNAP (2014) Stanford large network dataset collection. http://snap.stanford.edu/data/

Tan P, Steinbach M, Kumar V (2005) Introduction to data mining. Addison Wesley Inc., Boston

Tang L, Wang X, Liu H, Wang L (2010) A multi-resolution approach to learning with overlapping communities. In: KDD workshop on social media analytics, pp 14–22

West P, Sweeting H (1995) Background rationale and design of the west of Scotland 11–16 study. Working paper No. 52. MRC Medical Sociology Unit Glasgow

Traud A, Kelsic E, Mucha P, Porter M (2011) Comparing community structure to characteristics in online collegiate social networks. SIAM Rev 53:526–543

Xie J, Kelly S, Szymanski B (2011) Overlapping community detection in networks: the state of the art and comparative study. CoRR. http://arxiv.org/abs/abs/1110.5813

Xie J, Szymanski BK, Liu X (2011) Slpa: uncovering overlapping communities in social networks via a speaker–listener interaction dynamic process. In: Data mining technologies for computational collective intelligence workshop at ICDM, pp 344–349

Zachary WW (1977) An information flow model for conflict and fission in small groups. J Anthropol Res 33:452–473