FINDING COMMUNITIES BY CLUSTERING A GRAPH INTO OVERLAPPING SUBGRAPHS[*]

Jeffrey Baumes
Rensselaer Polytechnic Institute, Computer Science Department
110 8th St., Troy, NY 12180, USA

Mark Goldberg
Rensselaer Polytechnic Institute, Computer Science Department
110 8th St., Troy, NY 12180, USA

Mukkai Krishnamoorthy
Rensselaer Polytechnic Institute, Computer Science Department
110 8th St., Troy, NY 12180, USA

Malik Magdon-Ismail
Rensselaer Polytechnic Institute, Computer Science Department
110 8th St., Troy, NY 12180, USA

Nathan Preston
Rensselaer Polytechnic Institute, Computer Science Department
110 8th St., Troy, NY 12180, USA

ABSTRACT

We present a new approach to the problem of finding communities: a community is a subset of actors who induce a *locally optimal* subgraph with respect to a *density function* defined on subsets of actors. Two different subsets with significant overlap can both be locally optimal, and in this way we may obtain overlapping communities. We design, implement, and test two novel efficient algorithms, RaRe and IS, which find communities according to our definition. These algorithms are shown to work effectively on both synthetic and real-world graphs, and also are shown to outperform a well-known *k*-neighborhood heuristic.

KEYWORDS

algorithms, communities, clustering, testing

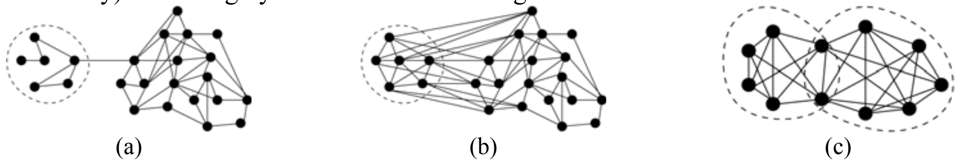
1. INTRODUCTION

Actors (nodes) in a large social communication network, such as the WWW, form *overlapping* communities. Our goal is to present algorithms for discovering overlapping communities, based on the communication pattern (graph), not using the communication semantics. The characteristic property of a community, as opposed to an arbitrary collection of actors, is that members of a community communicate "more densely" with each other than with actors not in the community. Thus, finding communities is related to finding

[*] This research was partially supported by NSF grants 0324947 and 0346341

"dense" clusters in a graph, which is the guiding principle for most classical clustering algorithms such as: distance-based algorithms [11]; flow-based algorithms [8]; partitioning algorithms [3, 13]; and matrix-based algorithms, such as SVD [5]. These existing approaches allow an actor to be in at most one cluster, which is a severe limitation since communities in social networks can have significant overlap.

Our Contribution. We present a new principle for finding overlapping communities: a community is a subset of actors whose induced subgraph *locally optimizes* some graph property, where we define the locality of a subset as those subsets that are sufficiently "close" to it. We call the graph property optimized a *metric*, or *density*. Two different subsets with significant overlap can both be locally optimal, and in this way we may obtain overlapping communities. We contrast the notion of locally dense subsets (for an intuitive "visual" definition of density) versus highly dense subsets in the figures below.


(a) (b) (c)

In (a) the highlighted subset is locally dense, and thus should be considered a community, even though there are other subsets with much higher density. In (b), the same subset, though it has higher density than in (a) is not locally dense. In (c) we show how two subsets can both be locally dense and have overlap. We thus re-formulate the problem of finding communities to the problem of finding all locally optimal graph clusters (we use the term clustering, even though in the classical literature it often refers to partitioning into disjoint subsets e.g. [14]).

We construct two novel, efficient heuristics for clustering: IS, which executes a sequence of iterative scans; and RaRe, which is based on the idea of decomposing the graph by removing "highly ranked" nodes (one notion of rank could be the well known page-rank, [15]). The theoretical analysis of these algorithms is challenging, and so we give a rigorous experimental analysis of the algorithms using both random graph models (including $G(n,p)$ random graphs [7]), and real communication graphs (E-mail, Web, News Groups). These algorithms are shown to perform uniformly better than a simple algorithm for constructing overlapping clusters using the k-neighborhood of nodes. Our algorithms are general in that they do not commit to particular density measures, and allow these to be input. In our simulations, we used three particularly intuitive measures of density (Section 2), all of which measure the degree of connectedness within a cluster, as compared to that between the cluster and its complement. Even though the measures are similar, they can lead to significantly different communities. Additionally, our algorithms accept size specification parameters which allow one to search for communities in a specified size range. Such parameters allow us to find communities that would otherwise be missed because they are not significantly dense when compared with the density of larger portions of the network. The experiments show that both for random as well as real graphs, our heuristics give reasonable communities (based on visual inspection).

Related Work. Data clustering, graph clustering and partitioning encompasses a large body of work in several disciplines, [11, 12, 16]. There even exist public domain software, such as METIS [10] and Birch [19]. All such classical approaches output disjoint clusters, and can be classified based on their methodology (hierarchical, non-hierarchical, supervised, and unsupervised), and by the type of data (numeric, spatial, temporal (time-series), sequential, categorical).

Fuzzy sets, [4, 6, 9, 17, 18], are predominantly used in pattern recognition to represent vagueness in data. The main idea is that a point has some probability to belong to a set (fuzzy set membership) – ultimately it belongs to one set, except that to which is uncertain. There is a certain similarity to our problem: on account of the "fuzziness", an object can be viewed as belonging to more than one cluster. However, in our setting, an object deterministically belongs to different clusters, not due to fuzziness.

Paper Organization. In Section 2 we discuss preliminary definitions. In Section 3, we present our algorithms, and Section 4 gives the experimental analysis.

2. PRELIMINARIES

Let $G=(V,E)$ be a graph whose nodes represent individuals, web pages, etc. and whose edges represent communications, links, etc. The graph may be directed or undirected. We present the definitions for directed graphs, the undirected case being similar. A graph cluster *C* is a set of vertices which can be viewed as a binary vector of length |*V*| that indicates which nodes are members of *C*. The set of all graph clusters, K , is the power set of *V*.

A weight function or metric is a function $W : \mathrm{K} \mapsto \Re$ that assigns a weight to a graph cluster. Associated to cluster *C*, we define three edge sets: $E(C)$, the edges induced by *C*; $E(C,\overline{C})$, the edges in *E* from *C* to its complement; $E(\overline{C},C)$, the edges in *E* to *C* from its complement. Let $E_{out}(C) = E(C,\overline{C}) + E(\overline{C},C)$. We define the *internal and external edge intensities*,

$$p_{in}(C) = \frac{E(C)}{|C| \cdot (|C|-1)}, \qquad p_{ex}(C) = \frac{E_{out}(C)}{2|C| \cdot (n-|C|)},$$

($p_{ex} = 1$ when $|C| = |V|$). We will consider three weight functions: the *internal edge-probability* W_p ; the *edge ratio* W_e ; and, the intensity ratio W_i ,

$$W_p(C) = p_{in}(C), \qquad W_e(C) = \frac{E(C)}{E(C) + E_{out}(C)}, \qquad W_i(C) = \frac{p_{in}(C)}{p_{in}(C) + p_{ex}(C)}.$$

These metrics are measures of how intense the communication within the cluster is, relative to that outside the cluster; they can be efficiently updated locally, i.e. the metric may be updated by knowing only the connectivity of the one node that is added or removed (which improves the efficiency of the algorithms). A *set-difference* function δ is a metric that measures the difference between two clusters C_1, C_2 . Two useful set-difference functions are the *Hamming or edit distance* δ_h , and the *percentage non-overlap* δ_p :

$$\delta_h(C_1,C_2) = \left|(C_1 \cap C_2)\right| \cup \left|(C_1 \cap C_2)\right|, \qquad \delta_p(C_1,C_2) = 1 - \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}.$$

The ε *-neighborhood* of a cluster $B_\varepsilon^\delta(C)$ is the set of clusters that are within ε of *C* with respect to δ , i.e., $B_\varepsilon^\delta(C) = \{C' \mid \delta(C,C') \le \varepsilon\}$. For weight function *W*, we say that a cluster C^* is ε *-locally optimal* if $W(C^*) \ge W(C)$ for all $C \in B_\varepsilon^\delta(C^*)$.

We are now ready to formally state our abstraction of the problem of finding overlapping communities in a communication network. The input is a graph *G*, the communication graph, along with the functions *W*, δ and ε . The output is a set of clusters $\mathrm{O} \subseteq \mathrm{K}$ such that $C \in \mathrm{O}$ *iff C* is ε -locally optimal. While our heuristic approaches are easily adapted to different weight and set-difference functions, we will focus on the choices $W = W_e$, $\delta = \delta_h$ and $\varepsilon = 1$, referring to the output clusters as locally optimal.

As stated, the problem is NP-hard. In fact, the restriction to $\delta = \delta_h$ and $\varepsilon = |V|$ asks to find all the globally optimal clusters according to an arbitrary weight function *W*, which is well known to be NP-hard. Thus, we present heuristic, efficient (low-order polynomial time) algorithms that output candidate (overlapping) clusters, and then evaluate the quality of the output (see Section 4).

3. ALGORITHMS

3.1 Iterative Scan (IS)

Algorithm IS explicitly constructs clusters that are local maxima w.r.t. a density metric by starting at a "seed" candidate cluster and updating it by adding or deleting one vertex at a time as long as the metric strictly improves. The algorithm stops when no further improvement can be obtained with a single change. Different local maxima can be obtained by restarting the algorithm at a different seed, and or changing the order in which vertices are examined for cluster updating. One continues to change the seed until some heuristic stopping condition is met. In our final algorithm, the seed is some randomly chosen edge, and the stopping condition is met if the last *max_fail* restarts yielded identical maxima to clusters already obtained. Our procedure assumes the Hamming set-difference metric, δ_h . The algorithm terminates if the addition to *C* or deletion from C of a single vertex does not increase the weight. During the course of the algorithm, the cluster C follows some sequence, $C_1, C_2, \ldots$, with the property that $W(C_1) < W(C_2) < \cdots$, where all the inequalities are strict. This means that no cluster can appear twice in the sequence. Since the number of possible clusters is finite, we conclude that the algorithm must terminate when started on *any* seed, and the set of clusters output will be a subset of the set of all locally optimal clusters. This algorithm is given in pseudo-code format to the right.

```
procedure IS(seed,$G$,W)
$C \leftarrow$ seed; $w \leftarrow W(C)$;
increased $\leftarrow$ TRUE;
while increased do
  for all $v \in V(G)$ do
    if $v \in C$ then
      $C' \leftarrow C \setminus \{v\}$;
    else
      $C' \leftarrow C \cup \{v\}$;
    if $W(C') > W(C)$ then
      $C \leftarrow C'$;
  if $W(C) = w$ then
    increased $\leftarrow$ FALSE;
  else
    $w \leftarrow W(C)$;
return $C$;
```

In our algorithm we can allow a desired cluster size to be enforced heuristically by incorporating this desire into the weight function by adding a penalty to clusters with size outside the desired range. Such an approach will not impose hard boundaries on the cluster size. If the desired range is $[C_{\min}, C_{\max}]$, then a simple penalty function $Pen(C)$, that linearly penalizes deviations from this range is

$$Pen(C) = \max\left\{0, h_1 \cdot \frac{C_{\min} - |C|}{C_{\min} - 1}, h_2 \cdot \frac{|C| - C_{\max}}{|V| - C_{\max}}\right\},$$

where $C_{\min}, C_{\max}, h_1, h_2$ are user specified parameters. All the user specified inputs are summarized in Table 1.

Table 1. User specified inputs to IS algorithm

Parameter	Description
W	Weight function.
δ	Set-difference function ($\delta = \delta_h$ in our implementation).
ε	Size of set neighborhood ($\varepsilon = 1$ in our implementation).
max_fail	Number of unsuccessful restarts to satisfy stopping condition.
$[C_{\min}, C_{\max}]$	Desired range for cluster size.
h_1, h_2	Penalty for a cluster of size 1 and $\|V\|$.

We emphasize that algorithm IS can be used to improve any seed cluster to a locally optimal one. Instead of building clusters from random edges as a starting point, we can refine clusters that are output by some other algorithm – these input clusters might be good "starting points", but they may not be locally optimal. IS then refines them to a set of locally optimal clusters.

3.2 Rank Removal (RaRe)

Algorithm RaRe is based on the assumption that within a communication network, there is a subset of "important" or high-ranking nodes, which do a significant amount of communication. RaRe attempts to identify these nodes and remove them from the graph, in order to disconnect the graph into smaller connected components. The removed node(s) are added to a set R. This process is repeated, until the sizes of the resulting connected components are within a specified range. These connected components can be considered the *core* of each cluster. Next, the vertices in R are considered for addition into one or more of these cores. If a vertex from R is added to more than one cluster, then these clusters now overlap. Note, however, that the cores of each cluster are disjoint, and only communicate with each other through vertices in R.

"Important" or high-ranking nodes are determined by a ranking function ϕ. These are the nodes which are removed at each iteration. We wish to remove nodes that will result in disconnecting the graph as much as possible. One choice is to remove vertices with high degree, corresponding to the choice $\phi_d(v) = \deg(v)$. Another approach that we have found to be experimentally better is to rank nodes according to their page rank, $\phi_p(v)$ [15]. The page rank of a node is defined implicitly as the solution to the following equation,

$$\phi_p(v) = c \sum_{u,v} \frac{\phi_p(v)}{\deg^-(v)} + \frac{1-c}{n}$$

where *n* is the number of nodes in the graph, $\deg^-(v)$ is the out degree of vertex *v*, and *c* is a decay factor between 0 and 1. An iterative algorithm to compute $\phi_p(v)$ for all the nodes converges rapidly to the correct value.

Once we have obtained the cores, we must add the vertices in *R* back into the cores to build up the clusters. Intuitively, a vertex $v \in R$ should be part of any cluster to which it is immediately adjacent, as it would have been part of the core if it were not removed at some step. Also, if we do not take this approach, we run the risk of *v* not being added to any cluster, which seems counter-intuitive, as *v* was deemed "important" by the fact that it was at one time added to *R*. This is therefore the approach which we take. We also add vertices in *R* to any cluster for which doing so increases the metric *W*. The algorithm is summarized in Figure 1, and all the user specified inputs are summarized in Table 2.

```
procedure RaRe($G, W$)
global $R \leftarrow \emptyset$;
$\{H_i\}$ are connected components in $G$;
for all $H_i$ do
    ClusterComponent($H_i$);
Initial clusters $\{C_i\}$ are cluster cores;
for all $v \in R$ do
    for all Clusters $C_i$ do
        Add $v$ to cluster $C_i$ if $v$ is adja-
        cent to $C_i$ or $W(v \cup C_i) > W(C_i)$;

procedure ClusterComponent($H$)
if $|V(H)| > max$ then
    $\{v_i\}$ are $t$ highest rank nodes in $H$;
    $R \leftarrow R \cup \{v_i\}$; $H \leftarrow H \setminus \{v_i\}$;
    $\{F_i\}$ are connected components in
    $H$;
    for all $F_i$ do
        ClusterComponent($F_i$);
else if $min \le |V(H)| \le max$ then
    mark $H$ as a cluster core;
```

Figure 1. Algorithm RaRe.

Table 2. User specified inputs for RaRe algorithm

Parameter	Description
W	Weight function.
ϕ	Ranking function.
min, *max*	Minimum and maximum core sizes.
t	Number of high-ranking vertices to remove.

It is important to note that the initial procedure of removing vertices, though not explicitly attempting to optimize any single metric, does produce somewhat intuitive clusters. The cores that result are mutually disjoint and non-adjacent. Consider a connected component *C* at iteration *i*. If *C* has more vertices than our maximum desired core size *max*, we remove a set R_i of vertices, where $|R_i| = t$. If the removal of R_i results in disconnecting *C* into two or more connected components $C_1, C_2, \ldots, C_k$, we have decreased the diameter of $C_1, C_2, \ldots, C_k$ with respect to *C*, resulting in more compact connected components. If the removal of R_i does not disconnect the graph, we simply repeat the procedure on the remaining graph until it either becomes disconnected or its size is less than *max*.

As an added performance boost, the ranks may be computed initially, but not recomputed after each iteration. The idea is that if the set R' is being removed, the rank of a vertex *v* in *G* will be close to the rank of *v* in $G - R'$.

3.3 *k*-Neighborhood (*k*-N)

k-N is a trivial algorithm that yields overlapping clusters. The clusters are simply the *k*-neighborhoods of a randomly selected set *S* of cluster centers. The inputs to this algorithm are *k* and $|S|$.

4. EXPERIMENTS

We ran a series of executions to determine the empirical runtimes of both the RaRe and IS algorithms. The summary of the results is in Figure 2. In general, IS runs in less time than RaRe, and both seem to be quadratic in the number of nodes in the graph.

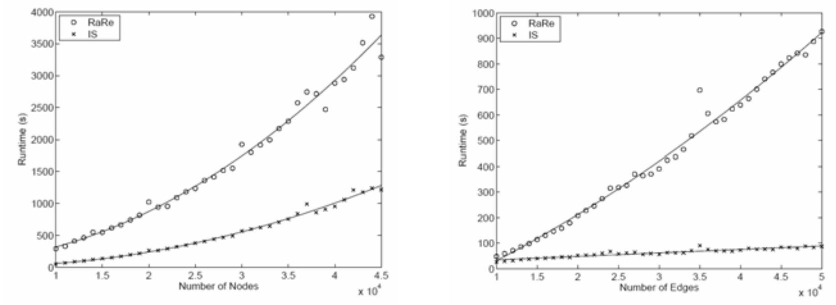


Figure 2. The runtime of the RaRe and IS algorithms in terms of increasing the number of nodes and edges.

To determine the validity of these algorithms, we performed a range of experiments on them. The density metric used in these tests is the edge ratio metric W_e described previously, with one modification. The penalty function $Pen(C)$ described for the IS algorithm was subtracted from the density in order to penalize excessively small and large clusters. The IS algorithm was run with random edges as seeds and parameter values *max_fail* = 5, $C_{\min}$ = 5, $C_{\max}$ = 20, h_1 = 0.1, and h_2 = 1. The RaRe algorithm was run with parameter values $\phi = \phi_p$, *t* = 15, *min* = 3, and *max* = 15. The *k*-N algorithm was set to find 100 clusters with *k* = 2. The results of both the RaRe and 2-N algorithms were further refined by IS. This is denoted by RaRe→IS and 2-N→IS.

The first experiment tested the algorithms on $G(n, p)$ random graphs. In order to assess the quality of the clusters, we computed the mean and standard deviation of the edge ratio density metric among all clusters found (we present only the mean here). It is more desirable to achieve a high average of this metric, and also to find a significant number of clusters. Table 3 shows the results.

An extension of this experiment was to test the algorithms on what we call *group random graphs*. These randomly generated graphs uniformly select *g*-vertex subsets of size *m*, with overlap allowed. If two vertices belong to at least one group together, directed edges are added between them with probability p_{in}. Otherwise, edges are added with probability p_{out}, where in general we take $p_{out} < p_{in}$. This simulates group structure where nodes are more likely to be well-connected to members of the same group than with nodes outside their groups. With these graphs there is also the added advantage that we know what we expect to be the true clusters in the graph.

Table 3. Synthetic data. The first entry is the average value of W_e. The two entries in parentheses are the average number of clusters found and the average number of nodes per cluster. The fourth entry (if available) is the average clustering accuracy. The $G(n,p)$ graphs were created with n=2000, p=0.001. The group random graphs had parameters n=1000, m=20, g=200, p_{in}=0.04, and p_{out}=0.0008. For preferential attachment graphs, n=2000 and d=7.

Algorithm	$G(n, p)$	Group Random	Pref. Attach
RaRe	0.24 (235,17)	0.13 (165,20); 0.096	0.01 (907,59)
RaRe→IS	0.40 (235,36)	0.23 (165,48); 0.080	0.09 (907,381)
IS	0.39 (987,41)	0.22 (961,57); 0.022	0.12 (47,99)
2-N	0.24 (100,22)	0.11 (100,73); 0.053	0.00 (100,1601)
2-N→IS	0.39 (100,53)	0.22 (100,92); 0.045	0.03 (100,1238)

One method of analyzing the accuracy of a clustering involves matching found clusters as closely as possible to known clusters. Let the known clusters be $C_1, C_2, \ldots, C_h$, and the found clusters be $S_1, S_2, \ldots, S_t$. Select the pair that have the smallest distance $\delta_p(C_k, S_l)$, and let λ_1 equal that distance. Remove these closest pair of sets and repeat, greedily finding the best match. When one of the sets of clusters is exhausted, let $\lambda_i = 0$ for $\min\{h,t\}+1 \le i \le \max\{h,t\}$ as a penalty for not having the same number of clusters. Then compute the average of all values of λ_i to determine the distance of the found clusters from the known clusters. In Table 3 we do not show distance but rather accuracy, which is defined as $1-\sum \lambda_i / \max\{h,t\}$. Note that since the accuracy metric is rigorous, and the algorithms are not given the number of clusters that should be found, this severely limits the level of accuracy obtained. The accuracy is mainly for comparison among algorithms.

The algorithms were also tested on a simple preferential attachment model of random graphs. Here, we begin with a small $G(n,p)$ random graph and add nodes one at a time, each time adding exactly d edges from the new node to existing nodes. A new edge points to an existing node v with probability proportional to $\deg(v)+1$. Here, as in the $G(n,p)$ case, there are no known clusters to compare to, so we only compare W_e and cluster sizes (See Table 3).

In addition to these synthetic graphs, each algorithm was run on a number of graphs derived from real-world data. These results are collected in Table 4. In the CiteSeer graph, papers in a common cluster were verified to relate to each other, and there were a number of papers that were members of more than one cluster. The algorithms more naturally allow one paper to appear in more than one cluster, such as in the case where the same paper is referenced in two or more different fields.

Table 4. Real data. The format is as in Table 3. The CiteSeer graph [2] is a 575,600 node set of papers where an edge corresponds to a citation. DBLP is a citation graph for the Digital Bibliography and Library Project (1,392 nodes). The e-mail graph represents e-mails among the RPI community on a single day (6,514 nodes). The web graph is a network representing the domain www.cs.rpi.edu/~magdon (701 nodes). In the newsgroup graph, edges represent replies to posts on the alt.conspiracy newsgroup (4,526 nodes).

Algorithm	Citeseer	DBLP	E-mail	Web	Newsgroup
RaRe	0.35(4474,60)	0.83(192,5)	0.20(229,35)	0.63(35,19)	0.10(1193,26)
RaRe→IS	0.42(4474,27)	0.91(192,7)	0.46(229,67)	0.77(35,35)	0.44(1193,197)
IS	0.40(9632,25)	0.91(77,10)	0.42(444,95)	0.69(25,31)	0.40(393,146)
2-N	0.24(100,14)	0.77(100,10)	0.16(100,697)	0.65(100,31)	0.20(100,168)
2-N→IS	0.56(100,36)	0.93(100,13)	0.42(100,843)	0.76(100,59)	0.51(100,438)

Figure 3 shows a visualization of some of the clusters discovered by the IS, RaRe, and 2-N algorithms on the graph from [1].

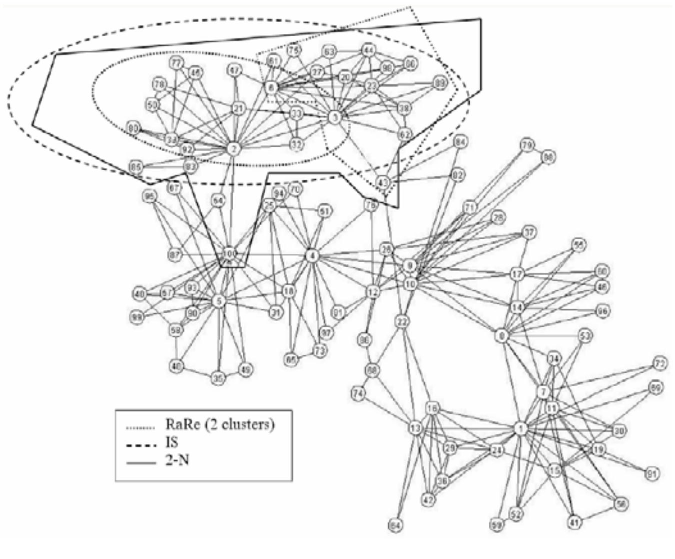


Figure 3. A comparison of clusters found in one section of a semantic web graph. IS - 26 nodes, 76 edges, W_e=0.927. 2-N - 31 nodes, 84 edges, W_e =0.808. RaRe - 14/14 nodes, 37/41 edges, W_e =0.597/0.631

5. DISCUSSION AND FUTURE WORK

We have formulated the problem of finding clusters with overlapping vertices. We have also defined metrics for comparing algorithms for solving this problem. Five different heuristics have been provided and tested with both real and synthetic graphs. We find that RaRe improved by IS (Rank Removal algorithm improved by Iterative Scan) performed the best among both synthetic and real data. The 2-N performed the poorest, but significant improvements were found when the clusters were then refined by IS.

There is much that may be done to expand on the work presented. The weight functions defined here are only a few examples of metrics that may be optimized; these may be refined or entirely new functions may be created. In particular, the weight function W_e used here was not an excellent predictor of the accuracy of a clustering when clusters were known in the random group graph model (See Table 3). There may also be other approaches to designing algorithms that find locally optimal clusters. Future work will also involve applying this formulation to graphs with evolving communities.

REFERENCES

[1] http://www.backspaces.net/oreilly/blogs/DuncanAndMark/TdS+100-2-1.8.png

[2] http://citeseer.ist.psu.edu/

[3] J. Berry and M. Goldberg, 1999. Path optimization for graph partitioning problem. *Discrete Applied Mathematics*, Vol. 90, pp. 27-50.

[4] J. C. Bezdek, 1974. Cluster validity with fuzzy sets. *Journal of Cybernatics*.

[5] P. Drineas, et al., 1999. Clustering in large graphs and matrices. *In Proceedings ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

[6] J. C. Dunn, 1973. A fuzzy relative of the isodata process and its use in detecting compact, well separated clusters. *Journal of Cybernatics*.

[7] P. Erdos and A. Rényi, 1959. On random graphs. *Publ. Math. Debrecen*, Vol. 6, pp. 290–297.

[8] G.W. Flake, K. Tsioutsiouliklis, and R.E. Tarjan, 2002. Graph clustering techniques based on minimum cut trees. Technical report, NEC, Princeton, NJ.

[9] E. E. Gustafson and W. C. Kessel, 1979. Fuzzy clustering with a fuzzy covariance matrix. *In Proc. IEEE CDC*.

[10] E-H Han, et al., 1997.. Clustering based association-rule hypergraphs. Technical report, University of Minnesota.

[11] A. K. Jain and R. C. Dubes, 1988. *Algorithms for Clustering Data*. Prentice-Hall.

[12] L. Kaufman and P. J. Rousseeuw, 1990. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley and Sons.

[13] B. W. Kernighan and S. Lin, 1970. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, Vol. 49, pp. 291–307.

[14] M. E. J. Newman, 2003. The structure and function of complex networks. *SIAM Reviews*, Vol. 45, No. 2, pp. 167–256.

[15] L. Page, et al., 1998.. The pagerank citation ranking: Bringing order to the web. Stanford Digital Libraries Working Paper.

[16] A.K. Pujari, 2001. *Data Mining Techniques*. University Press, India.

[17] S. J. Roberts, 1997. Parametric and non-parametric unsupervised cluster analysis. *Pattern Recognition*.

[18] L. A. Zadeh, 1965. Fuzzy sets. *Information and Control*.

[19] T. Zhang, R. Ramakrishnan, and M. Livny. Birch, 1996. An efficient data clustering method for very large databases. *In Proc. ACM SIGMOD Conference on Management of Data*.