Efficient Identification of Overlapping Communities*

Jeffrey Baumes, Mark Goldberg, and Malik Magdon-Ismail

Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180, USA.
Email: {baumej,goldberg,magdon}@cs.rpi.edu.

Abstract. In this paper, we present an efficient algorithm for finding overlapping communities in social networks. Our algorithm does not rely on the contents of the messages and uses the communication graph only. The knowledge of the structure of the communities is important for the analysis of social behavior and evolution of the society as a whole, as well as its individual members. This knowledge can be helpful in discovering groups of actors that hide their communications, possibly for malicious reasons. Although the idea of using communication graphs for identifying clusters of actors is not new, most of the traditional approaches, with the exception of the work by Baumes et al, produce disjoint clusters of actors, de facto postulating that an actor is allowed to belong to at most one cluster. Our algorithm is significantly more efficient than the previous algorithm by Baumes et al; it also produces clusters of a comparable or better quality.

1 Introduction

Individuals (actors) in a social community tend to form groups and associations that reflect their interests. It is common for actors to belong to several such groups. The groups may or may not be well publicized and known to the general social community. Some groups are in fact "hidden", intentionally or not, in the communicational microcosm. A group that attempts to intentionally hide its communication behavior in the multitude of background communications may be planning some undesirable or possibly even malicious activity. It is important to discover such groups before they attempt their undesirable activity.

In this paper, we present a novel *efficient* algorithm for finding *overlapping communities* given only the data on who communicated with whom. Our primary motivation for finding social communities is to use this information in order to further filter out the hidden malicious groups based on other criteria, see [2]. However, the knowledge of all communities can be crucial in the analysis of social behavior and evolution of the society as a whole, as well as its individual members [9].

The need for an automated tool to discover communities using only the presence or absence of communications is highlighted by the sheer impracticality

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

of conducting such a discovery by looking at the volumes of the actual contents of the communications, or trying to interview actors regarding the social groups to which they belong (or think they belong). The idea of using communication graphs for identifying "clusters" of users is not new. It is the guiding principle for most classical clustering algorithms such as distance-based algorithms [7]; flow-based algorithms [6]; partitioning algorithms [3, 8]; and matrix-based algorithms that employ the SVD-technique [5]. The main drawback of these approaches is that they all produce *disjoint clusters* of actors, de facto postulating that an actor is allowed to belong to at most one cluster. This is a severe limitation for social communication networks. A serious need exists for efficient tools to produce *overlapping communities* or *clusters*.

The mathematical formulation of the problem of determining clusters that may possibly be overlapping was introduced in [1], which defines a cluster as a *locally optimal* subgraph with respect to a given *metric*. We briefly review this notion of a cluster in Section 2. Since locally optimal subgraphs may overlap, this formulation allows for overlapping clusters. The algorithm for finding such locally optimal subgraphs, presented in [1] consists of two parts: **initialization**, RaRe, which creates *seed clusters*; and **improvement**, IS, which repeatedly scans the vertices in order to improve the current clusters until one arrives at a locally optimal collection of clusters.

The focus of this paper is to provide a new *efficient* algorithm for initializing the seed clusters and performing the iterative improvements. Specifically, our main contributions are a procedure List Aggregate (LA) for initializing the clusters and a procedure IS^2 which iteratively improves any given set of clusters. The combined algorithm develops overlapping subgraphs in a general graph. Our algorithm is a significant improvement over the algorithms from [1]. In particular, our algorithm can be applied to large ($\sim 10^6$) node networks. The computational experiments, comparing the new algorithm with that from [1], show that the new algorithm is an order of magnitude faster, and simultaneously produces clusters of a comparable or better quality.

2 Clusters

In this paper we adopt the idea formulated in [2] that *a group C of actors in a social network* forms a community if its communication "density" function achieves a local maximum in the collection of groups that are "close" to C. We call two groups close if they become identical by changing the membership of just one actor. Several different notions of density functions were proposed in [2]; here we consider and experiment with one more notion; specifically, the density of a group is defined as the average density of the communication exchanges between the actors of the group.

Thus, a group is a community if adding any new member to, or removing any current member from, the group decreases the average of the communication exchanges. We call a *cluster* the corresponding subgraph of the graph representing the communications in the social network. Thus, our definition reduces the task of identifying communities to that of graph clustering.

Clustering is an important technique in analyzing data with a variety of applications in such areas as data mining, bioinformatics, and social science. Traditionally, see for example [4], clustering is understood as a partitioning of the data into disjoint subsets. This limitation is too severe and unnecessary in the case of the communities that function in a social network. Our definition allows for the same actor to be a member of different clusters. Furthermore, our algorithm is designed to detect such overlapping communities.

3 Algorithms

3.1 The Link Aggregate Algorithm (LA)

```
procedure LA(G = (V, E), W)
C ← ∅;
Order the vertices v_1, v_2, ..., v_|V|;
for i = 1 to |V| do
  added ← false;
  for all D_j ∈ C do
    if W(D_j ∪ v_i) > W(D_j) then
      D_j ← D_j ∪ v_i; added ← true;
  if added = false then
    C ← C ∪ {{v_i}};
return C;
```

The IS algorithm performs well at discovering communities given a good initial guess, for example when its initial "guesses" are the outputs of another clustering algorithm such as RaRe, [1], as opposed to random edges in the communication network. We discuss a different, efficient initialization algorithm here.

The Rank Removal algorithm (RaRe) [1] begins by ranking all nodes according to some criterion, such as Page Rank [10]. Highly ranked nodes are then removed in groups until small connected components are formed (called the cluster cores). These cores are then expanded by adding each removed node to any cluster whose density is improved by adding it.

While this approach was successful in discovering clusters, its main disadvantage was its inefficiency. This was due in part to the fact that the ranks and connected components need to be recomputed each time a portion of the nodes are removed. The runtime of RaRe is significantly improved when the ranks are computed only once. For the remainder of this paper, RaRe refers to the Rank Removal algorithm with this improvement, unless otherwise stated.

Since the the clusters are to be refined by IS, the seed algorithm needs only to find approximate clusters. The IS algorithm will "clean up" the clusters. With this in mind, the new seed algorithm Link Aggregate LA focuses on efficiency, while still capturing good initial clusters. The pseudocode is given above. The nodes are ordered according to some criterion, for example decreasing Page Rank, and then processed sequentially according to this ordering. A node is added to any cluster if adding it improves the cluster density. If the node is not added to any cluster, it creates a new cluster. Note, every node is in at least one cluster. Clusters that are too small to be relevant to the particular application can now be dropped. The runtime may be bounded in terms of the number of output clusters C as follows

Theorem 1. *The runtime of* LA *is* $O(|C||E| + |V|)$.

Proof. Let C_i be the set of clusters just before the ith iteration of the loop. The time it takes for the ith iteration is $O(|C_i|deg(v_i))$, where $deg(v_i)$ is the number of edges adjacent to v_i. Each edge adjacent to v_i must be put into two classes for every cluster in C_i: either the other endpoint of the edge is in the cluster or outside it. With this information, the density of the cluster with v_i added may be computed quickly ($O(1)$) and compared to the current density. If $deg(v_i)$ is zero, the iteration takes $O(1)$ time. Therefore the total runtime is asymptotically on the order of

$$\sum_{deg(v_i)>0} |C_i|deg(v_i) + \sum_{deg(v_i)=0} 1 \leq \sum_{i=1}^{|V|} |C_i|deg(v_i) + \sum_{i=1}^{|V|} 1$$

$$\leq \sum_{i=1}^{|V|} |C|deg(v_i) + |V| = 2|C||E| + |V| = O(|C||E| + |V|).$$

3.2 Improved Iterative Scan Algorithm (IS^2)

The original algorithm IS explicitly constructs a cluster that is a local maximum w.r.t. a density metric by starting at a "seed" candidate cluster and updating it by adding or deleting one node at a time as long as the metric strictly improves. The algorithm stops when no further improvement can be obtained with a single change. The original process consists of iterating through the entire list of nodes over and over until the cluster density cannot be improved.

The new algorithm IS^2, based on IS, is given in pseudocode format to the right. In order to decrease the runtime of IS, we make the following observation. The only nodes capable of increasing the cluster's density are the members of the cluster itself (which could be removed) or members of the cluster's immediate neighborhood, defined by those nodes adjacent to a node inside the cluster. Thus, rather than visiting each node on every iteration, we may skip over all nodes except for those belonging to one of these two groups. If the neighborhood of a cluster is much smaller than the entire graph, this could significantly improve the runtime of the algorithm. Note that this algorithm is not

```
procedure IS²(seed, G, W)
C ← seed; w ← W(C);
increased ← true;
while increased do
  N ← C;
  for all v ∈ C do
    N ← N ∪ adj(v);
  for all v ∈ N do
    if v ∈ C then
      C' ← C \ {v};
    else
      C' ← C ∪ {v};
    if W(C') > W(C) then
      C ← C';
  if W(C) = w then
    increased ← false;
  else
    w ← W(C);
return C;
```

strictly the same as the original IS algorithm, since potentially a node absent from N could become a neighbor of the cluster while the nodes are being examined. This node has a chance to join the cluster in the original IS algorithm, while in IS^2 it is skipped. This is not an issue since the node will have a chance to join the cluster in the next iteration of IS^2.

This algorithm provides both a potential decrease and increase in runtime. The decrease occurs when the cluster and its neighborhood are small compared to the number of nodes in the graph. This is the likely case in a sparse graph. In this case, building the neighborhood set N takes a relatively short time compared to the time savings of skipping nodes outside the neighborhood. An increase in runtime may occur when the cluster neighborhood is large. Here, finding the neighborhood is expensive, plus the time savings could be small since few nodes are absent from N. A large cluster in a dense graph could have this property. In this case, the original algorithm IS is preferable.

4 Experiments

Algorithm	E-mail	Web
RaRe → IS	1.96 (234,9); 148	6.10 (5,8); 0.14
LA → IS^2	2.94 (19,25); 305	5.41 (6,19); 0.24
Algorithm	Newsgroup	Fortune 500
RaRe → IS	12.39 (5,33); 213	2.30 (104,23); 4.8
LA → IS^2	17.94 (6,40); 28	2.37 (288,27); 4.4

Table 1. Algorithm performance on real-world graphs. The first entry in each cell is the average value of W_{ad}. The two entries in parentheses are the average number of clusters found and the average number of nodes per cluster. The fourth entry is the runtime of the algorithm in seconds. The e-mail graph represents e-mails among the RPI community on a single day (16,355 nodes). The web graph is a network representing the domain www.cs.rpi.edu/∼magdon (701 nodes). In the newsgroup graph, edges represent responses to posts on the alt.conspiracy newsgroup (4,526 nodes). The Fortune 500 graph is the network connecting companies to members of their board of directors (4,262 nodes).

A series of experiments were run in order to compare both the runtime and performance of the new algorithm with its predecessor. In all cases, a seed algorithm was run to obtain initial clusters, then a refinement algorithm was run to obtain the final clusters. The baseline was the seed algorithm RaRe followed by IS. The proposed improvement consists of the seed algorithm LA followed by IS^2. The algorithms were first run on a series of random graphs with average degrees 5, 10, and 15, where the number of nodes range from 1,000 to 45,000. In this simple model, all pairs of communication are equally likely.

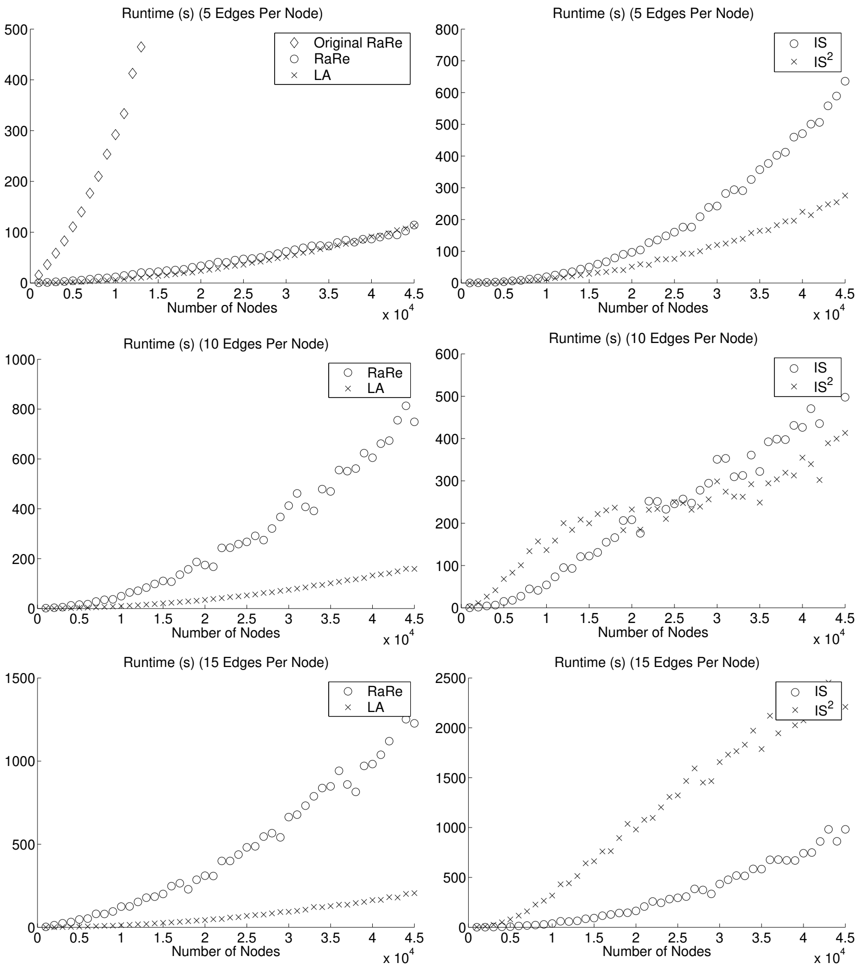

Fig. 1. Runtime of the previous algorithm procedures (RaRe and IS) compared to the current procedures (LA and IS^2) with increasing edge density. On the left is a comparison of the initialization procedures RaRe and LA, where LA improves as the edge density increases. On the right is a comparison of the refinement procedures IS and IS^2. As expected, IS^2 results in a decreased runtime for sparse graphs, but its benefits decrease as the number of edges becomes large.

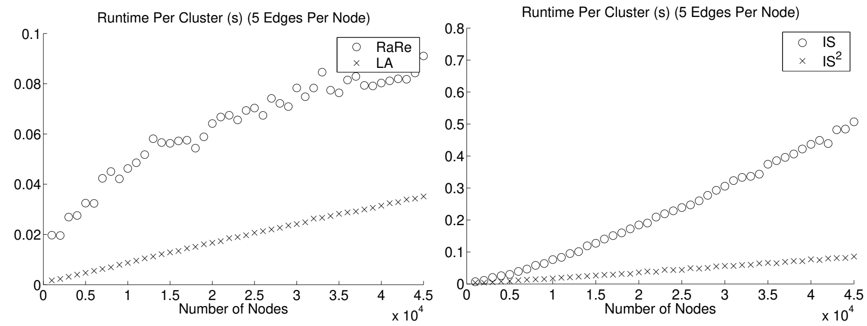

Fig. 2. Runtime per cluster of the previous algorithm (RaRe followed by IS) and the current algorithms (LA followed by IS^2). These plots show the algorithms are linear for each cluster found.

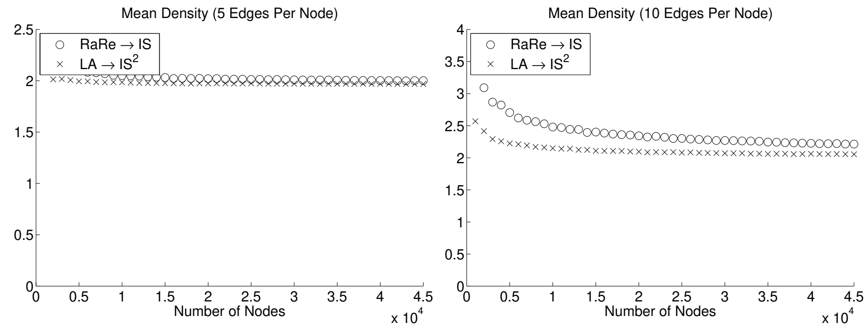

Fig. 3. Performance (average density) of the algorithm compared to the previous algorithm.

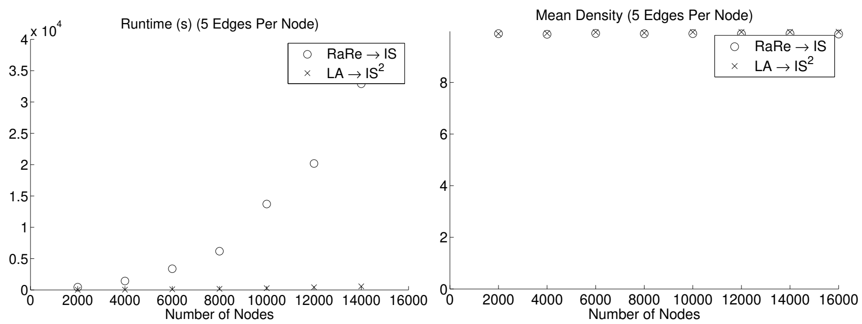

Fig. 4. Runtime and performance of the previous algorithm (RaRe followed by IS) and the current algorithm (LA followed by IS^2) for preferential attachment graphs.

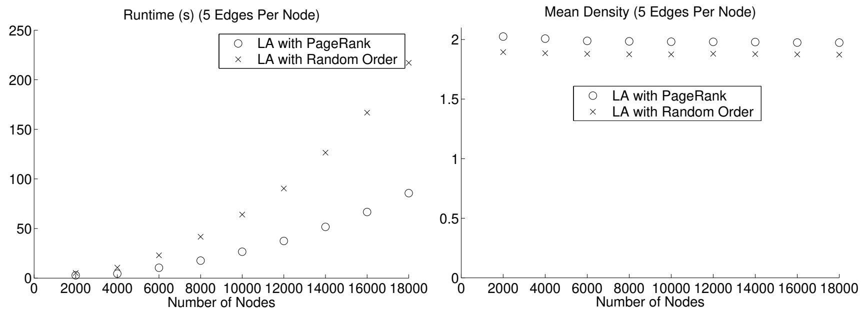

Fig. 5. Runtime and performance of LA with two different ordering types.

All the algorithms take as input a density metric W, and attempt to optimize that metric. In these experiments, the density metric was chosen as W_{ad}, called the *average degree*, which is defined for a set of nodes C as

$$W_{ad}(C) = \frac{2|E(C)|}{|C|},$$

where $E(C)$ is the set of edges with both endpoints in C.

The runtime for the algorithms is presented in Figure 1. The new algorithm remains quadratic, but both the seed algorithm and the refinement algorithm run-times are improved significantly for sparse graphs. In the upper left plot in Figure 1, the original version of RaRe is also plotted, which recalculates the node ranks a number of times, instead of precomputing the ranks a single time. LA is 35 times faster than the original RaRe algorithm and IS^2 is about twice as fast as IS for graphs with five edges per node. The plots on the right demonstrate the tradeoff in IS^2 between the time spent computing the cluster neighborhood and the time saved by not needing to examine every node. It appears that the tradeoff is balanced at about 10 edges per node. For graphs that are more dense, the original IS algorithm runs faster, but for less dense graphs, IS^2 is preferable.

Figure 2 shows that the quadratic nature of the algorithm is based on the number of clusters found. When the runtime per cluster found is plotted, the resulting curves are linear.

Runtime is not the only consideration when examining this new algorithm. It is also important that the quality of the clustering is not hindered by these runtime improvements. Figure 3 compares the average density of the clusters found for both the old and improved algorithms. A higher average density indicates a clustering of higher quality. Especially for sparse graphs, the average density is approximately equal in the old and new algorithms, although the older algorithms do show a slightly higher quality in these random graph cases.

Another graph model more relevant to communication networks is the preferential attachment model. This model simulates a network growing in a natural way. Nodes are added one at a time, linking to other nodes in proportion to the degree of the nodes. Therefore, popular nodes get more attention (edges), which is a common phenomenon on the web and in other real world networks. The resulting graph has many edges concentrated on a few nodes. The algorithms were run on graphs using this model with five links per node, and the number of nodes ranging from 2,000 to 16,000. Figure 4 demonstrates a surprising change in the algorithm RaRe when run on this type of graph. RaRe removes high-ranking nodes, which correspond to the few nodes with very large degree. When these nodes are added back into clusters, they tend to be adjacent to most all clusters, and it takes a considerable amount of time to iterate through all edges to determine which connect to a given cluster. The algorithm LA, on the other hand, starts by considering high-ranking nodes before many clusters have formed, saving a large amount of time. The plot on the right of Figure 4 shows that the quality of the clusters are not compromised by using the significantly faster new algorithm $\mathsf{LA} \rightarrow \mathsf{IS}^2$.

Figure 5 confirms that constructing the clusters in order of a ranking such as Page Rank yields better results than a random ordering. LA performs better in terms of both runtime and quality. This is a surprising result since the random ordering is obtained much more quickly than the ranking process. However, the first nodes in a random ordering are not likely to be well connected. This will cause many single-node clusters to be formed in the early stages of LA. When high degree nodes are examined, there are many clusters to check whether adding the node will increase the cluster density. This is time consuming. If the nodes are ranked, the high degree nodes will be examined first, when few clusters have been created. These few clusters are likely to attract many nodes without starting a number of new clusters, resulting in the algorithm completing more quickly.

The algorithms were also tested on real-world data. The results are shown in Table 1. For all cases other than the web graph, the new algorithm produced a clustering of higher quality.

5 Conclusions

We have described a new algorithm for the discovery of overlapping communities in a communication network. This algorithm, composed of two procedures LA and IS^2, was tested on both random graph models and real world graphs. The new algorithm is shown to run significantly faster than previous algorithms presented in [1], while keeping the cluster quality roughly the same and often better. In addition, we demonstrated that the LA procedure performs better when the nodes are ranked, as opposed to a random order. Surprisingly, though the ranking process initially takes more time, the procedure runs more quickly overall.

Directions for our ongoing work are to test different metrics, and to apply the algorithms to a variety of networks ranging from social communication networks to protein networks, ranging in sizes from hundreds of nodes to a million nodes. There are a variety of options for parameter settings available to the user, and it will be useful to provide the practitioner with an exhaustive database of test-cases, giving guidelines for how to set the parameters depending on the type of input graph.

References

1. J. Baumes, M. Goldberg, M. Krishnamoorty, M. Magdon-Ismail, and N. Preston. Finding communities by clustering a graph into overlapping subgraphs. *Proceedings of IADIS Applied Computing 2005*, pages 97–104, February 2005.
2. J. Baumes, M. Goldberg, M. Magdon-Ismail, and W. Wallace. Discovering hidden groups in communication networks. *2nd NSF/NIJ Symposium on Intelligence and Security Informatics*, 2004.
3. J. Berry and M. Goldberg. Path optimization for graph partitioning problem. *Discrete Applied Mathematics*, 90:27–50, 1999.
4. U. Brandes, M. Gaertler, and D. Wagner. Experiments on graph clustering algorithms. *Lecture Notes in Comuter Science*, Di Battista and U. Zwick (Eds.):568–579, 2003.
5. P. Drineas, R. Kannan, A. Frieze, S. Vempala, and V. Vinay. Clustering in large graphs and matrices. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1999.
6. G.W. Flake, K. Tsioutsiouliklis, and R.E. Tarjan. Graph clustering techniques based on minimum cut trees. Technical report, NEC, Princeton, NJ, 2002.
7. A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.
8. B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49:291–307, 1970.
9. M. E. J. Newman. The structure and function of complex networks. *SIAM Reviews*, 45(2):167–256, June 2003.
10. L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. *Stanford Digital Libraries Working Paper*, 1998.