Overlapping Community Detection by Collective Friendship Group Inference

Bradley S. Rees
Department of Computer Science
Durham University
Durham, UK
b.s.rees@durham.ac.uk

Keith B. Gallagher
Department of Computer Science
Florida Institute of Technology
Melbourne, Florida, USA
kgallagher@fit.edu

***Abstract*—There has been considerable interest in improving the capability to identify communities within large collections of social networking data. However, many of the existing algorithms will compartment an actor (node) into a single group, ignoring the fact that in real-world situations people tend to belong concurrently to multiple groups. Our work focuses on the ability to find overlapping communities by aggregating the community perspectives of friendship groups, derived from egonets. We will demonstrate that our algorithm not only finds overlapping communities, but additionally helps identify key members, which bind communities together. Additionally, we will highlight the parallel feature of the algorithm as a means of improving runtime performance.**

Keywords-component; Network clustering, community detection, social networks, egonets, complex networks

I. INTRODUCTION

An escalation in the number of Community Detection [5-15, 22-26] algorithms has occurred in recent years, with the focus of the algorithms shifting away from the classical clustering principles of grouping nodes based upon some type of shared attribute or characteristic [1, 6], to one where the relationship and interactions between individuals is of importance. That shift has caused algorithms to start viewing the data as a graph and focus on exploiting (detecting) the "small-world effect" [16] found in social networks - the phenomena that a small path length separates any two randomly selected nodes - and on detecting the clustering property of social networks where the density of the edges is higher within the group than between the groups [5-15, 22].

Moody and White [4] reasoned that communities are held together by the presence of multiple independent paths between members. Extrapolating from the goal of discovering clusters, where internal edge density is maximized, it follows that the identification of cliques [9, 13, 20] {k-cliques, k-clans, or k-cores, where k is the number of nodes comprising the group} would be a viable approach, as density is maximal within those structures. However, given that a 5-clique, for example, contains a number of overlapping 4-cliques, each of which is a community in its own right [20], presents the question of whether the algorithm is really revealing communities or just doing pattern matching.

Other approaches have focused on centrality [3] to identify key nodes or edges, and follow a hierarchical clustering approach to recursively extract clusters [5,9,11]. While centrality is a powerful and useful idea for identifying key (central) actors in a network, be it a node or an edge, many of the centrality approaches require that the centrality measurement be recalculated after each graph edit, causing the algorithms to be highly inefficient, running in $O(n^3)$ or worse [6, 8,11].

In this paper, we propose a radically different approach to group detection that finds communities based on the collective viewpoint of individuals. The notion postulated is that each node in the network knows, by way of its egonet [18, 19], who is in its friendship groups. Therefore, by aggregating each individual's views of friendship groups, communities can be uncovered. We use the term "friendship group" to represent the small clusters, extracted from egonets, containing the central node and connected neighbors. Additionally, the algorithm is designed to be highly parallelizable as a means of improving runtime.

A. Terminology

A graph is defined as G = {V, E} where V is a set of vertices (nodes) and E is a set if edges. The neighbors of a vertex, v, is defined as the set of vertexes connected by way of an edge to that vertex N(v) = {U} where $v \in V$ and $\forall u \in U \exists \text{edge}(v,u) \in E$. The degree of a vertex, δ(v), is the number of edges incident to that vertex. In the case where the graph contains no loops, edges that have the same starting and ending vertex, the degree of a vertex is also equal to the number of neighbors, δ(v)=|N(v)|. The density of a graph, or subgraph, is the measure of the number of edges in the graph, over the maximum number of possible edges.

The term egonet derives from egocentric network, and is the view of the network from an individual node's (actor) point of view, with that node at the center. That individual's viewpoint reduces the network to just those vertices adjacent to the central "ego" node and any edges between those nodes. A more formal definition of an egonet {ego network, ego-network, or egocentric network} [2, 18, 19] is an induced subgraph consisting of a central node, its neighbors, and all edges among nodes in the ego-net that also existed in the main graph.

All graphs in this work are considered to be unweighted, undirected, and containing no loops. The algorithm could be modified to work with weighted and/or directed graphs.

II. RELATED WORK

One of the more prevalent algorithms comes from work by Girvin & Newman [5, 6] (GN). The GN algorithm follows a divisive hierarchical method, which iteratively removes edges with the highest edge-betweenness centrality score. This is based on the principle that between community edges have higher centrality than within community edges. The algorithm

978-0-7695-4138-9/10 $26.00 © 2010 IEEE
DOI 10.1109/ASONAM.2010.28

was a major step forwards by recognizing that the centrality score must be recalculated after each edge removal. However, the recalculating of centrality causes the algorithm to have high computational demands, running in $O(n^3)$ to $O(n^4)$ time on sparse graphs. Newman addressed the performance factor in a subsequent paper [8] by developing a modularity function that reduced runtime to $O(n^2)$. The "fast-Newman" algorithm employs an agglomerative hierarchical approach that starts with a graph of just nodes and iteratively adds edges that increase the overall modularity (Q) values.

However, the previous approaches present some problems. As Newman points out [8] *"... the GN community structure algorithm always produces some division of vertices into communities, regardless of whether the network has any natural such divisions*." That problem is characteristic of hierarchical clustering [6] and the "fast-Newman" algorithm suffers a similar problem. The selection of an optimal Q value, an NP-complete problem [23], for modularity can have a significant affect on the number of clusters detected.

The notion of using some form of centrality as the means for determining which edges to cut was extended by Hwang et al. [7], with the concept of Bridging Centrality. A bridge, in graph theory terms, is an edge whose removal will break the graph into two subgraphs. It follows that communities would be connected by a number of bridges. Hwang et al. defined Bridging Centrality as the ranked product of betweenness centrality and a bridging coefficient, with bridging coefficient being the probability that an edge leaves the community. The runtime for their algorithm is $O(nm + n^2 logn)$, where n is the number of nodes and m is the number of edges in a network.

Agglomerative and Divisive methods both process from one extreme of the graph to the other. These processes iterations can be represented as a dendrogram, or tree. Selecting different stopping points in those processes will produce different numbers of communities [6,7]. The challenge is that the decision of where to stop should to be done apriori. The following illustration (Figure 1) shows a dendrogram with three possible cut points (A, B, and C), producing two, four, or six possible cluster, each of which doesn't necessarily equate to a community [22]. Modularity and density have both been used as means of determining stopping point [6, 9].

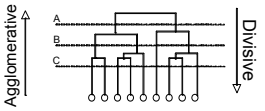


Figure A

While performance gains have been made, the previously mentioned approaches suffer the additional problem that nodes are forced to exist only in a single community. Real-world networks are not so nicely constrained, and contain realistic amounts of overlap between communities [4,13,25]. Each person (node) could have a community for family, friends, work, interest, etc., and community detection algorithms must allow for, and detect, overlapping groups. Forcing a node into a single community and not allowing for overlap could prevent the detection of the true underlying community structure(s) [13, 25].

A number of solutions for finding overlapping communities have been developed [11-15, 25]. Gregory [14], for example, modified the GN algorithm to highlight overlapping communities by splitting nodes, thus permitting a node to be represented in the graph multiple times, and allowing each instance of the node to clustered into a different community. While the modification does find overlapping communities, it also degrades the algorithm's performance.

The notion of a clique being synonymous with a community is not new, and approaches for finding cliques originated as early as the late 1940s [20]. Palla et al. [13] extends the theory of cliques as communities by introducing the definition that a community, specifically a *k*-clique-community, is a union of all *k*-cliques that can be reached via adjacent *k*-cliques. The process works by rolling, or percolating, a *k*-clique over the network to find other *k*-cliques that share *k*-1 nodes. The percolating [26] is performed by moving the selection of one node within the *k*-clique to an unselected neighbor node that also form a *k*-clique. Since only one node is reselected each time, the subsequent *k*-clique must share exactly *k*-1 nodes.

III. Our Approach

A. Defining Community and Friendship-Group

There is no formal, or conventional, definition of social community [24] beyond "a collection of individuals linked by a common interest" [21]. Rather than redefining community, we turn to work by Moody & White [4], who focused on defining four characteristics that bind a community together, referred to as "structural cohesion." One definition of interest from Moody & White is that community cohesion is tied to the number of independent paths between members. That definition is supported by the qualitative observations [22] that communities have greater internal edge density than external, inter-community, density. Consider the graph in Figure 2, it contains two obvious communities with a single edge between them. As the number of links between communities increases, the ability of clustering algorithms to find distinct communities degrades [5]. Reexamine the graph in Figure 2, and increasing the number of edges between the two communities, Figure 3, are there still two communities, have the two merged into one, or are there three communities?

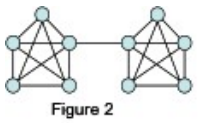


Figure 2

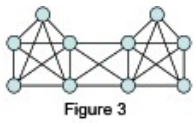


Figure 3

A second definition from Moody & White, is that the removal of one member (node) should not cause the community to collapse. Therefore, for this version of the algorithm, a group of two nodes joined by a single edge, is not being considered a community. Likewise, any node with degree of one is not being considered as part of a community. That restriction is enforced only for this version of the algorithm.

We define a "*Friendship-Group*" to be a subgraph extracted from an egonet, adhering to the same constraints mentioned above for a community; multiple paths and no dyads or single

nodes. We distinguish between the two types of communities since a friendship-group is myopic view of the egonet, and one or more friendships-groups can be combined to form a community.

B. Algorithm

The algorithm executes in two phases; the first phase is the detection of friendship groups, the second phase comprises the aggregation of friendship groups into communities.

Pseudo code:

1. For each node ($\forall\ n \in \{V\}$)
   - Get egonet of *n*: *H = ego(n)*
   - Find Friendship-Groups
     - Remove n from the egonet
     - Find the connected components
     - Add n to each component node list
2. Merge & Reduce Sets
   - Remove proper subsets
   - Merge "close" matches
   - Repeat until no more merges can be performed

In phase one, the algorithm iterates through every vertex in the graph, centering on the selected vertex and deriving the egonet. From that egonet, friendship-groups are extracted. For the first step, the central node is removed, since it is known to exist in multiple friendship groups. Therefore by removing the ego-node, the graph breaks into multiple connected components. The egocentric vertex is then added back to each found component to form the friendship groups.

For example, given the following simple network, Figure A, the egonet for vertex D would be just those vertices connected to D, or B, C, E, and F, as shown in Figure B.

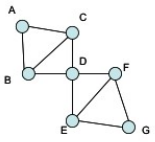


Figure A

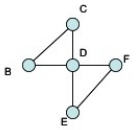


Figure B

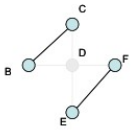


Figure C

From the point-of-view of vertex D, nodes B and C are friends and E and F are friends. The removal of D, grayed out in Figure C, creates two distinct components. That yields two friendship groups, with the ego-node added back in, of {B, C, D} and {D, E, F}. That process is repeated for every vertex in the network. The result of that first phase is a collection of friendship groups, from an egocentric point-of-view.

The next step, phase two, is to merge all the groups into cohesive sets, or communities. That process is done by first merging all exact matches, groups that are complete or proper subsets of other groups. The final step is to merge groups that are relatively close, in this case, groups that match all but one item from the smaller group. Giver two sets, S_l and S_s, where S_l is larger than, or equal to, S_s, then the sets are merged (union) if the size of the intersection is equal to one less than the size of the smaller set: $|S_l \cap S_s| = |S_s| - 1$. This step compensates for egonets not having a complete picture of the community. Continuing the example from above, group {A, B, C}, obtained from egonet centered on node A, would merge with group {B, C, D} to form {A, B, C, D}.

C. Performance

The runtime performance of the algorithm is greatly influenced by the density of the graph being analyzed. Given that an egonet is a central node and its surrounding neighbors, the number of which can be generalized as equivalent to the average degree. As the density of the graph approaches 1.0, the average degree increases to *(n-1)*. For sparse scale-free graphs, Hwang et al. [9] points out that the average degree is approximately *(log n),* which we will use for the anticipated engonet size of a sparse graph, and *(n)* for worst-case scenario. Additionally, as with any algorithm, the method of implementation can affect performance.

The first phase of the algorithm comprises the identification of friendship groups from egonets. The process of identifying the egonet can be done in constant time, since the base graph does not have to be modified. The process only needs to identify the neighbors of the selected node. If the data is stored in an edge matrix, then the neighbors are specified in the row corresponding to the ego-node.

The process of finding the egonet friendship groups, or connected components, can be done with a Breadth-First search (BFS), which runs in $O(n' + m')$ where n' is the number of nodes and m' is the number of edges in the egonet. In a sparse network, m scales with n, and for the egonet the size of n is equal to the average degree, or $(\log n)$. However, the BFS needs to be modified, such that only edges within the egonet are examined, increasing runtime to $O(\log n)^2$. Alternatively, the egonet could be extracted from the main graph and processed independently, with an additional cost. Since the process of finding friendship groups is done for each node in the network, the runtime for the first phase is $O(n\ (\log n)^2)$.

The second step is filtering and merging, which can be accomplished with a modified merge-sort algorithm. A traditional merge-sort runs in $O(n \log n)$, however the merging process in this case produces a new set of items that needs to be reexamined and compared to the remaining set. That modification increases runtime to $O(n^2 \log n)$. Unlike the friendship group find portion of the algorithm, the merge-sort performance increases as the density increases. The rationale behind this is that, as density increases, the number of similar friendship groups also increases. When density equals 1.0, every friendship group will be exactly the same, and the result set will reduce neatly to a single entry in $O(n)$ time. Total runtime performance on a sparse graph is $O(n(\log n)^2 + n^2 \log n)$. Increasing to $O(n^3 + n)$ as density approaches 1.0.

D. Improving Performance

The runtime performance shown above is not improved, and in some cases inferior to existing algorithms. Conversely, our algorithm is designed to operate in a parallel fashion as a means of improving performance and scalability.

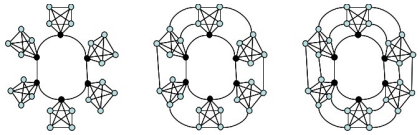
The initial phase of the algorithm is the identification of friendship groups by iterating over all nodes in the graph. Friendship groups are found for each node, independent of the other nodes, and therefore, can be performed in a parallel

fashion. The second phase is an all-pair comparison where a selected set (friendship group or community) is compared with all others to determine if the set warrants merging, deletion, or retention. As each comparison is acted independently from the previous examination, these processes can also be performed in parallel.

IV. Application

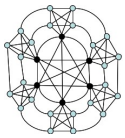
A. Caveman

The algorithm was first applied against a Caveman graph, a term coined by Watts and Strogatz [16] for a network containing a number of fully connected clusters (cliques) or "caves." The number of connections between the caves is increased to determine at which point the algorithm stops identifying the core cave groups.



In all three cases, the algorithm found the groups with no errors. Although this was a simple case and the links were not really added at random, additional links did not form any new triads and thus no additional groups were detected.

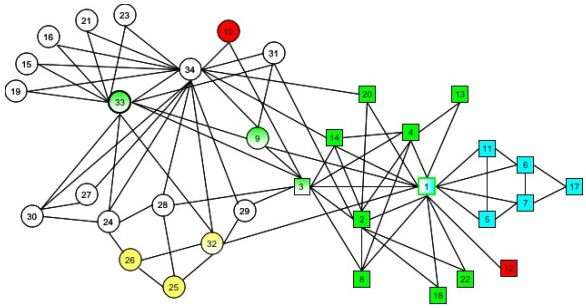
When additional links were added linking all the center nodes, the algorithm then detected the six original groups plus a new overlapping community formed by the center nodes.



B. Zachary

The Zachary [16] Karate Club dataset is well studied, and utilized as a test bed for many community detection algorithms [5-8, 10, 11, 14]. Zachary observed the social interactions of members of his karate club over a period of two years. By chance a dispute broke out between two members that caused the club to split into two smaller groups. The GN algorithm identifies the two communities within the dataset, when programmed to extract only two communities.

When our algorithm was applied to the Zachary dataset, four communities were found. The following graph illustrates the discovered networks as well as highlighting the two clubs formed after the split, shown by the circles and squares.



Cluster A: [1, 17, 7, 11, 6, 5]
Cluster B: [13, 33, 1, 4, 14, 3, 22, 20, 2, 9, 18, 8]
Cluster C: [25, 32, 26]
Cluster D: [29, 33, 1, 21, 3, 31, 9, 15, 34, 28, 24, 30, 16, 27, 32, 19, 23]
Not a member of a community: 10, 12

At first glance, it might appear that our algorithm was in error when it detected four communities in contrast with what the Zachary states as the final outcome. However, the focus of the Zachary paper was on group fission and not on communities, or overlapping communities, within the group. Additionally, the Zachary paper presented a method for creating edge weights based on an aggregation of the number of different social interaction domains at individuals attended together. Each of these domains has the possibility of defining a community.

Hierarchical clustering allows for the algorithm to be stopped at various points, producing from 1 to n clusters. Since the anticipated results were two clusters, that is the stopping point of most benchmarks against the Zachary dataset. Donetti & Muñoz [24] presented an algorithm based on modularity that stops processing when modularity is maximized. Their algorithm finds four clusters and one single node.

If the goal was to simply produce two clusters then a few additional communities merging would have to occur. Looking at Community A, this community is virtually independent from the rest of the communities, with the overlap occurring solely due to node 1. When the split in the karate group happened, this group would follow node 1 and community A would merge in with community B. Looking at the dendrogram from the GN [5] and the Donetti [24] papers, the node comprising cluster A and B are merged in the final step. Community C is less independent than A, but only has an overlap with community D at node 32 and would merge in with that cluster. A merger of cluster A with cluster B and cluster C with cluster D would produce two communities, with the only error being the overlapping nodes that appear within both communities.

Yet the purpose of our algorithm was to find overlapping communities and not graph partitioning. In detecting communities, the algorithm also identifies those nodes that form the overlap and act as brokers, or social bridges, between communities. Of particular interest from the karate club are nodes 1, 3, 9, and 33. Those four nodes appear to be the glue that held the groups together. For example, breaking the edge between nodes 3 and 33 and nodes 9 and 1 causes our algorithm to remove the overlap between the two groups.

C. Other Dataset and Follow-on Work

A number of other datasets were processed by our algorithm and are shown in the following table (Table 1). However, as the sizes of the graphs examined grew, so did the complexity of displaying and analyzing the results. Since our algorithm presents a new definition of community, it was anticipated that there would be significant deviation in results between our algorithm and what others have achieved. Identifying the most efficient methods to measure and compare our results against other algorithms, beyond a simple Jaccard similarity score, is an area for future research. As our algorithm also identifies the nodes that form the overlap, an

analysis of those nodes for their ability to act as brokers, and as structural holes, should be cultivated.

Additionally, in processing the larger datasets, a growing number of nodes not belonging to any community were detected, raising two questions that we plan to further investigate: 1) Is our assumption that a single edge node need not belong to a community valid? 2) Can link weighting be used to further cast a node into one community or another?

The main focus of this research has been on the community detection portion of the algorithm and not on the merging of the friendship groups. An investigation of that area is expected to aid in the reduction of overall runtime. Lastly, we would like to further study the parallel capabilities of the algorithm by exploring multi-threading options or rewriting the algorithm in OpenCL

Table 1

Dataset	Nodes	Edges	Avg. Degree	Density	Communities Detected
Zachary (a)	34	78	4.6	0.14	4
Jazz (b)	198	2,742	27,69	0.14	171
Email (c)	1,133	5,452	9.6	0.008	605
PGP (d)	10,680	24,316	4.55	0.26	1,308

(a) See reference 17
Dataset b - d from http://deim.urv.cat/~Eaarenas/data/welcome.htm
(b) P.Gleiser and L. Danon , Adv. Complex Syst.6, 565 (2003).
(c) Guimera, L. Danon, A. Diaz-Guilera, F. Giralt and A. Arenas, Physical Review E , vol. 68, 065103(R), (2003).
(d) M. Boguña, R. Pastor-Satorras, A. Diaz-Guilera and A. Arenas, Physical Review E, vol. 70, 056122 (2004)

V. CONCLUSION

Detection of the underlying community structure is an important part of intuitive network analysis. Failure to consider and account for overlapping groups creates a situation where the true structure can go undetected. In this paper, we have presented a new approach for detecting overlapping communities, based on the unique perspective of individual group members, which we called friendship-groups. This approach, we believe, defines a more insightful notion of community and creates a potential for future performance enhancements.

ACKNOWLEDGMENTS

The authors are grateful to Graham Cruickshank for his comments and proofreading skill.

REFERENCE

[1] Getoor, L., and Diehl, C. P. “Link mining: a survey.” SIGKDD Explorations. Newsletter. 7, 2 (2005), 3-12

[2] de Nooy, W., Mrvar, A., and Batagelj, V. “Exploratory Social Network Analysis with Pajek.” Cambridge University Press, Cambridge, (2005)

[3] Freeman, L. C. “Centrality in social networks conceptual clarification.” Social Networks 1, 3 (1979).

[4] Moody, J., and White, D. R. “Structural cohesion and embeddedness: A hierarchical concept of social groups.” American Sociological Review 68, 1 (2003), 103-127.

[5] Girvan, M., and Newman, M. E. “Community structure in social and biological networks.” Proceedings of the National Academy of Science USA 99, 12 (June 2002), 7821–7826.

[6] Newman, M. E. J. and Girvan, M. “Finding and evaluating community structure in networks”. Physical Review E. Aug 2003.

[7] Newman, M. E. J. “Detecting community structure in networks.” Eur. Phys. J. B. 38 (2004)

[8] Newman, M. E. J. “Fast algorithm for detecting community structure in networks.” Physical Review E 69, 6 (2004)

[9] Hwang, W., Kim, T., Ramanathan, M., and Zhang, A. “Bridging centrality: graph mining from element level to group level.” In KDD ’08: Proceedings of the 14th ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining, ACM, pp. 336–344. (2008)

[10] Wu, F. and Huberman, B. A. “Finding communities in linear time: A physics approach.” European Physical Journal B, 38:331–338. (2004)

[11] Du, N., Wu, B., Pei, X., Wang, B., and Xu, L. “Community detection in large-scale social networks.” In WebKDD/SNA-KDD ’07: Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis, ACM, pp. 16–25. (2007)

[12] Du, N., Wang, B., and Wu, B. “Overlapping community structure detection in networks.” In CIKM ’08: Proceeding of the 17th ACM conference on Information and knowledge management, ACM (2008)

[13] Palla, G., Derenyi, I., Farkas, I., and Vicsek, T. “Uncovering the overlapping community structure of complex networks in nature and society.” Nature 435 (2005), 814.

[14] Gregory, S. “An algorithm to find overlapping community structure in networks.” In Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2007), Springer-Verlag.

[15] Baumes, J., Goldberg, M., and Magdon-ismail, M. “Efficient Identification of overlapping communities.” In In IEEE International Conference on Intelligence and Security Informatics ISI (2005).

[16] Watts, D. J., and Strogatz, S. H. “Collective dynamics of ’small-world’ networks.” Nature 393, 6684 (June 1998), 440–442.

[17] Zachary, W. “An information flow model for conflict and fission in small groups,” Journal of Anthropological Research 33, 452-473 (1977)

[18] Freeman, L. C. “Centered graphs and the structure of ego networks.” Mathematical Social Sciences 3, 3 (October 1982), 291–304.

[19] Everett, M., and Borgatti, S. P. “Ego network betweenness.” Social Networks 27, 1 (January 2005), 31–38.

[20] Everett, M. G., and Borgatti, S. P. “Analyzing clique overlap.” CONNECTIONS 21, 1 (1998), 49–61.

[21] Marriam-Webster online dictionary http://www.merriam-webster.com

[22] Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., and Parisi, D. “Defining and identifying communities in networks.” Proceedings of the National Academy of Sciences of the United States of America 101, 9 (March 2004).

[23] Xu, X., Yuruk, N., Feng, Z., and Schweiger, T. A. J. “Scan: a structural clustering algorithm for networks.” In KDD ’07: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2007), ACM, pp. 824–833.

[24] Donetti, L., and Munoz, M. A. “Detecting network communities: a new systematic and efficient algorithm.” Journal of Statistical Mechanics: Theory and Experiment 2004, 10 (2004).

[25] Davis, G. and Carley, K. (2008). “Clearing the fog: Fuzzy, overlapping groups for social networks.” Social Networks, 30(3):201–212.

[26] Derényi, I., Palla, G., and Vicsek, T. “Clique percolation in random networks.” Physical Review Letters 94, 16 (April 2005).