Community Detection in Collaborative Environments: A Comparative Analysis *

Andreas Kalaitzakis
Department of Applied Informatics and Multimedia
TEI of Crete, 71500 Heraklion, Crete, Greece

Harris Papadakis
Department of Applied Informatics and Multimedia
TEI of Crete, 71500 Heraklion, Crete, Greece
adanar@ics.forth.gr

Costas Panagiotakis †
Department of Commerce and Marketing
TEI of Crete,72200 Ierapetra, Crete, Greece
cpanag@csd.uoc.gr

Paraskevi Fragopoulou ‡
Department of Applied Informatics and Multimedia
TEI of Crete, 71500 Heraklion, Crete, Greece
fragopou@ics.forth.gr

ABSTRACT

In this paper, we analyze and compare the performance of four different community detection algorithms, each following a different approach. The performance of the algorithms is compared on a variety of benchmark graphs with known community structure. Experiments reveal the strengths and weaknesses of the involved algorithms and demonstrate the necessity to devise local and efficient community detection techniques that perform well under a variety of changing conditions.

Categories and Subject Descriptors

I.5.3 [**Clustering**]: [Algorithms, Similarity measures]; G.2.2 [**Graph Theory**]: [Graph labeling]

1. INTRODUCTION

PETRA'11, May 25 - 27, 2011, Crete, Greece.

†Costas Panagiotakis is also with the Department of Computer Science, University of Crete, 71409 Heraklion, Crete, Greece.

‡Paraskevi Fragopoulou is also with the Foundation for Research and Technology-Hellas, Institute of Computer Science, 70013 Heraklion, Crete, Greece.

Networks in various application domains present an internal structure, where nodes form groups of tightly connected components which are more loosely connected to the rest of the network. These components are mostly known as *communities*, *clusters*, or *groups*, terms used interchangeably in the rest of this paper. Uncovering the community structure of a network is a fundamental problem in complex networks. With the advent of Web 2.0 technology, came along the emerging need to analyze network structures like web communities, social network relations, and in general user's collective activity in collaborative environments. A *strong community* is defined as a group of nodes for which each node has more edges to nodes of the same community than to nodes outside the community [1]. This definition is relatively strict, since it does not allow for overlapping communities and creates a hierarchical community structure, since the entire graph can be a community itself. A *generalized community*, is defined as a subgraph in which the sum of all node degrees within the community is larger than the sum of all node degrees towards the rest of the graph [3]. Other definitions are related to the optimization of some "fitness" criterion like for example the intracommunity edge density [7], or the intercommunity edge cut [1]. In general, community detection is the problem of finding a partition of a graph into subgraphs that maximizes some quality criterion which reflects the density of the subgraph(s).

Variations appear in the methodology used to identify communities. Certain algorithms follow an iterative approach starting by characterizing either the entire network, or each individual node as community, and splitting [2, 6] or merging [3] communities, respectively. These methods produce a hierarchy of nested communities. By merging or splitting communities one can build a hierarchical tree of community partitions called *dendrogram*. Several researcher aim to find the entire hierarchical community dendrogram [2, 6], while others try to identify only the optimal community partition [1]. More recent approaches aim to identify the community surrounding one or more seed nodes [7]. Some researchers aim to discover distinct (non-overlapping) communities, while others allow for overlaps [5].

In this paper we compare the performance of four different com-

munity detection algorithms found in the literature. These are *Newman's* algorithm [2] that follows a divisive-agglomerative approach, *CiBC* [3], an algorithm which starts my merging individual nodes into larger communities, *Bridge Bounding* [7], a local community detection algorithm that exploits edges with low clustering coefficient to identify communities bridges (inter-community edges), and *Fortunato's* [4] another local algorithm whose operation is based on the evaluation of a fitness function. The ability of the algorithms for unveil the entire community structure of networks is exploited. The performance of these algorithms is compared on a number of benchmark graphs with known community structure using the well established *modularity metric* [2].

2. COMMUNITY DETECTION

2.1 Newman's algorithm

One of the well known community finding algorithms was developed by Girvan an Newman [2, 6]. This algorithm follows, what is known as, the divisive-agglomerative method, a hierarchical approach based on which communities are detected by removing edges iteratively from the graph. An edge that belongs to many shortest paths between nodes has high *betweeness* and has to be removed, because it is more likely to be an inter-community edge. By removing gradually edges, the graph is split and its hierarchical community structure is revealed. The algorithm is computationally intensive, because following the removal of an edge, the shortest paths between all pairs of nodes have to be recalculated. However, it reveals not only individual communities, but the entire hierarchical community dendrogram of the graph. An important element of the algorithm is the *modularity* calculation, which is used to evaluate the quality of a community partition resulting and also as a termination criterion for the algorithm.

Although there is a wide range of betweenness measures available, shortest-path betweenness is used due to the lower computational cost and the satisfactory results. Shortest-path betweenness can be calculated by finding the shortest paths between all pairs of vertices and summing up how many of those run along each edge. Experimental results for implementations of the algorithm based on different betweenness metrics have shown no significant impact on the quality of the community structure output.

The general form of Newman's algorithm is as follows:

1. Calculate betweenness scores for all edges of the network.
2. Find the edge with the highest betweenness value and remove it from the network.
3. Recalculate betweenness for all remaining edges.
4. Repeat from step (2).

The output of this process can be represented as a dendrogram depicting the successive splits of the network. The detection process can be stopped at any point the output community structure is judged to be satisfactory. An accurate way to determine if a network division is satisfactory is to use an appropriate evaluation metric. A metric that can carry out this task is the *modularity* metric. Modularity essentially indicates the extent to which a given community partition is characterized by high number of intra-community edges compared to inter-community ones. Essentially, the algorithm proceeds as long as network partitions with higher modularity are produced after edge removals. An appropriate modularity threshold is applied in order to identity the optimal community structure and the algorithm to terminate.

The key in order to achieve acceptable results is the recalculation step. After the removal of an interconnecting edge, the workload for the remaining edges standing between two communities is increased. The fewer are the remaining edges, the more dramatic becomes the increase. As a consequence a single betweenness calculation followed by the serial removal of edges in descending betweenness order could lead to the faulty removal of an edge and thus poor results for the algorithm.

Summing up, despite being a fairly successful and robust algorithm, Newman is computationally intensive and thus fails to keep up with the perpetual evolution in the field of community detection. This becomes more obvious when we have to deal with large datasets or streaming data produced by modern web systems.

2.2 CiBC

Compared to other community detection algorithms applied in various scientific fields, CiBC was designed to fulfil the very specific task of identifying Web communities from a web server content, in order to improve the performance of CDNs.

The algorithm is based on a slightly different definition from what is traditionally considered as a network community. Usually, a community is defined as a subgraph for which each node has more edges to nodes of the same community than to nodes outside the community (strong community). A more flexible definition, called generalized community, is the following: a community is a subgraph in which the sum of all node degrees within the community is larger than the sum of all node degrees towards the rest of the graph. Apart from the capability of identifying overlapping communities, CiBC has one more innovative characteristic, its hybrid nature, using both local and global graph's properties in order to accomplish its mission. Community detection is performed in three phases, with each phase including further steps.

In the first phase, the *Betweeness Centrality* (BC) is calculated for each node of the graph as shown in [3]. BC is a metric used to measure how "central" a node is in the graph. Last step before proceeding with phase two includes the sorting of the nodes of the graph by ascending BC value.

The second phase concentrates on the initialization of the cliques. This is achieved using an iterative procedure starting with the nodes with the lowest BC values. Although a high BC value may indicate that a node is central within a community, it can also be an indication of a node that is central within the graph, connecting different communities. Moreover, if we start with a node characterized by a high BC value, it is highly possible to end up with a single community that includes all nodes of the graph. In each iteration, if the currently-selected node v is not assigned to any group yet, a new subset of the graph called clique is created. In this clique, we include all nodes that belong to the neighborhood of v. Moreover, we further expand this clique using Bounded-BFS with typical depth value $\sqrt{(N)}$ (where N is the number of nodes). By applying this procedure, after the completion of all iterations, a set of groups (cliques) will be created. This fatefully leads to phase three of the algorithm.

The large number of the created groups raises the need for some sort of minimization, in order to get the desired generalized community structure. This task is carried out by merging these groups through an iterative process. We define an $l \times l$ matrix B, where l refers to the number of the groups produced at phase two. Each element $B[i,j]$, with $i \neq j$ of the matrix corresponds to the number of edges that connect directly nodes assigned in group i to nodes assigned in group j. On the other hand, each element $B[i,j]$ with $i = j$ corresponds to the number of edges internal in group i. In each iteration, the pair of groups with maximum $\frac{B[i,j]}{B[i,i]}$ value is selected for merging and then the recalculation and repopulation of matrix B is required. The process terminates when there is no pair of groups with $\frac{B[i,j]}{B[i,i]} \geq 1$.

2.3 Bridge Bounding

The authors of [7] introduce a local methodology for community detection, named *Bridge Bounding*. The algorithm initiates the community detection from a certain seed node and progressively expands the community trying to identify *bridges*, i.e. edges that act as community boundaries. The *edge clustering coefficient* is calculated for each edge, looking at the edge's neighborhood, and edges are characterized as bridges depending on whether their clustering coefficient exceeds a threshold. The method is local, has low complexity and allows the flexibility to detect individual communities. Additionally, the entire community structure of a network can be uncovered starting the algorithm at various unassigned seed nodes, till all nodes have been assigned to a community.

In order to identify a community around a seed node s the algorithm uses a flooding technique. Starting at node s, nodes in the neighborhood of s are gradually attached to the community if the following two conditions are satisfied: neighbor v does not belong to any other community and the edge connecting s to v is not a bridge (community boundary). The term bridge defines an edge connecting two nodes that are members of different communities. The steps described above are repeated for every node until no other node can be attached to the community. Repeating the same procedure for different nodes, inevitably leads to the discovery of the overall community structure of the graph.

A critical task for the algorithm is the definition of function B_L, which maps each edge to a real number in the interval $[0, 1]$, quantifying the extent to which an edge acts as a "bridge". More precisely, the value B_L for edge (s, v) is defined as: $B_L(s, v) = 1 - \frac{N(s) \cap N(v)}{min(d(s)-1, d(v)-1)}$, where $N(s)$ denotes the neighborhood for node s and $d(s)$ its degree. A low B_L value indicates an edge connecting nodes of the same community. Conversely, a high B_L value indicates an edge that acts as a bridge connecting two different communities.

During the flooding process, the algorithm can decide which edges are community bridges by comparing the result of the function with a predefined threshold. In order to assign a value to the threshold, we first have to examine the distribution of B_L values all graph edges. In that way, we introduce a global step with low computation cost. Assuming that similar networks have similar B_L values distribution the threshold value can be pre-defined. This assumption can be proven experimentally.

2.4 Fortunato's algorithm

An interesting method for community detection appears in [5]. This algorithm is developed based on the observation that network communities may have overlaps, and, thus, algorithms should allow for the identification of overlapping communities. Based on this principle, a local algorithm is devised developing a community from a seed node and expanding around it. A community is identified as a subgraph that has a certain *fitness*. The authors provide an appropriate fitness function, whose calculation is based on the number of inter- and intra-community edges and a tunable parameter a. Starting at a node, at each iteration, the community is either expanded by a neighboring node that increases the community fitness, or shrinks by omitting a node if this action results in higher fitness for the community. The algorithm stops when the insertion of any neighboring node would lower the fitness of the community. This algorithm is local, and able to identify individual communities. The entire overlapping and hierarchical structure of complex networks can also be found.

For a community $\mathcal{G}$ of the graph, the fitness $f_{\mathcal{G}}$ is calculated as follows:

$$f_{\mathcal{G}} = \frac{K_{in}^{\mathcal{G}}}{(K_{in}^{\mathcal{G}} + K_{out}^{\mathcal{G}})^a} \quad (1)$$

where $K_{in}^{\mathcal{G}}$ and $K_{out}^{\mathcal{G}}$ refer to the total internal and external degrees of community $\mathcal{G}$ respectively, and a is a positive real-valued parameter which controls the size of the community.

As mentioned above, in order to reveal the entire community structure of a network, each node should belong to at least one community. To achieve this goal we apply the process summarized below:

1. Select at random node A.
2. Discover the natural community of node A.
3. Randomly select a node B that has not been assigned to a community.
4. Discover the natural community of B, by exploring all the candidate nodes regardless of whether they belong to other communities (allow for overlaps).
5. Repeat from step (3).

A major advantage of the above approach is the lower computational cost. In summary, Fortunato's algorithm is a modern and highly successful algorithm as it is revealed by its experimental results.

3. EXPERIMENTAL EVALUATION

In this section we describe the experimental framework, namely the way benchmark graphs were created, the modularity metric used for the comparison of the algorithms, and finally the a comparative analysis of the algorithms' performance. We have created a variety of benchmark graphs with known community structure to test the accuracy of our algorithm. Benchmark graphs are essential in the testing of a community detection algorithm, since there is an apriori knowledge of the structure of the graph and thus one is able to accurately ascertain the accuracy of the algorithm. Our benchmark graphs were generated randomly given the following set of parameters: number of graph nodes N, number of communities $Comm$, node degree $degree$, and finally ratio of intra-community edges to node degree $local/degree$. The parameters used for the creation of the benchmark graphs and their corresponding values are shown in Table 1.

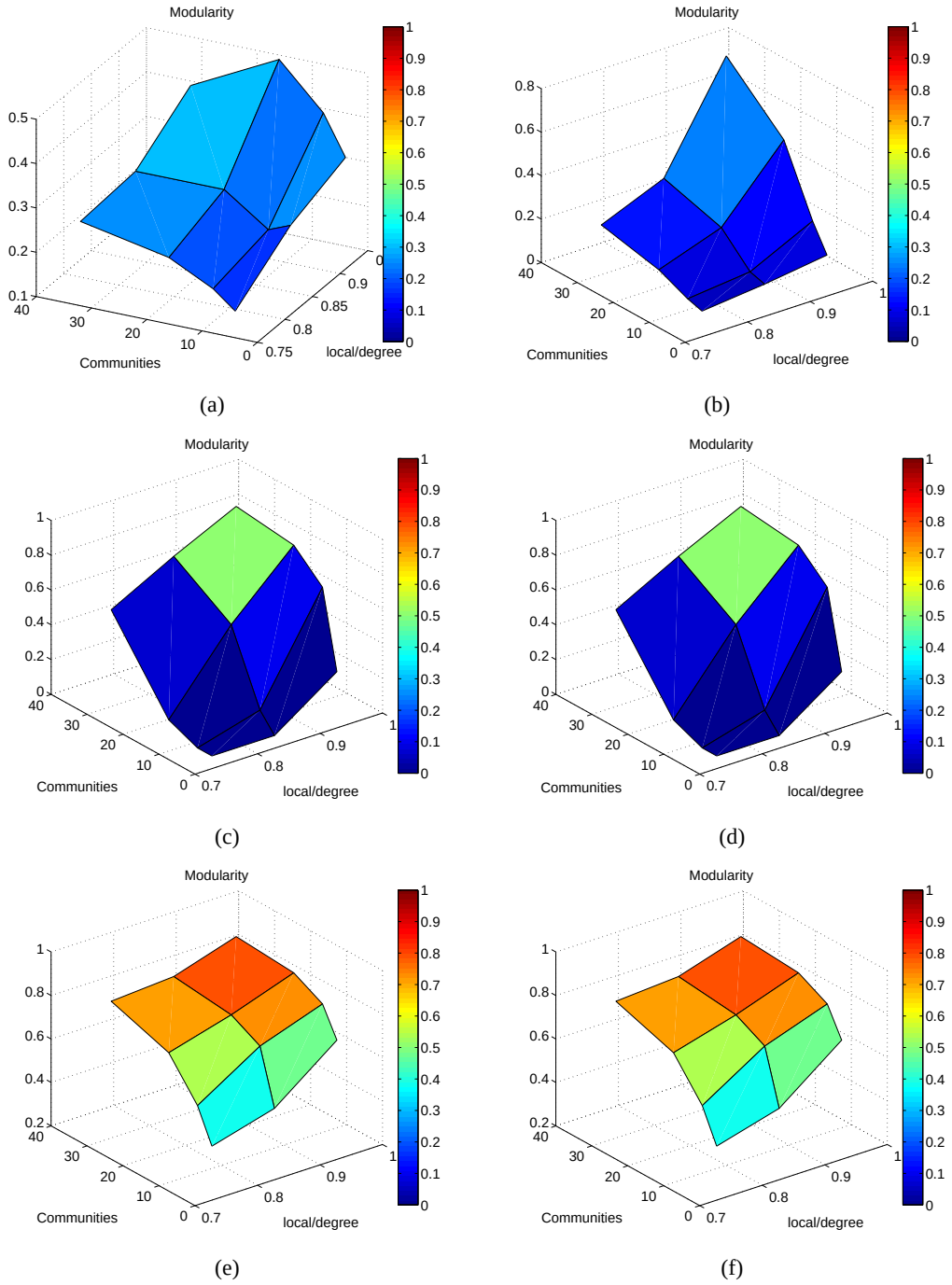

Figure 1: (a) Newman's algorithm (b) Bridge bounding algorithm. (c,d) CiBC algorithm with (c) depth 2 BFS and (d) without BFS. (e,f) Fortunato's algorithm with (e) $a = 0.6$ **and (f)** $a = 0.8$**.**

N	512, 1024
$Comm$	4, 8, 16, 32
$Degree$	10, 20, 30
$local/degree$	0.75, 0.85, 0.95

Table 1: Parameters used for the benchmark graphs.

In order to compare the performance of the algorithms, we use the well established *modularity metric*. Modularity essentially indicates the extent to which a given community partition is characterized by high number of intra-community edges compared to inter-community ones and is calculated as follows: for a network community structure with l communities, an $l \times l$ symmetric matrix e is defined whose element e_{ij} is the fraction of all edges in the network that connect nodes in community i to nodes in community j. The row sums $a_i = \sum_j e_{ij}$ of this matrix represent the fraction of edges with an endpoint in community i. Modularity Q of a community partition is defined as follows: [2, 6]:$Q = \sum_{i=1}^{l} e_{ii} - a_i^2$. Modularity measures the fraction of intra-community edges minus the expected value of the same quantity in a network with the same community division and random connections between nodes. If the number of intra-community edges is no better than random, we will get a modularity value close to 0, while modularity values approaching 1 (which is the maximum possible) indicate networks with strong community structure.

Fig. 1 show the performance of the four algorithms on the benchmark graphs we created with respect to modularity. Modularity is plotted against two parameters that were shown to play the most important role in the performance of the algorithms. For all algorithms these two parameters are the number of communities in the graph and the local/degree which is an indication of the density of the communities. The experimental results on the given benchmark graphs demonstrate that although Newman is a computationally intensive community finding algorithm, it is not very effective in identifying communities in the given graphs, and starts to be effective only when community density becomes very high (Fig. 1(a)).

Similarly, Bridge Bounding is mostly effective for graphs with very dense communities, where $\frac{local}{degree}$ approaches 0.9 and higher (Fig. 1(b)). However, Bridge Bounding (BB) is a local algorithm and requires very little time. We can thus conclude that BB can be safely used on graphs with very dense communities. CiBC seems to perform better than the previous two algorithms. When cliques are not expanded with BFS , CiBC seems to give decent results when community density exceeds 0.8 ($\frac{local}{degree} > 0.8$), as shown in Fig. 1(c). However, comparing CiBC to Bridge Bounding, we need to emphasize that CiBC is not a local algorithm as Bridge Bounding is, and its operation requires a global view of the entire graph. However, starting with individual nodes and merging them into larger communities, renders it far less computationally intensive compared to Newman which uses the inverse approach, namely, starts with the entire graph and splits it into communities by gradually removing edges. Using community merging, CiBC never reaches the point to manipulate the entire graph as a whole.

Figs. 1(e) and 1(f) demonstrate FortunatoŠs performance for two different values of a, namely $a = 0.6$ and $a = 1.0$. In both cases Fortunato outperforms its previous counterparts demonstrating its ability to identity communities even in graphs with community density $\frac{local}{degree} > 0.7$. Furthermore, Fortunato is a local algorithm a little more computationally efficient compared to Bridge Bounding. The advantage of Fortunato is that the fitness function it utilized to expands a community from a seed node is directly related to the density of the community. Thus, Fortunato is the best of all candidates and could be safely used to identify communities in various application domains. The fact that it is a local algorithm makes it easy to apply in situations where a global view of the network is not easy to obtain. Finally, we have to mention that FortunatoŠs algorithm allows for further improvement.

4. CONCLUSIONS

We compared the performance of four community detection algorithms, each one following a different approach. The performance of the algorithms was demonstrated on a variety of benchmark graphs with known community structure. In general, based on the above observations we can derive the conclusion that although community finding is a problems that appears in many different version and exhibits a richness of solutions, there is still plenty of room for improvement of existing solutions and for the derivation of new ones that would allow the manipulation of graphs from various new and emerging application domains like social networks and other types of collaborative environments. There is an emerging need to devise community detection algorithms for dynamic graphs, i.e. graphs whose structure evolved over time. Such algorithms would be able to capture for example the dynamic evolution of social networks.

5. REFERENCES

[1] G. W. Flake, S. Lawrence, C. L. Giles, and F. M. Coetzee. Self-organization and identification of web communities. *IEEE Computer*, 35:66–71, March 2002.

[2] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12):7821–7826, June 2002.

[3] D. Katsaros, G. Pallis, K. Stamos, A. Vakali, A. Sidiropoulos, and Y. Manolopoulos. Cdns content outsourcing via generalized communities. *IEEE Transactions on Knowledge and Data Engineering*, 21:137–151, 2009.

[4] A. Lancichinetti and S. Fortunato. Community detection algorithms: a comparative analysis. *Physical Review E*, 80(5 Pt 2):056117, Sep 2009.

[5] A. Lancichinetti, S. Fortunato, and J. Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3):033015+, March 2009.

[6] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113, Feb 2004.

[7] S. Papadopoulos, A. Skusa, A. Vakali, Y. Kompatsiaris, and N. Wagner. Bridge bounding: A local approach for efficient community discovery in complex networks. Technical Report arXiv:0902.0871, Feb 2009.