A Distributed Algorithm for Community Detection in Large Graphs

Harris Papadakis*, Costas Panagiotakis† and Paraskevi Fragopoulou*
*Department of Applied Informatics and Multimedia
†Department of Commerce and Marketing,
Technological Educational Institute of Crete, Heraklion, Crete, Greece
Email:adanar@epp.teicrete.gr, cpanag@staff.teicrete.gr, fragopou@ics.forth.gr

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. Several attempts have been made to provide a formal definition to the generally described "community finding" concept, providing different approaches. Some algorithms follow an iterative approach starting by characterizing either the entire network, or each individual node as community, and splitting [1] or merging communities respectively, producing a hierarchical tree of nested communities, called *dendrogram*. Several researchers aim to find the entire hierarchical community dendrogram [1] while others wish to identify only the optimal community partition. Some researchers aim at discovering distinct (non-overlapping) communities, while others allow for overlaps [2]. The Blondel algorithm described by Blondel et al. in [3], follows a bottom-up approach. Each node in the graph comprises a singleton community. Two communities are merged into one if the resulting community has larger modularity value that both the initial ones. This is a fast and accurate algorithm which, detects all communities in the graph. In suffers however in the sense that it constantly, during its execution, requires the knowledge of a global information of the graph, namely the number of its edges (which change as the algorithm modifies the graph), limiting its distributed nature. The Infomap algorithm [4] transforms the problem of community detection into efficiently compressing the structure of the graph, so that one can recovered almost the entire structure from the compressed form. This is achieved by minimizing a function that expresses the tradeoff between compression factor and loss of information (difference between the original graph and the reconstructed graph).

In this paper we propose an algorithm to identify the entire community structure of a network based on interactions between neighboring nodes. In the core of our proposal lies the spring metaphor which also inspired the Vivaldi synthetic network coordinate algorithm [5]. The algorithm comprises two main phases. First, each node selects a "local" set (supposedly) containing nodes of the same community, and a "foreign" set containing nodes of different communities. As the algorithm evolves, and the springs connecting local and foreign nodes are tightened and relaxed, nodes of the same community pull each other close together, while nodes of different communities push each other further away. In the second phase of the algorithm, a hierarchical clustering algorithm on points in space is used to automatically identify the natural communities formed in space. Extensive experiments on benchmark graphs with known community structure indicate that our algorithm is highly accurate in identifying community membership of nodes.

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

I. Local community finding

The proposed local community finding algorithm comprises the following steps: a) The position estimation algorithm, which is a distributed algorithm inspired by Vivaldi [5], and b) The community detection algorithm using hierarchical clustering. In the core of our proposal lies the spring metaphor which inspired the Vivaldi algorithm. As we mentioned, Vivaldi uses the spring relaxation metaphor to position the nodes in a virtual space (the n-dimensional Euclidean space $\Re^n$), so as the Euclidean distance of any two node positions approximates the actual distance between those nodes. In the original application of Vivaldi, the actual distances were the latencies between Internet hosts. Our algorithm is based on the idea that by providing our algorithm with our own, appropriate, definition of distance between nodes, we can use Vivaldi to position the nodes in a manner as to reflect community membership, i.e. nodes in the same community will be placed closer in space than nodes of different communities. In other words nodes belonging to the same community will form natural clusters in space. Thus, we define the actual distance of two nodes in the same community as 0, and for nodes in different communities as 100.

Given this definition of distance, we can employ the core part of the Vivaldi algorithm to position the nodes appropriately in $\Re^n$. As one can expect from those dual distances, Vivaldi will position nodes in the same community close-by in space, while place nodes of different communities away from each other. In addition, Vivaldi requires a selection of nodes to probe. Each node calculates a "local" set containing (presumably) nodes of the same community, and a "foreign" set containing nodes of different communities. The size of the local set as well as the size of the foreign set of a node equals the degree of the node. The perfect construction of these sets depends on the apriori knowledge of node community membership, which is the actual problem we are trying to solve. However, even though we do not know the community each node belongs to, there are two facts we can exploit to make Vivaldi work without this knowledge:

The first is the fact that the number of *intra-community* *links* exceeds the number of *inter-community links* for each node. This means that, if we consider all of a node's neighbors belonging to the same community, this assumption will be, *mostly*, correct, which in turn means that even though some times the node may move to the wrong direction, most of the time it will move to the right direction and thus, will eventually acquire an appropriate position in space. The second fact we exploit concerns the *foreign links*. Since we consider all a node's links as local links, we need to find some nodes which most likely do not belong to the same community as that node, and therefor will be considered as foreign nodes. This can simply be done by randomly selecting a small number of nodes from the entire graph. Assuming that the number of communities in the graph is at least three, the majority of the nodes in this set will belong to a different community than the node itself. These nodes will comprise the "foreign set".

Iteratively, each node randomly selects a node from either the local or the foreign set. It then uses Vivaldi to update its current position using that node and the appropriate distance (i.e: 0 or 100). Each node continues this process until it deems its position to have stabilized as much as possible. This is done by calculating the distance of the two positions of the node between 100 position updates. Should this value be less than a certain threshold ($= 20$) for 20 consecutive times, the node declared itself to have stabilized. It still continues, however, to execute the algorithm until at least 90% of its "foreign" and "local" sets have also stabilized.

As we mentioned before, both the "local" and the "foreign" sets of a node will contain erroneous nodes. One of the most important augments of our algorithm is its ability to dynamically correct both those sets. This is based on the assumption that, as the algorithm progresses, those nodes in the "local" set of a node which do not actually belong in the same community as that node, will still (after several position updates) be located a long distance away from that node, whereas the nodes in the same community will gravitate towards another much faster. As a result, even though the distances between a node and the nodes in its "local" will initially be uniformly distributed, after a while we will notice the distances of the nodes to be divided into two groups of smaller and larger values. This is a good indication that we can separate the wheat from the chaff. This is implemented in the following fashion: Every 50 position updates each node checks its "confidence level" in identifying the erroneous nodes. This is done by calculating a modified normalized standard deviation of the distances to the nodes in its "local" set, where instead of the mean, each distance is subtracted from the median value of the node distances. Should this value exceed a certain threshold ($= 0.6$) , the node iterates between the nodes in both its "local" and "foreign" sets, removing any inappropriate nodes. In order to identify a node as "local" or "foreign" based on its distance, another threshold is required. This threshold is calculated as $minDistance + max(\frac{maxDistance - minDistance}{3}, \frac{100}{3})$, where $minDistance$ and $maxDistance$ are the minimum and maximum values of the distances to all the nodes in the "local" set.

After each node has converged to a point in space, we use a hierarchical clustering algorithm to perform the actual grouping of points-nodes into clusters-communities. The main advantage of the hierarchical clustering algorithms is that the number of clusters need not be specified a priory, and problems due to initialization and local minima do not arise [6]. The following procedure is executed: First, each node is considered as a (singleton) cluster. In addition, to make the procedure completely distributed, each node-cluster is aware of the location only of its neighboring node-clusters. Then, the following loop is executed repeatedly, until no appropriate pair of clusters can be located: Given a pair of neighboring clusters A and B, if A is B's closest neighbor and likewise B is A's closest neighbor, then those two clusters are merged in the following fashion: the merged cluster contains the union of the neighbors of A and B, its position is calculated as the weighted (based on the population of nodes in each cluster A and B) average of the positions of A and B.

II. Experimental results

We have created a variety of benchmark graphs with known community structure to test the accuracy of our algorithm. Our benchmark graphs were generated randomly given the following set of parameters: the number of nodes N of the graph (1000, 5000, 10000), the number of communities $Comm$ of the graph (5, 10, 20, 40, 80), the ratio of local links to node degree $local/degree$ (0.55, 0.65, 0.75, 0.85), and the (average) degree of nodes $degree$ (10, 20, 30, 40). Even though the number of the nodes, the number of the communities and the degree of the nodes are parameters of the construction of the graph, the degree of each node as well as the number of nodes in a single community varies based on a pareto distribution. This enables us to create graphs of community sizes and individual degrees varying up to an order of magnitude. In total, we created a number of 208 graphs.

We compared the five algorithms using two accuracy-related metrics found mostly used in the literature, to compare the calculated communities with the correct ones. The first is a simple accuracy metric (number of nodes in the intersection of the two sets divided by the number of nodes in their union). We also used the Normalized Mutual Information (NMI) metric to evaluate the correctness of the detected communities [7]. In both cases, a value close to one indicates correct detection of the communities of the graph.

Fig. 1, 2, 3 show the performance of a single algorithm using the accuracy metric, on all 208 graphs. Since there are four types of parameters which describe the graphs of the experiments, we decided to used the two most important factors which affect the algorithms' performances, in order to plot the accuracies in 3D graphs. The first of these factors is always the local links to node degree value, which dictates how strong the communities in the graph are. The second factor in some cases is the number of communities in the graph, while in other cases is the (average) density of these communities. Thus, we plot, for each algorithm and accuracy metric, two 3D graphs using, in one case, the number of communities as the values on one the axes and the average density on the other. Each accuracy value in the graphs is the average of the accuracies of all the experiments based on the benchmark graphs with the same value on the aforementioned factors.

This research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong

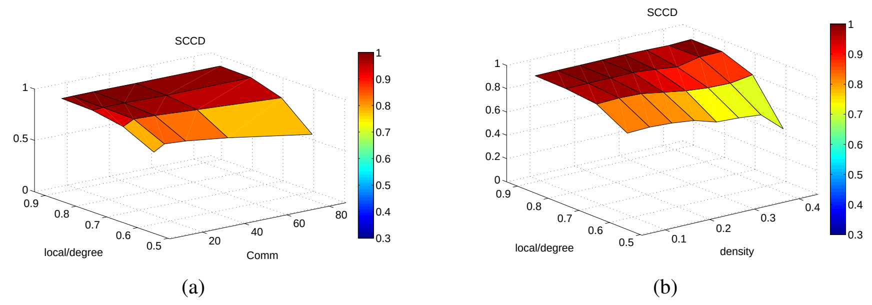

Fig. 1: **(a),(b)** The mean value of accuracy under **(a)** different ratios of total degree to local links ($local/degree$) and number of communities ($Comm$) and **(b)** total degree to local links ($local/degree$) and densities for our algorithm.

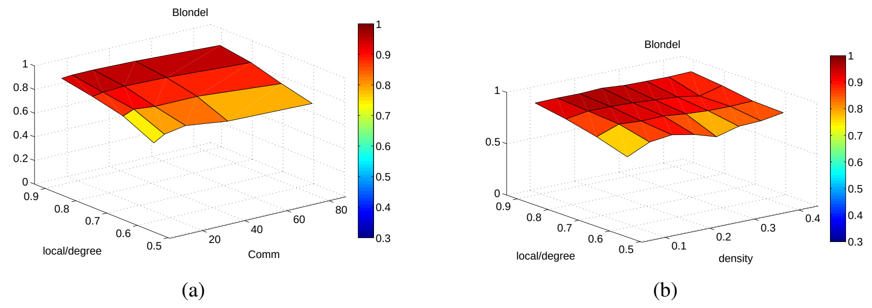

Fig. 2: **(a),(b)** The mean value of accuracy under **(a)** different ratios of total degree to local links ($local/degree$) and number of communities ($Comm$) and **(b)** total degree to local links ($local/degree$) and densities for the Blondel algorithm.

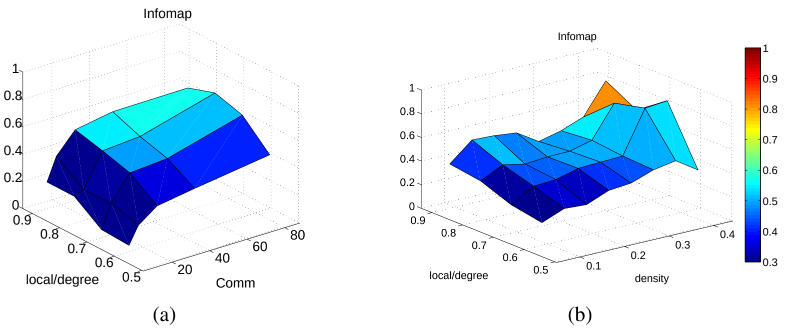

Fig. 3: **(a),(b)** The mean value of accuracy under **(a)** different ratios of total degree to local links ($local/degree$) and number of communities ($Comm$) and **(b)** total degree to local links ($local/degree$) and densities for the Infomap algorithm.

Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: ARCHIMEDE III-TEI-Crete-P2PCOORD.

References

[1] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proc of the National Academy of Sciences of the United States of America*, vol. 99, no. 12, pp. 7821–7826, June 2002.

[2] A. Lancichinetti, S. Fortunato, and J. Kertész, "Detecting the overlapping and hierarchical community structure in complex networks," *New J of Physics*, vol. 11, no. 3, pp. 033 015+, March 2009.

[3] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, 2008.

[4] M. Rosvall and C. T. Bergstrom, "An information-theoretic framework for resolving community structure in complex networks," *Proc of the National Academy of Sciences*, vol. 104, no. 18, pp. 7327–7331, 2007.

[5] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *Proc of the ACM SIGCOMM '04 Conference*, August 2004.

[6] H. Frigui and R. Krishnapuram, "A robust competitive clustering algorithm with applications in computer vision," *IEEE Trans on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 450–465, 1999.

[7] A. Strehl, J. Ghosh, and C. Cardie, "Cluster ensembles - a knowledge reuse framework for combining multiple partitions," *J of Machine Learning Research*, vol. 3, pp. 583–617, 2002.