An Ultra-Fast Modularity-Based Graph Clustering Algorithm

Leonardo Jesus Almeida and Alneu de Andrade Lopes

Institute of Mathematics and Computer Science - ICMC
University of São Paulo/Campus of São Carlos - (USP São Carlos)
P.O. Box 668, São Carlos, SP, Brazil

Abstract. In this paper, we propose a multilevel graph partitioning scheme to speed up a modularity-based graph clustering technique. The modularity-based algorithm was proposed by Newman for partitioning graphs into communities without the input of the number of clusters. The algorithm seeks to maximize a modularity measure. However, its worst-time complexity on sparse graphs is $O(n^2)$, where n is the number of vertices, which can be prohibitive for many applications. The multilevel graph partitioning scheme consists of three phases: (i) reduction of the size (coarsen) of original graph by collapsing vertices and edges, (ii) partitioning the coarsened graph, and (iii) uncoarsen it to obtain a partition for the original graph. The rationale behind this strategy is to apply a computationally expensive method in a coarsened graph, i.e., with a significantly reduced number of vertices and edges. Empirical evaluation using this approach demonstrate a significant speed up of the modularity-based algorithm, keeping a good quality clusters partitioning.

Key words: Clustering, Graph Clustering, Multilevel Graph Clustering

1 Introduction

Clustering is an important problem with many applications, and a number of different algorithms have been proposed over the past decades. Recently, compelling application domains have become available in which to concept learning requires effective handling of relational data. Social networks and web analysis, for instance, require the representation of structural relations among people or web pages. In these domains, data must represent relationship between entities rather than entity's attributes. Such representation motivates approaches to mine concepts from graph-based data.

Graphs are formed by a set of vertices and a set of edges that are connections between pairs of vertices [1, 2]. Graph clustering is the task of partitioning similar vertices into clusters taking into consideration the edge structure of the graph so that there are many edges within each cluster and few edges between the clusters [3, 4]. Unfortunately, finding an optimal partition is a problem known to be NP-complete [5], so it's necessary to use some heuristics for a practical solution.

To overcome the problem of finding the number of clusters and speed up the clustering process we apply the Newman's modularity algorithm [6] as the partitioning algorithm in a multilevel graph partitioning approach [7–9]. A multilevel graph partitioning scheme [10, 8, 9] consists of three phases: (i) reduction of the size (coarsen) of the original graph by collapsing vertices and edges; (ii) partitioning the coarsened graph; and (iii) uncoarsen it to construct a partition for the original graph . The rationale behind this strategy is to apply a computationally expensive method in a coarsened graph with significant reduction of number of vertices and edges. This approach can speed up the fast modularity algorithm, keeping a good quality cluster partitioning.

In the following section we describe some background concepts in graph clustering. In Section 3, we explain the proposed approach of joining Multilevel Graph Partition and Newman's modularity Q. In Section 4 we evaluate the approach on some datasets. Finally, in Section 5, we present the conclusions and future work.

2 Graph Clustering

The problem of partitioning a graph in k different clusters is defined as follows: Given a graph $G = (V, E)$ with $|V| = n$, finding subgraphs $V_1, V_2, ..., V_k$ such that $V_i \cap V_j = \emptyset$ for $i \neq j$, $\bigcup_i V_i = V$ and the number of edges of E that connects vertices from different clusters are minimized. This quantity is known as *edge-cut*. The clusters founded are generally represented in a vector P of length n, such that for each vertex $v \in V$,$P[v]$ is in $[1, k]$. The value of k is usually given as input for many graph clustering algorithm, although there are techniques to suggest a good value.

The above goal described can be interpreted in various ways leading different criteria for optimization. We briefly review some specific objective functions that are most related to our work:

Ratio association maximization It takes the partitioning scheme that leads to the highest intra-cluster average degree, i.e., the summation of number of edges in the cluster i, $links(V_i, V_i)$, divided by the number of vertices in i, $|V_i|$, expressed by Equation 1.

$$RAssoc(G) = \max_{V_1,...,V_k} \sum_{i=1}^{k} \frac{links(V_i, V_i)}{|V_i|}. \quad (1)$$

Ratio cut minimization The difference between Ratio cut and Ratio association is that the first seeks to minimize the *edge-cut* while the second seeks to maximize internal cluster edges weight. It is obtained with Equation 2

$$RCut(G) = \min_{V_1,...,V_k} \sum_{i=1}^{k} \frac{links(V_i, \overline{V_i})}{|V_i|}, \quad (2)$$

where $\overline{V_i}$ is the subset of vertices $V - V_i$.

Normalized cut minimization The normalized cut objective [11] seeks to minimize the cut relative to the number of edges in a cluster instead of its size. The objective is expressed by Equation 3

$$NCut(G) = \min_{V_1,\ldots,V_k} \sum_{i=1}^{k} \frac{links(V_i, \overline{V_i})}{degree(V_i)}, \tag{3}$$

where $degree(V_i) = |\{(u, v) \in E | u, v \in V_i\}|$.

Modularity maximization Modularity measures the fraction of the cluster's inner edges e_{ii} minus the expected value a_i if the edges are placed at random [12]. Let e_{ij} be the half fraction of edges in G that connects vertex in different clusters. So, $|E| = e_{ij} + e_{ji} + e_{ii}$. We can calculate a_i by $a_i = \sum_j e_{ij}$. The measure is given by Equation 4

$$Q = \sum_i (e_{ii} - a_i^2). \tag{4}$$

If the edges are connected at random the fraction of inner edges in a cluster i is a_i^2, which produces $Q = 0$. Any $Q \neq 0$ indicates the presence of communities structure.

In 2004, Newman proposed an agglomerative hierarchical algorithm for partitioning graphs into communities without the input of the number of clusters [6]. The algorithm seeks to maximize the modularity measure given by Eq.4. Starting with each vertex as a single community, it repeatedly joins communities together in pairs. At each step is selected the pair which promotes the highest modularity gain, that is calculated by Equation 5.

$$\Delta Q = 2(e_{ij} - a_i a_j). \tag{5}$$

Following a join, it's necessary to update the e_{ij}'s values, a task with worst-time complexity $O(n)$. At any time, there will be at most $|E|$ edges in graph, thus each step of the algorithm takes $O(|E| + n)$. The worst-time cost occurs when it's necessary to perform $n - 1$ joins, so the entire algorithm runs in time $O((|E| + n)n)$, or $O(n^2)$ in a sparse graph [6], which can be prohibitive.

In the next section, we propose a graph clustering technique based on the Newman's modularity algorithm [6] applied in a multilevel graph partitioning approach [7–9], aiming to speed up the clustering process.

3 Multilevel Graph Partitioning

Multilevel schemes [10, 7] are relatively fast and provide excellent partitions for a wide variety of graphs. Formally, a multilevel scheme works as follows: consider a weighted graph $G_0 = (V_0, E_0)$, with weights both on vertices and edges. A multilevel graph partitioning algorithm consists of the following three phases, illustrated in Figure 1.

Coarsening phase The graph G_0 is transformed into a sequence of smaller graphs $G_1, G_2, ..., G_t$ such that $|V_0| > |V_1| > |V_2| > ... > |V_t|$.

Partitioning phase A k-way partition P_t of the graph $G_t = (V_t, E_t)$ is computed that partitions V_t into k parts.

Uncoarsening phase The partition Pm of G_t is projected back to G_0 by going through intermediate partitions $P_{t-1}, P_{t-2}, ..., P_1, P_0$.

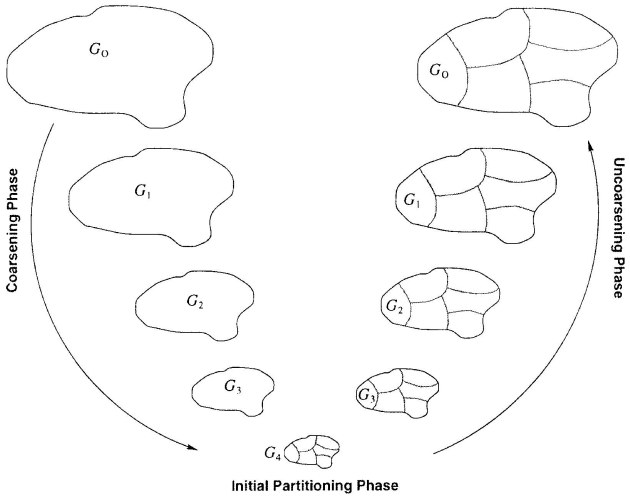


Fig. 1. Multilevel Graph Partition Scheme [8]

3.1 Coarsening phase

In the Coarsening phase, the initial graph G_0 is iteratively reduced to smaller graphs, such that the number of vertices and edges decrease. Coarsen a graph from G_i to G_{i+1}, set of vertices in G_i are combined to form supervertices in G_{i+1}. In order to preserve the connectivity in the smaller graph, the edges of supervertex are taken to be the union of the edges of corresponding vertices in the previous graph. In the case where the vertices which form the supervertex have common adjacent vertices, the edge between the supervertex and these vertices will be the union of the old edges.

The subset of vertices pairs chosen to be collapsed can be formally defined in terms of matchings. A matching of a graph is a set of edges, no two of which are incident on the same vertex [7]. In the following we describe three criteria for selecting the maximal matching. We notice that the complexity of the coarsening phase, $O(|E|)$, where $|E|$ is again the number of edges in the graph, is

asymptotically similar to three methods, although the proportionality constant of Modified heavy-edge matching is greater [8].

Random matching (RM) The random matching generates the maximal matching using a randomized algorithm. It works as follows. Initially, all vertices in the graph are marked as unmatched. Then, the vertices are randomly visited until all vertices are visited or the graph size desired reduction is achieved. If an unmatched vertex v is selected for matching, we then seek randomly one of its neighbors that has not been matched yet. If a vertex u exists, we mark both vertices as matched and include the edge (v, u) in matching.

Heavy-edge matching (HEM) The idea behind heavy-edge matching is to reduce the edge-weight of the coarser graph by selecting a maximal matching whose edges have large weight. A heavy-edge matching is, using a randomized algorithm, similar to random matching. Let v be the selected vertex, and H the subset of unmatched v's neighbors. Heavy-edge matching selects for matching the vertex $u \in H$, such that the edge (v, u) has greater weight.

Modified heavy-edge matching (MHEM) The Modified heavy-edge matching is closely similar to heavy-edge matching. The difference occurs when there are more than one maximal weight adjacent vertices. Let v be the selected vertex, and H the subset of unmatched v's neighbors with maximal edge weight, and W_{v-u} the sum of the weights of the edges of u that connect u to vertices adjacent to v. Modified Heavy-edge matching selects for matching the vertex $u \in H$, such that W_{v-u} is maximized.

3.2 Partitioning phase

The partitioning phase of a multilevel scheme computes the partition P_t of the coarser graph G_t. Various algorithms can be used to obtain the partition such as iterative, hierarchical, divisive or agglomerative [4]. Since the size of the coarser graph G_t is significantly smaller then the original one (depending on coarsening stop criterion, $|V_t| < 100$), this step will spend a small amount of time [7].

Our implementation uses the Fast Modularity algorithm in this phase. The algorithm is an agglomerative hierarchical method that seeks a greedy optimization of modularity Q measure. As one feature, the method calculates the modularity value along each step and it selects the partition of the step which had the greater modularity value.

3.3 Uncoarsening phase

During the uncoarsening phase, the partition found out in partitioning phase (P_t) is projected back through each level until the original graph is achieved. Since each supervertex of G_{i+1} should contain two or more distinct vertices of G_i, we can obtain P_i by simply assigning to the collapse vertices the same partition of supervertex in P_{i+1}. Even though P_{i+1} is a local minima partition of G_{i+1}, the projected partition P_i may not be at a local minima with respect

to G_i . Since G_i is less coarse, it has more degrees of freedom that can be used to improve P_i , and decrease the edge-cut. Hence, it may still be possible to improve the projected partition of G_{i-1} by local refinement heuristics [7].

4 Experiments

We have applied our algorithm on two scientific graphs: *Netscience* and *CBR-ILP-IR*. The *Netscience* is a network of coauthorships between scientists [13]. The graph has a total of 1,589 vertices in it, representing scientists from a broad variety of fields. As the graph has more than one component, only the 379 vertices and 914 edges falling in the largest connected were used. The *CBR-ILP-IR* is a network of similarity between articles from three different topics: Case Based Reasoning, Inductive Logic Programming and Information Retrieval. The graph has a total of 574 vertices that represents documents, and the 19,213 edges represent the similarity between the documents calculated using cosine similarity measure.

We evaluated the proposed algorithm partitioning quality considering the objective functions described in Section 2. We also considered the algorithm runtime in milliseconds. The three matching schemes described in Section 3 were implemented and used in Coarsening Phase for the two graphs. In Partitioning Phase we used our Fast Modularity algorithm [6] implementation. We decided using no refinement approach, so in the Uncoarsening Phase we simple assign to inner vertices the same partition of Supervertex. The decision was taken because we aimed to verify the effectiveness of using the Fast Modularity Algorithm on the partitioning Phase. Adding a refinement algorithm would certainly improve the quality performance, but it could hide an bad initial partition created by the Fast Modularity. As the three approaches are based on random choices, we repeated 100 times each run and we considered the measures average. The experiments were perfomed on an Intel Xeon 2 GHz processor with 4GBytes of RAM memory.

In Table 1 is presented the results of our experiments with *Netscience* and *CBR-ILP-IR* data sets. For the both graphs, HEM and MHEM lead to partitions whose quality measures are better than that ones produced by RM. HEM and MHEM have similar overall quality. Nevertheless, all the three matching schemes have worse quality results than that obtained by applying the Fast Modularity algorithm on entire *CBR-ILP-IR* graph. The *CBR-ILP-IR* graph is denser than *Netscience*, and for many edges, the weights values are similar. Thus, as the matchings are performed at random, it could make unsuitable joins.

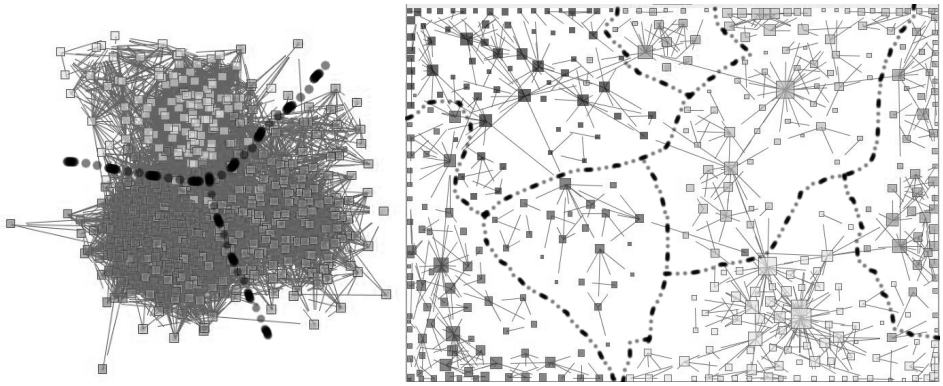
The variation of modularity Q is minimal, but it had a little increase when it was used multilivel partitioning HEM or MHEM on *NetScience*. Otherwise, it decrease a little on *CBR-ILP-IR*. On *Netscience*, the coarsening phase produced reduced graphs for partitioning better than on *CBR-ILP-IR*, because of its sparsity. So, as the number of edges is smaller in reduced graph, the fractions e_{ii} and e_{ij} are more meaningful for Fast Modularity. For both graph, the RM schemes

produced low modularity values, which can be explain because RM tends do produce graphs with larger edge-cut than HEM and MHEM.

In Figure 2 it's shown a visualization of *Netscience* and *CBR-ILP-IR* graphs. The regions limited by dashed lines represent the partitions founded by our algorithm using HEM in coarsening phase, and Fast Community as partitioning algorithm. In spite of the high number of edges, in Figure 2(a) we can see the three different groups representing the three main areas CBR, ILP and IR. The *Netscience* graph, Figure 2(b), has less edges, so it is clearer to we see each of seven partitions founded by the our algorithm. Again the lines separate the partitions, and the shape's size indicates the vertex degree, which means how many papers an author published with coauthors.

Table 1. Objectives functions values and elapsed algorithm runtime for *CBR-ILP-IR* and *Netscience* data sets

CBR-ILP-IR	None	RM	HEM	MHEM	*Netscience*	None	RM	HEM	MHEM
RAssoc(G)	**161.37**	84.57	120.34	122.81		**74.83**	30.86	34.62	35.35
RCut(G)	**40.11**	97.12	49.13	53.42		5.64	2.69	**1.84**	1.87
NCut(G)	**0.63**	1.44	0.69	0.76		1.19	0.58	**0.38**	0.39
Q	**0.388**	0.194	0.352	0.345		0.462	0.463	**0.478**	0.477
$Time(ms)$	68514	**24345**	26276	26841		22670	6422	**5779**	5895



(a) *CBR-ILP-IR* (b) *Netscience*

Fig. 2. Graph representing CBR-ILP-IR and Netscience data sets rendered by Prefuse [14]. The colors represent the clustering found by the algorithm using HEM in Coarsening Phase and Fast Modularity in Partitioning Phase.

We ran another experiment comparing Fast Modularity and HEM on two other networks. The *cond-2003* is a network of coauthorships between scientists posting preprints on the Condensed Matter E-Print Archive between 1995 and 2003, and the *cond-2005* which is an extension including postings until 2005 [15]. Both networks are available on Newman's network datasets [1]. The *cond-2003* network has 31,162 vertices and 116,181 edges, while *cond-2005* has 40,420 vertices and 171,734 edges.

In Table 2 is presented the results of second experiment. For *cond-2003* network the modularity values produced by both algorithms are close, besides HEM were a little better on *cond-2005*. For both networks, HEM runtime is smaller than Fast Modularty. As our aim was to analise how runtime grows along with the network size, we also plotted a graphic comparison between both algorithms runtime as is shown in Figure 3. The points in each graphics are related to the number of vertices (edges) in *netscience, CBR-ILP-IR,cond-2003* and *cond-2005*, respectively. Taking into account only the number of vertices in the network the Fast Modularity's runtime grows faster than HEM, Figure 3(a). The same occurred when the runtimes was plotted against the number of edges, as it is presented in Figure 3(b). However the HEM runtime grow curve tends to fit a liner function in both graphics.

Performance is an important issue when we work on large graphs, and the main advantage of our approach is the runtime reduction. On the evaluated graphs the multilevel approach speed up the clustering tasks by factors from 2 to 7 compared to Fast Modularity algorithm.

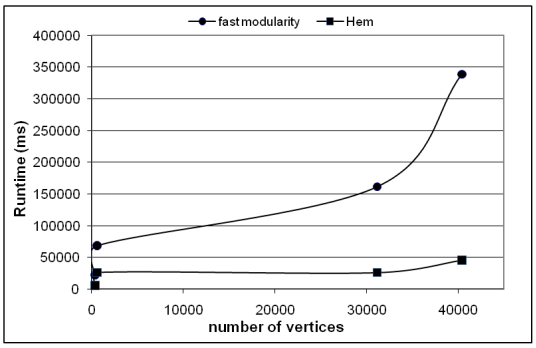
Table 2. Modularity quality and elapsed algorithm runtime for *cond-2003* and *cond-2005* networks

cond-2003	Fast Modularity	HEM	*cond-2005*	Fast Modularity	HEM
Q	**0.415**	0.411		0.397	**0.406**
$Time(ms)$	161399	**25097**		338995	**45444**

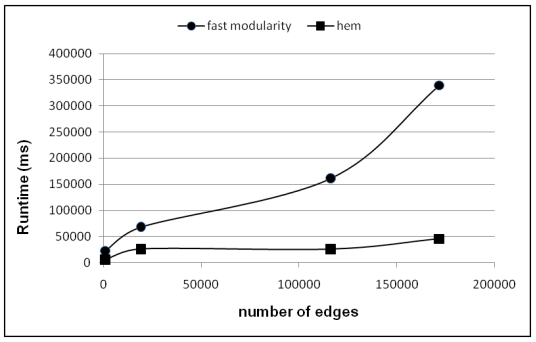
5 Conclusion

In this paper we presented a method for clustering data represented as graphs. The method follows a multilevel graph clustering schemes, making use of the Fast Modularity algorithm in the partitioning phase. This approach has a considerable speed advantage over direct applying the partitioning algorithm on an entire graph. As the partitioning algorithm works on the reduced graph, the runtime complexity is smaller than it would be if it was done on entire graph. So, we observed that the entire method worst-time is $O(|E|)$, which allow us to perform

[1] http://www-personal.umich.edu/ mejn/netdata/



(a) *Runtime by number of vertices*



(b) *Runtime by number of edges*

Fig. 3. Performance comparison of Fast Modularity and HEM

clustering in large graphs in a short time. Moreover, the method keeps partition quality, since the coarser graph, on which the clustering algorithm is applied, is built attempting to be a good representation of original one.

In our experiments, we evaluated the proposed algorithm on three different graphs observing the runtime and four objectives measures. For the two graphs, we compared the results of using multilevel partitioning RM, HEM and MHEM against the original Fast Modularity Algorithm. HEM and MHEM produced quality results close to Fast Modularity, but in much smaller runtime.

We consider that our method can also have the partitions quality improved by applying a refinement algorithm on uncoarsening phase. The refinement algorithm will allow to switch boundary vertices from a partition to other upgrading an objective function, without increase the global worst-time. Another interesting point of improvement is to modify Fast Modularity algorithm in the Partitioning Phase to take into account edges weights.

References

1. Bollobas, B.: Modern Graph Theory. Springer (July 1998)
2. Diestel, R.: Graph Theory (Graduate Texts in Mathematics). Springer (2005)
3. Cook, D.J., Holder, L.B., Ketkar, N.: Unsupervised and supervised learning in graph data. In: Mining Graph Data. John Wiley & Sons (2007)
4. Schaeffer, S.E.: Graph clustering. Computer Science Review **1**(1) (2007) 27–64
5. Garey, M., Johnson, D., Stockmeyer, L.: Some simplified NP-complete graph problems. Theoretical Computer Science **1** (1976) 237–267
6. Newman, M.E.J.: Fast algorithm for detecting community structure in networks. Physical Review E **69** (2004) 066133
7. Karypis, G., Kumar, V.: Analysis of multilevel graph partitioning. In: Supercomputing '95: Proceedings of the 1995 ACM/IEEE conference on Supercomputing, New York, NY, USA, ACM Press (1995)
8. Karypis, G., Kumar, V.: Multilevel k-way partitioning scheme for irregular graphs. Journal of Parallel and Distributed Computing **48** (1998) 96–129
9. Dhillon, I.S., Guan, Y., Kulis, B.: Weighted graph cuts without eigenvectors: A multilevel approach. IEEE Transactions on Pattern Analysis and Machine Intelligence **29**(11) (2007) 1944–1957
10. Hendrickson, B., Leland, R.: A multilevel algorithm for partitioning graphs. Technical Report SAND93-1301, Sandia National Laboratories (1993)
11. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **22**(8) (2000) 888–905
12. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Physical Review E **69** (2004) 026113
13. Newman, M.E.J.: Finding community structure in networks using the eigenvectors of matrices. Physical Review E (Statistical, Nonlinear, and Soft Matter Physics) **74**(3) (2006) 036104
14. Heer, J., Card, S.K., Landay, J.A.: prefuse: a toolkit for interactive information visualization. In: CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, ACM (2005) 421–430
15. Newman, M.E.J.: The structure of scientific collaboration networks. Proceedings of the National Academy of Sciences of the United States of America **98** (2001) 404