An Approach for Community Cutting

György Frivolt and Mária Bieliková

Institute of Informatics and Software Engineering
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava
Ilkovičova 3, 842 16 Bratislava, Slovakia
frivolt@fiit.stuba.sk, bielik@fiit.stuba.sk

Abstract. Informational networks and networks created based on social interactions possess high complexity. Yet the networks can be decoupled into sets of vertices, communities, more interconnected among each other then by other vertices of the network. For mining internal structure of informational and social networks approach for finding community around a seed of vertices is described. Most of existing clusterization approaches intend to identify all communities in the network, our goal is only to identify the community around a vertex or a set of vertices. We give also short overview of tools for network analysis and their functionalities is given.

1 Introduction

Massive networks can be observed from various aspects of the world. Nowadays more massive networks have measurable and analysable form: informational networks such as the Web, networks of social interactions as phone call, SMS or email communication can be captured. We give overview of operations, representations and tools for modelling graphs. The intention is to build a basis for further development of a software tool for analysing massive graphs. We also introduce an approach for finding community around a set of vertices, which should perform better than other approaches mining all communities in the network.

Those networks that are interesting for research usually perform huge number of vertices and edges. The amount of data forces us to think about how to tackle spatial and computational complexity. Often the data does not fit in the system memory. Just for a glance: the crawls of a search engine has to process around 200 million web pages and 2 billion hyperlinks [4, page 10], the number of communications in a phone network can scale as 50 million for a month. Scalability is perhaphs the most obvious problem spot one can notice when implementing and testing a tool for network analysis.

2 Network Analysis Tools

Projects for massive network analysis and visualisation are being developed (Pajek [1], JUNG, InFlow, DyNet, Cyram NetMiner). These tools mostly provide

functions such as ranking vertices according to their importance, centrality measurements, clustering and visualization of the network and other functionalities.

We consider network representations as list of vectors: for every vertex a list of its neighbour vertices is defined; matrix representation: the adjacency matrix serves a good representation as useful operations can be performed on the adjacency matrix (computing co-citation matrix, importance ranking algorithms, like PageRank, HITS). However as most of networks being analysed are sparse, storing network as a matrix can cause too expensive space consumption. We find external storage of the network to a relational database as the simplest and most effective solution currently. Edges are defined as relations, containing edge head, tail and edge attributes.

The initial networks are usually objects for manipulations. These network manipulations result in changed networks, which are further analysed and should serve a basis for making decisions for the user or should present a list of results which helps navigation in the network. The following list of operations were chosen as the most interesting from our point of view.

Clustering Identification of clusters/community structures in the network. The input of the clustering operation is a graph and generates a list of clusters containing vertices, and a hierarchy of clusters.

Shrinking Shrinking sets of vertices to one vertex, for instance shrinking identified clusters to vertices.

Filtering Operations for filtering out edges and vertices with given criteria.

Co-citation network Operation over the adjacency matrix: $\mathbf{E}^T\mathbf{E}$.

Ranking Implementation of vertex/edge importance ranking, such as centrality measurements [2], PageRank [6] or HITS [3]

These operations generate a new network from the original one. We distinguish two modes for this process. In *generate-mode* a new graph is produced, which is completely independent of the originally processed. Generation of a graph from the source graph causes computational effort when it is executed, but every operation is done on the produced graph afterwards. For *wrap-mode* the graph manipulation decorates the graph under processing, every operation on the wrapper graph delegated to the data of the wrapped graph and does not create physically any graph. The produced graph is kind of a view of the processed graph. This mode causes small effort when it is executed, but a little more computation when the wrapper graph is processed. Also the spatial demands of this operation should be much smaller, as no graph is generated.

Shrinking, clustering and generation of cluster hierarchy are operations of *global decomposition. Local decompositions* are cutting out a part of the graph (for instance vertices of a component) or shrinking all but one cluster produces a context of the left alone cluster. Fig. 1 shows an illustration of the decompositions.

Clusterization of massive networks, consisting of even more million vertices can show up as too complex problem to solve. For instance Newman-Girvan edge-removal algorithm takes $O(mn^2)$ [5], other algorithms based on hierarchical

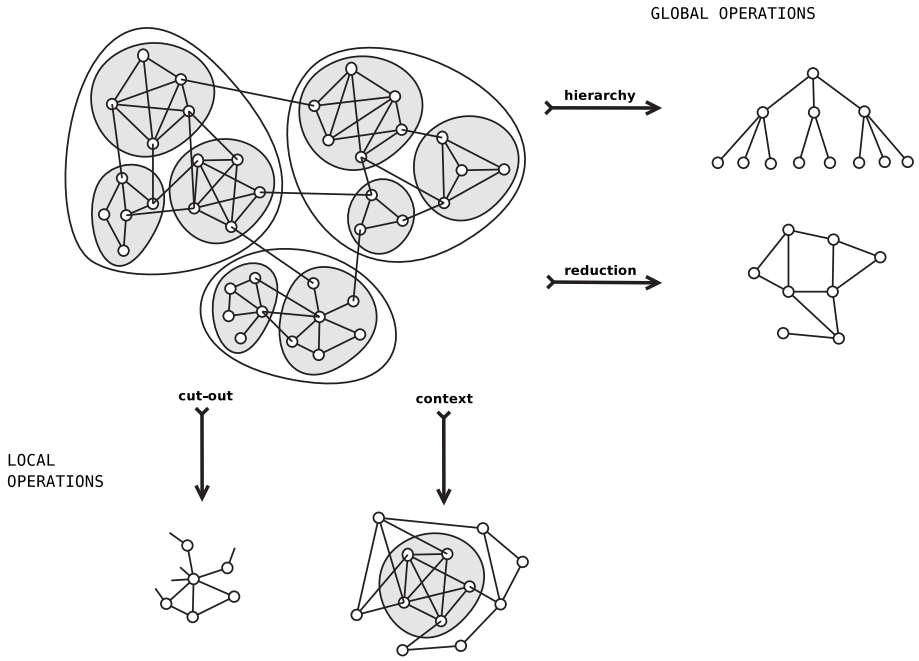

Fig. 1. Illustration of transforming graphs to other forms.

performing worse separation of communities need $O(n^2)$ time for computation in case of sparse graph. Even this may be unfeasible for networks with more million vertices.

3 A community-cutting approach

We propose an algorithm for finding community containing an initial seed of vertices. As our intention is not to classify all vertices to communities, we need to traverse a smaller part of the network. We call this approach as community-cutting.

Our approach is a modified breath-first search algorithm (alg. 1). Whereas the BFS prioritises the vertices based on the distance from the seed vertices, varying priorities set as the proportion of the number of neighbours already cut-out and the total number of adjacent vertices are used. The priorities are updated based of eq.1:

$$priority = \frac{|N_k \cap C|}{|N_k|} \tag{1}$$

where N_k is the number of neighbors of vertex k and C is the of vertices assigned to the community.

Algorithm 1: Community-cutting algorithm

Data: graph $G(V, E)$, seed of initial vertices S, maximum number of vertices to cut L
Result: set of cut-out vertices C

```
1  C = S;
2  list = S;
3  while list is not empty and |C| < L do
4      v = pop(list);
5      add v to C;
6      forall neighbours k of v do
7          if k not in C then
8              if k in list then
9                  re-rank k in list with priority: |N_k ∩ C| / |N_k|;
10             else
11                 add k to list with priority: 1 / |N_k|;
12 return C;
```

Algorithm traversing through the whole network visits $O(n+m)$ vertices and edges. Visiting a new vertex takes $O(\log n)$ time for updating the priority using Fibonacci-heap in the worst case. Therefore the whole algorithm takes generally $O(m + n \log n)$ time.

We used a network of mobile communications of one month period. The communication consisted of more types including calls, SMS and MMS. This is a huge network of more than 57 million edges, well capturing social interactions. We considered the network as oriented. During the testing problem with vertices of very high degree (higher than 100.000) was noticed. For tackling the problem vertices with degree higher than 1000 were filter out and not visited during the community cutting.

Fig.2 shows the priorities of vertices assigned to the community during running of alg.1. Priority of a vertex gives what proportion of its incoming neighbours is already assigned to the community and the total number of incoming edges. If the priorities are low, perhaps the algorithm reached the borders of a cluster of vertices and it is approaching to a neighbor cluster. Fig.2 shows a wavering evolution of the priorities.

4 Conclusions and future work

We presented briefly functionsalities usually implemented by tools for massive graph analysis. An algorithm for cutting out a part of the graph was proposed.

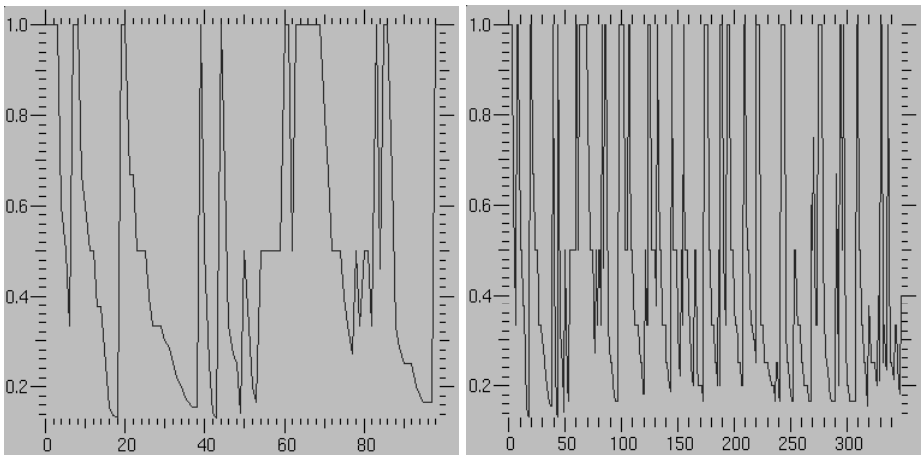

Fig. 2. Priorities of the vertices during the network traversal. The left and the right diagram illustrates the community cutting of 100 and 350 vertices respectively. The initial seed consisted of one vertex in both cases. The highest the priority is, the more neighbors of the vertex has been already assigned to the community.

This algorithm is necessary for finding a portion of the graph based on the given initial seed, which can be further processed in the memory.

Further improvement of the algorithm consist in changing the condition for finalization of the traverse. Setting a minimal threshold of the vertex priority to be processed perharps yield tighter or wider communities depending on the threshold. As it was shown on fig. 2 the priorties can decrease to low values, signing that the new vertex was surrounded by few vertices already included to the community. We belive that introducing threshold could result better defined community.

We intend to test the approach on theoretical such as random networks or small world model; and real networks, as Web and Wikipedia. We hope that our approach can be useful for finding surrounding topics in Wikipedia and the community including the topic, helping the user navigation. Fig. 3 shows visualized clusters in Wikipedia.

We belive that approaches for finding community only arround a seed of vertices, not visiting all vertices of the network are useful for their lower time consumption and applicability on huge networks.

References

1. Pajek. http://vlado.fmf.uni-lj.si/pub/networks/pajek.
2. Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25:163–177, 2001.
3. Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46:604–632, September 1999.

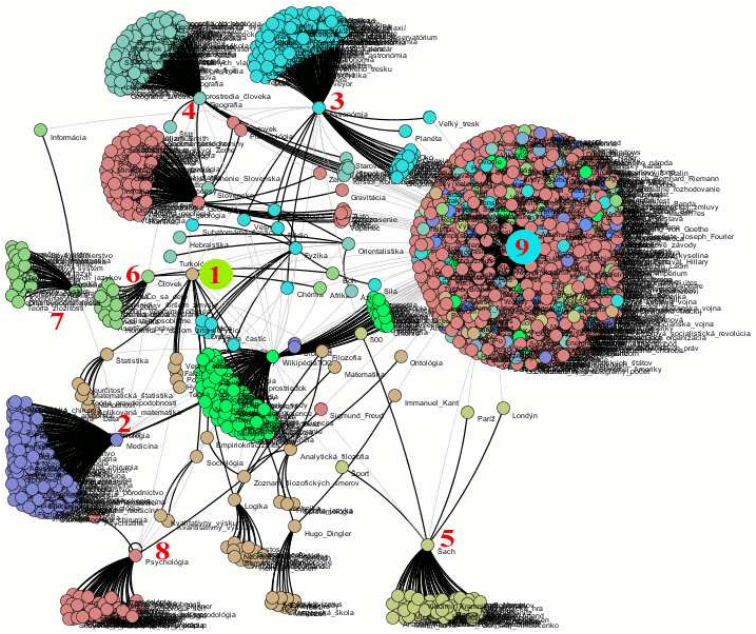

Fig. 3. Visualized part of Slovak Wikipedia. The picture shows that similar topics are referenced by and refers to hub topics such as Medicine (2) or Psychology (8).

4. Mark E. J. Newman. The structure and function of complex networks. *Physical Review Letters*, March 2003. cond-mat/0303516.
5. Mark E. J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical Review Letters*, 2004. cond-mat/026113.
6. Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Standford Digital Libraries, Technologies Project, 1998.