.

National Center for Scientific Research
““Demokritos””

Institute of Informatics and Telecommunications

Internship Report

Nikolaos Koulouris

October 16, 2014

1 Introduction

This is a report concerning the internship that I did at Demokritos, specifically at the Institute of Informatics and Telecommunications, under the supervision of Mr. Paliouras and Mr. Vogiatzis. This internship started on May 2013 and the biggest portion of the work was realized during the summer of 2013, while some extensions where made during the following year.

The internship was research orientated and its main goal was to study and analyze social networks from the scope of machine learning. Initially, many published research papers on clustering and classification were studied as they play an important role on social network analysis. The next step was to look through the literature for articles concerning clustering on hypergraphs. Finally, it was decided that I should focus on one of the few methods that have been proposed for this problem. The method proposed by Murata on [2] is discussed in detail over the next sections.

2 The algorithm

Murata on his paper [2] describes an algorithm for clustering on hypergraphs, that is in multi-partite networks. The main idea of the algorithm is based on the famous Louvain method [1] for clustering on simple networks. This idea is extended on this paper so as to apply to hypergraphs.

The main purpose of the algorithm is to find a cluster distribution among the edges that minimizes a metric quantity, named Q_h. This quantity indicates the quality of the clusters created in the network. The problem of finding the optimal clusters that minimize Q_h is an NP-hard problem, so the algorithm that is used is an approximation algorithm. The algorithm runs in two phases in order to avoid local minimums of Q_h. The algorithm's main idea is presented in the following pseudocode:

```
Require: Connectivity array $A$ of graph $H$
Assign each node in $H$ a unique label
repeat
    //Phase1
    repeat
        for every node in $H$ do
            for all possible labels do
                Calculate $Q_h$
            end for
            Update node's label according to $\min Q_h$
        end for
    until a local minimum of $Q_h$

    //Phase2
    Build reduced graph $G$, where each community in $H$ is a new node in $G$
    Edges between communities in $G$ become edges between nodes
    repeat
        for every node in $G$ do
            for all possible labels do
                Calculate $Q_h$
            end for
            Update node's label according to $\min Q_h$
        end for
    until a local minimum of $Q_h$
    Update node labels in $H$ according to changes in $G$
until no change in $Q_h$
```

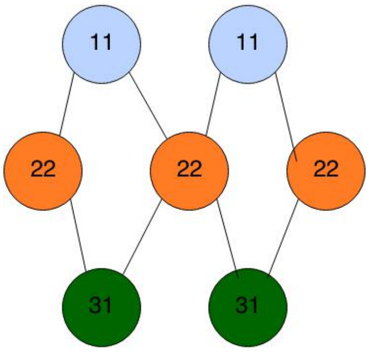
Initially, every node belongs to a different community. In the first phase of the algorithm a local minimum of the quantity Q_h is reached by changing the label of each node. For each node, the reduction in the quantity Q_h is calculated if the node belonged in a different community of the same partite and in each iteration the node is transfered to the community that minimizes Q_h. This procedure is repeated for every node in the graph until a local minimum is found in the whole graph.

In the second phase of the algorithm, a reduced graph is constructed. Each community label at the end of the first phase (when a local minimum is reached), becomes a node in the new reduced graph. In this reduced graph we apply the same algorithm as before until a new local minimum is reached. In this phase, when we change the label of a community, all the corresponding nodes in the simple graph change their labels. At the end of this phase we decompose the reduced graph and the labels of the simple graph are updated accordingly. The whole process, including these two phases, is repeated until a global minimum is reached.

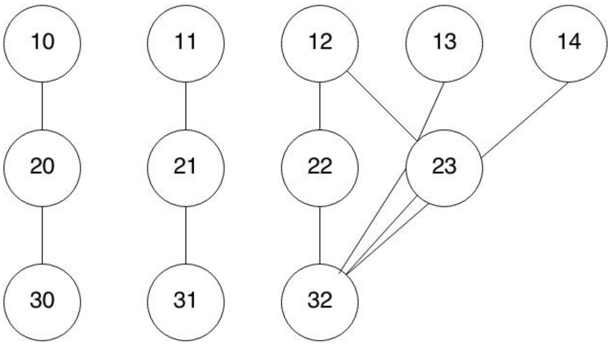
The quantity Q_h is the sum of two sub-quantities. The first one is called Φ and the second one Ψ. The quantity Φ is a complex expression that quantifies the quality of the graph if a node's label change to l. This is achieved by counting the occurrences of the new label l between connecting communities compared to the total summation of all the labels. The quantity Ψ is used mainly during the first iteration of the algorithm, when each node is a separate community. This quantity increases Q_h when a community consists of a single node and thus increases the chance that the node's label will change. In all other cases it is zero. A detailed expression of these quantities can be found on the appendix of Murata's paper.

3 Experiments

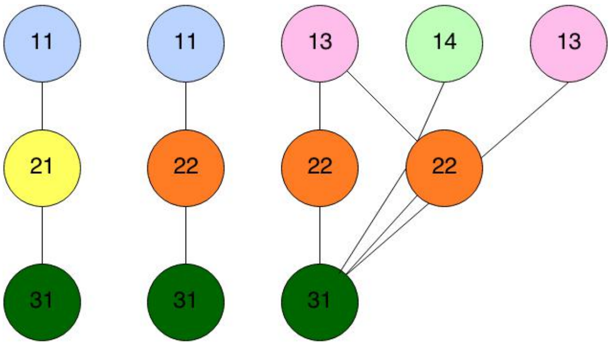
The data used to test this algorithms where small synthetic networks. For testing, some simple networks with 5-6 nodes where used and the algorithm's result was compared with the expected result. One such example, is a small network with three partites and seven nodes in total. The final result of the algorithm can be seen in the following figure. It can be noted that in each partite there is only one community (signified by a different color). This might not have been the most logical result, but the problem that the algorithm converges to graphs with very few communities is well known and discussed in a further section.



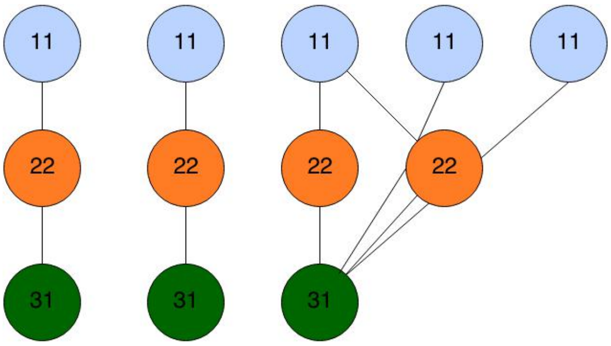
In the next figure, the initial labels of another network that was tested can be seen. In the initial state all the labels are different.



What is interesting in this example is to observe the communities at the end of the first phase, when the first local minimum of Q_h is reached. At this point, most communities seem to be logically separated (with some exceptions, probably caused by quantity Ψ in such small graphs). We can logically infer that the second phase is useless in such small networks because building a reduced graph when the initial network has only 12 nodes will only result in a worse result. This can be confirmed at the final figure.



As with the previous example, the global minimum of Q_h is reached when the algorithm converges to a state where there is only one community in each partite. This state can be seen in the following figure:



Last, MovieLens' dataset was used, which consists of 24000 nodes. The algorithm in this case seems to take too long to terminate.

4 Remarks

A known issue with the algorithm, that was observed and confirmed by the author of the paper, is that the algorithm in small graphs converges to very small graphs. That is to say that for small networks as inputs the algorithm usually converges to a graph with just a single community in each partite. This problem possible results from the quantity Ψ which aids in eliminating communities with just one node. This occurs always in the first iteration of the second phase and in the beginning of the first phase. The same problem has been observed on Louvain's method as well.

A second very important problem with the implementation of the algorithm is its running time. The running time is way too long in the initial step of the algorithm where each node is a different community. In this step, for each node we have to iterate over each community/node in the partite and calculate all the necessary quantities. Even though many quantities are precalculated, this step takes too long to complete.

5 Possible Future Improvements

As far as the convergence is concerned, there are quite a few possible solutions that could be tested. One idea is to look for a limit in the decrease of Q_h instead of a local minimum in each step. In this way the problem of finding networks with very few communities may be solved because the algorithm will stop a few steps before the minimum where there are too few communities. Another idea is to calculate Q_h in a different way on the reduced graph so as to take into account the weigh of the edges between communities. Another improvement during the second phase is that Ψ could not be used at all, or it could be adjusted in some way. That is because each node in this phase could represents a thousand or just one node in the simple graph and this has to been taken into consideration.

In order to improve the running time, the number of different communities in the initial graph should be reduced. This could be achieved with some preprocessing in order to find nodes with many similar hyperedges. These nodes could belong to the same community. Another idea is not to check all available labels that can replace a node's label, but to find the best k labels (according to some very simple metric) and check only among those for the best.

Finally, an interesting improvement could be to create a weighted reduced graph based on how many hyperedges exist between communities. The weighted reduced graph is very easy to be created but the problem that arises is that the quantity Q_h would have to be adjusted to take advantage of the weights.

6 Bibliography

[1] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[2] X. Liu and T. Murata. Detecting communities in k-partite k-uniform (hyper) networks. *Journal of Computer Science and Technology*, 26(5):778–791, 2011.