Propagation Mergence for Community Detection

Dongqi Su
Department of Computer Science
University of Illinois at Urbana-Champaign

May 2017

Abstract

The Louvain method [1] is a successful technique for community detection that decomposes a network by optimizing the modularity of the partitions. It was not designed, however, for cases where a network is incomplete and network nodes contain some unused information. A new method is introduced here, Propagation Mergence (PM), that was designed to handle incomplete networks, and leverage node information to improve community detection. PM utilizes PageRank, propagation, and greedy merging to 1) locate local hubs, 2) discover small subgraphs, and then 3) create larger subgraphs via merging. An illustrative literature network example is presented to demonstrate that PM yields results that improve upon the Louvain method.

1 Introduction

Propagation Mergence (PM) works differently than Louvain method. We believe its effectiveness should contribute to its ability to pick up subtle differences in textural information. To get a good measurement of the content (in this case textual information) of a given network, we introduce the concept of content vectors which describe the content of nodes or subgraphs. In PM, content vectors were generated by a data-driven approach (TopMine [2]) and TF-IDF [4] measurements from abstracts (which contain textual information) of each paper. Once the content vectors are ready, we will assign the similarity as the weight for each edge. Then we run PageRank on this network to identify the most influential nodes. Using these selected nodes as "local hubs," we can perform a variant of PageRank to propagate the topics information from the local hubs to form many small subgraphs. In the final stage, we greedily merge small subgraphs based on their similarity to obtain the final solution.

The complete algorithm consists of the following stages:
1. Vector preparation: set up content vectors and topic vectors for each node.
2. PageRank: find most influential nodes as sources.

3. Propagation: build local clusters.
4. Mergence: merge local clusters into final clusters.

2 Content vectors, PageRank

2.1 Content vectors

In applications involving textural information, phrases are more meaningful than words. Instead of using bag-of-words as features, we use bag-of-phrases to build the content vectors. The raw input for this stage are the text descriptions of the papers. In our experiment, we concatenate the abstract and three copies of the title (all from one paper) to form such text description.

Notice that the ordering of words and punctuations should be preserved for phrases extractor. For phrases extractor, we can choose NLP approach or data-driven approach. In this experiment, we choose a data-driven approach from the first half of the TopMine algorithm. This simple algorithm uses expectation of merging two words as a statistical measurement to discover phrases from the text. More details can be found in the original paper [2].

After obtaining bag-of-phrases for each paper, we then use them to generate TF-IDF vectors. TF-IDF (term frequency - inverse document frequency) can filter out common word/phrases among all the papers, for example, "case study" and "design problem" are very common in papers from Design Automation Conference. The TF-IDF weight for phrase i in document j is:

$$p_{ij} = TF \cdot IDF$$

$$TF = \frac{\text{appearances of phrase i in document j}}{\text{total number of phrases in document j}}$$

$$IDF = \ln \frac{\text{total number of document}}{\text{number of document with } i \text{ in it}}$$

After TF-IDF vectors are ready, we can define weight for each directed edge:

$$w_{ij} = \frac{\sigma(v_i, v_j)}{\sum_{c \in C} \sigma(v_i, v_c)}$$

where σ is the cosine similarity function, and v_i is the TF-IDF vector for node i. C_i is all the outgoing edges from node i. B_i is all incoming edges of node i. $A_i = B_i \cup C_i$ is the set of all edges connected of node i.

2.2 PageRank

After we set up the weight, we can run PageRank [3] on the entire network. The updating method for s_i (PageRank score of node i) is written as follow:

$$s_i = \sum_{j \in B_i} w_{ij} s_j$$

where $w_{ij} \in [0, 1]$ is the normalized weight of edge we defined previously.

Using power method, we can update score $s_i = 1$ for all the nodes in the network until converge. Since all edge were normalized, the sum of all score will remain as a constant throughout the updates.

In the sense of finding the most influential nodes, what PageRank essentially doing is transferring credits from papers to papers. The underlying assumption is that if paper i cites another paper j and their similarity is high, then more credit will go from i to j. But since the weights of all outgoing edges of each node add up exactly to 1, no credit will be created or be destroyed during the transferring process.

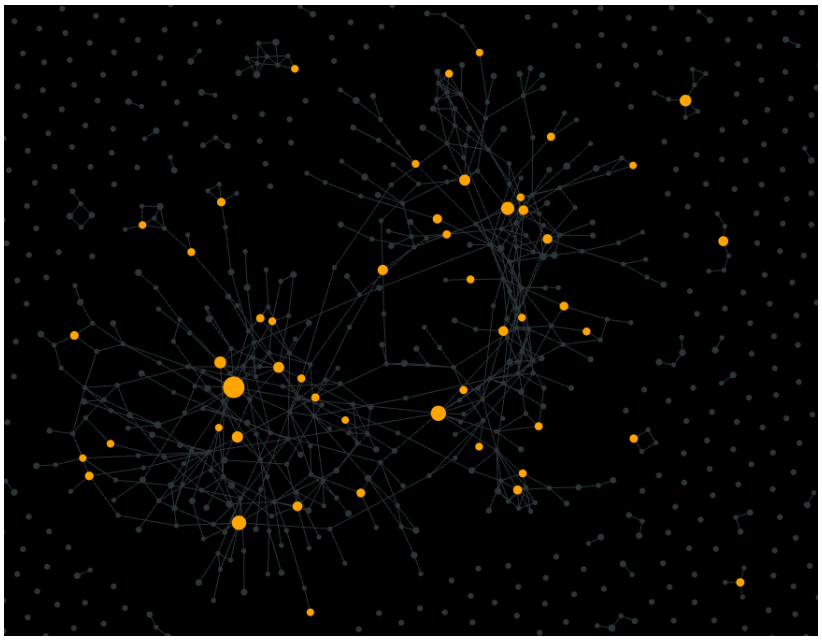

Figure 1: visualization of top 50 most influential nodes in the network.

Figure 1 highlight the top 50 nodes in a network. Notice that these nodes were distributed evenly on the network, thus this ranking can also be used to find the "local centers" of the network.

3 Propagation Mergence

The idea behind Propagation is that once we obtain the most influential nodes in the network, we can use these nodes as "sources" to propagate topics/belief through the network, and hopefully, the Propagation itself is enough to form

accurate subgraphs (communities) around the "sources." However in our experiment, using this approach alone yields poor result.

To improve this approach, we add another stage called Mergence into our algorithm. The idea behind Mergence is to decompose the network into many small local subgraphs first, and then greedily merge them into bigger subgraphs based on some criteria. This additional stage allows the coexistence of different sizes of subgraphs after the Mergence stage.

3.1 Propagation

For propagation, we design a simple algorithm using PageRank again. We first pick a set of "sources" from the previous stage (the most influential nodes) and set up their topics vectors as unique one-hot vectors (such that the length of the topics vectors is equal to the number of sources). For example, if we pick five sources, (in practice, we will need to pick up to a few hundred sources.) then the topics vector for the second source will be:

$$t_{\text{2nd source}} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

After setup, we update the topics vector of the "non-sources" nodes using PageRank via a topological order until they converge (topical order can speed up the convergence time; notice that we don't update the "source" nodes.) For each node i, the update method for the topic vector is defined as:

$$t_i = \sum_{j \in A_i} w_{ij} t_j$$

For example, when updating node 101, if it has 3 connections, each with weight 0.6, 0.3, 0.1, then the new topics vector for node 101 is:

$$t_{101} = 0.6 * \begin{bmatrix} 0 \\ 0.3 \\ 0.5 \\ 0.2 \\ 0 \end{bmatrix} + 0.3 * \begin{bmatrix} 0.1 \\ 0.8 \\ 0.1 \\ 0 \\ 0 \end{bmatrix} + 0.1 * \begin{bmatrix} 0 \\ 0 \\ 0.7 \\ 0.2 \\ 0.1 \end{bmatrix}$$

This algorithm can run quickly on an implementation such as Numpy in Python. However, if we replace the topic vector with a dictionary and set up a threshold for message passing, the overall computational complexity can be decreased to match the ordinary PageRank.

When all the topics vectors converge, we determine the initial subgraph assignments based on the index of the largest entry in the topic vector of each

nodes. For example, node 101 belong to cluster 2 if

$$t_{101} = \begin{bmatrix} 0.01 \\ \mathbf{0.56} \\ 0.29 \\ 0.1 \\ 0.04 \end{bmatrix}$$

3.2 Mergence

In the Mergence stage, we merge subgraphs greedily by ranking the options of merging two subgraphs based on the criteria we choose (unlike Louvain method, the merge operation in Propagation Mergence is permanent). In our experiment, we choose the textural similarity as the main criteria. The content vector of a subgraph is defined as:

$$g_i = \frac{\sum_{j \in G_i} v_j}{|G_i|}$$

where G_i is the set of all nodes in the subgraph i. The combined vector g_{ij} is defined as:

$$g_{ij} = \frac{|G_i| \cdot g_i + |G_j| \cdot g_j}{|G_i| + |G_j|}$$

The score of merging subgraph i and j is defined as the cosine similarity of two content vectors divided by the combined size:

$$score_{ij}^{similarity} = \frac{\sigma(g_i, g_j)}{|G_i| + |G_j|}$$

where the denominator could be seen as the regulation term for the "loss" function. By adding the size penalty, we prevent some subgraphs becoming too big and similar to all its neighbor.

Another criterion we may want to consider is the concentration of the content of the subgraph, which measurement could be calculated by the entropy of the combined vector of subgraphs i,j:

$$score_{ij}^{entropy} = -entropy(g_{ij})$$

In addition, we can also add modularity function as a filter to refuse options that decrease the overall modularity of the partition.

$$score_{ij}^{similarity+modularity} = score_{ij}^{similarity} \cdot (\Delta Q > 0)$$

where $(\Delta Q > 0)$ is a Boolean expression, and ΔQ can be simplified analytically from the modularity function Q [1].

Careful readers may speculate whether it's possible to omit propagation stage and let each node form a unique cluster directly from the Mergence stage. In practice, this is not recommended because we merge subgraphs greedily and irreversibly (unlike in Louvain method). Propagation is essential for forming the initial reliable subgraphs.

4 Observation and Result

4.1 Observation

Since the algorithm run in an unsupervised manner with no true "label" for the data we used in our application, we built a visualization tool for the human expert to verify the result. We also run a basic version of the Louvain method for comparison. We run the algorithm with the following setting:

* criterion = $score^{similarity+modularity}$
* Initial subgraphs number = 180
* Final subgraphs number = 89
* Propagation iteration number = 300 (times per nodes)

For the same density, Propagation Mergence identifies more communities than Louvain method. (89 vs. 69 clusters) After careful analysis, we discovered that some very small communities (1–2 nodes) gain independence from large communities in Propagation Mergence. Propagation Mergence correctly picks up the differences in textural information, hence making the separation possible.

To our surprise, Propagation Mergence performs very well even without considering the structure of the network. In some areas of the network, the partitions drawn by Louvain method are similar or even identical with those drawn by Propagation Mergence (see Figure 2a & 2b).

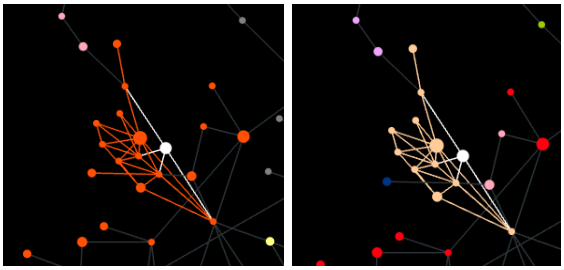

Figure 2a/2b: highlight of the decentralized system communities in Louvain Method (left) and Propagation Mergence (right)

In some areas when the structures are too complicated for Louvain method, Propagation Mergence started to show its advantage via reading the content of the nodes. Here we show a few scenarios where Propagation Mergence makes more accurate (less obvious mistakes) partitions than Louvain Method.

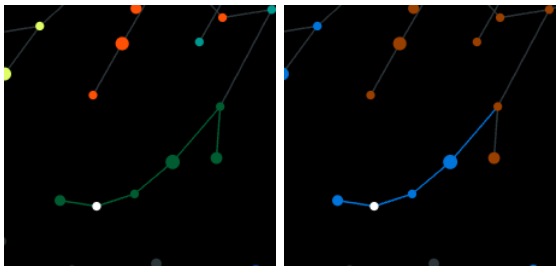

Figure 3a/3b: highlight of the Computer aid design communities in Louvain Method (left) and Propagation Mergence (right)

As shown in Figure 3b, Propagation Mergence recognizes the correct boundary between Computer aid design (blue) and Family Product design (brown), whereas Louvain method (Figure 3a) fails to recognize the boundary (green). We believe the property of correctly recognizing the boundary is coming from the Propagation stage. The similarity measurement dictates the amount of information that can pass from one node to another. If two nodes show low similarity, then one node simply cuts off the connection from the other side.

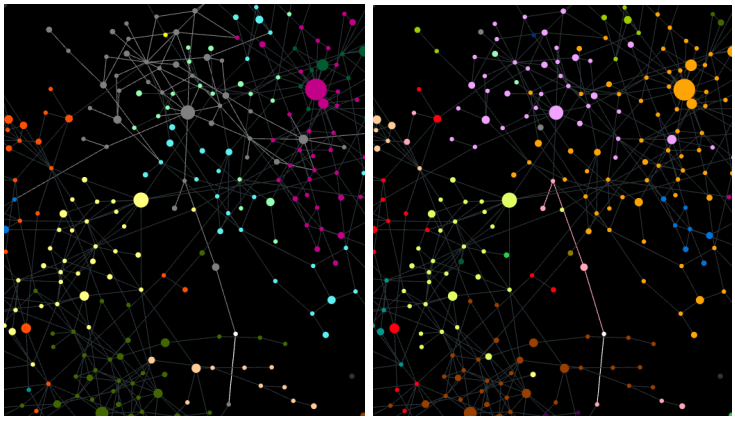

Figure 4a/4b: highlight of the surrogate modeling communities in Louvain Method (left) and Propagation Mergence (right)

As shown in Figure 4b, Propagation Mergence recognizes a community (pink) related to Bayesian network, whereas Louvain (Figure 4a) method merges this community into surrogate modeling community (grey).

Figure 5a/5b: highlight of the uncertainty design communities in Louvain Method (left) and Propagation Mergence (right)

As shown in Figure 5a, Louvain method creates two duplicated communities related to uncertainty design (magenta and green), whereas Propagation Mergence (Figure 5b) correctly merges them (orange). The above two cases show that Propagation Mergence has advantages when content are more informative than structural information.

4.2 Result

Figure 6 shows the complete network partitions by Propagation Mergence.

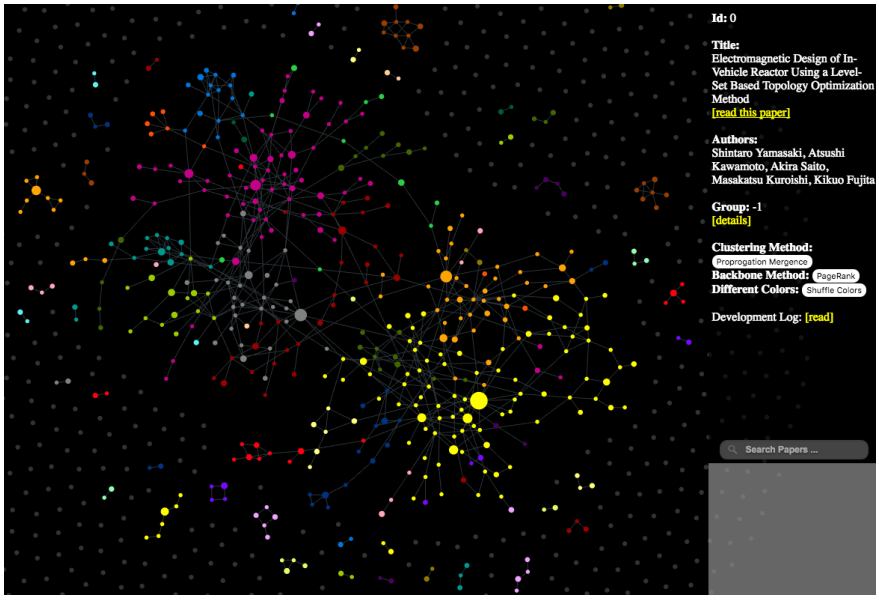

Figure 6: the complete partition result from Propagation Mergence

The proposed algorithm is able to detect the large communities, with the high frequency words/phrases reported below:

- reliability, uncertainty, error, random, reliability analysis, probabilistic, interval
- metamodeling, kriging, kriging model, surrogate, surrogate models, model, approximation
- market systems, market, profit, product, consumer, price, preference
- family, product family, product, product platform, redesign, platform, commonality
- visualization, preference, user, content, product, user generated, interaction
- reconfigurable, reconfigurable system, transformation, adaptive, product, concept generation, system
- decentralized, architecture, process architecture, mistakes, distribution, convergence, impulses
- analytical target, target cascading, analytical target cascading, coordination, target, atc, network
- robust design, robust optimization, robust, interval uncertainty, variation, computer experiments, sequential quadratic programming

The complete code and the visualization tool for the DAC networks can be found at (github) `https://github.com/sudongqi/Propagation_Mergence`. (demo) `http://sudongqi.com/Propagation_Mergence/index.html`.

5 Conclusion

We have introduced Propagation Mergence for network clustering based on content and structural information. It's a general, unsupervised community detection algorithm that has been successfully applied in ASME Design Automation Conference. We showed that when the content of the nodes is available in the network, using this algorithm can yield much more meaningful results than Louvain method.

6 Acknowledgements

I would like to thank Professor James Allison and Tinghao Guo for providing insightful feedbacks on this project. I would also thank Professor Neal Davis for reviewing this thesis.

References

[1] Vincent D Blondel et al. "Fast unfolding of communities in large networks". In: *Journal of statistical mechanics: theory and experiment* 2008.10 (2008), P10008.

[2] Ahmed El-Kishky et al. "Scalable topical phrase mining from text corpora". In: *Proceedings of the VLDB Endowment* 8.3 (2014), pp. 305–316.

[3] Lawrence Page et al. “The PageRank citation ranking: bringing order to the web.” In: (1999).

[4] Juan Ramos et al. “Using tf-idf to determine word relevance in document queries”. In: *Proceedings of the first instructional conference on machine learning*. 2003.