FIMSIM: Discovering Communities By Frequent Item-Set Mining and Similarity Search

Jakub Peschel[1], Michal Batko[1], Jakub Valcik[2],
Jan Sedmidubsky[1], and Pavel Zezula[1]

[1] Masaryk University, Brno, Czech Republic
{jpeschel, batko, sedmidubsky, pzezula}@mail.muni.cz
https://www.muni.cz
[2] Konica Minolta Global R&D, Brno, Czech Republic
Jakub.Valcik@konicaminolta.cz
https://research.konicaminolta.com

Abstract. With the growth of structured graph data, the analysis of networks is an important topic. Community mining is one of the main analytical tasks of network analysis. Communities are dense clusters of nodes, possibly containing additional information about a network. In this paper, we present a community-detection approach, called FIMSIM, which is based on principles of frequent item-set mining and similarity search. The frequent item-set mining is used to extract cores of the communities, and a proposed similarity function is applied to discover suitable surroundings of the cores. The proposed approach outperforms the state-of-the-art DB-Link Clustering algorithm while enabling the easier selection of parameters. In addition, possible modifications are proposed to control the resulting communities better.

Keywords: community mining · frequent item-set mining · similarity search · network analysis

1 Introduction

In recent years, the type of data has dramatically changed. The data is becoming more and more context-dependent, and, therefore, it is necessary to analyze it with respect to the context of a target application. An example of this kind of data is structured network data, such as internet pages, social interactions, protein-to-protein interactions, and many others. With the growing amount of structured network data, there is also a rising need to analyze it efficiently. One of the most important tasks in network data analysis is *discovering communities*, also referred to as the community-mining or community-detection task.

Community mining is a process of uncovering hidden relationships among the elements of network data. These relations lead to the creation of community structures that represent densely packed clusters of network elements. The discovery of such communities can help better understand graph dynamics and

an organizational network structure and can be used as an improvement of recommendation systems, police investigation, business reorganization, and many others.

To discover communities in network-based data, we consider a general data representation in the form of a graph consisting of *nodes* that are interconnected by *edges*. We further consider only undirected and unweighted graphs without self-loops over the nodes. The nodes are not required to contain any additional information, nor any additional network knowledge is taken into account for the purpose of community mining. It is also worth noting that the analyzed graph has to be sparse so that meaningful communities can be discovered. If the graph starts to be too much dense, most of the nodes are becoming the candidates of the community, which can easily degrade to the pathological case when all the nodes belong to one community.

A *community* itself is generally understood as a group of nodes that are more interconnected between themselves in comparison with external nodes. [8,15,7] However, there is no generally accepted definition of a community. For example, Radicchi et al. proposes two categories of definitions: strong communities and weak communities. [15] The strong communities consist of nodes with a majority of their respective neighbours as a part of the community, while the weak ones simplify the condition that a total number of connections (edges) between the community members must be higher than the number of edges connecting community members with the others. Newman et al. consider the community as some sort of hierarchy that can be gradually built from smaller communities, and a given level of hierarchy with a suitable community granularity can be selected. [11] This can be achieved by hierarchical clustering, where strongly connected nodes are gradually grouped together to form a community, and a dendrogram of such groupings then reflects the hierarchical representation. Nevertheless, the lack of a formal definition and the universality of abstract definition often leads to the approaches that strictly prefer disjoint partitioning of the graph instead of overlapping. To solve this issue, we consider the following main assumptions:

- Communities are clusters of highly interconnected nodes;
- Communities have some level of hierarchical structure;
- Nodes can belong to multiple communities.

In this paper, we propose to combine the principles of *frequent item-set mining* and *similarity searching* for the two-step discovery of overlapping communities. The proposed method offers a new perspective on solving the community detection problem as well as outperforms a traditional method for the discovery of overlapping communities.

2 Related Work

Community mining is an NP-complete problem due to the combinatorial nature of graph subsets. [16] In this paper, we focus primarily on three main approaches

for the discovery of overlapping communities: clique percolation, local expansion and link clustering.

2.1 Clique Percolation

Clique percolation methods are based on discovering small clique-like structures that are then based on their overlap merged together. For the purpose of better clique searching, k-cliques are used. These are completely connected subgraphs consisting of k nodes. Unlike maximal clique searching, k-clique searching for one fixed k is a polynomial problem. After the discovery of a set of all k-cliques, their overlap is checked. If two k-cliques share k-1 nodes, they are merged together as a new community. If two communities share a contained k-clique, they are connected together into one bigger community until no such merging is possible.

This approach was first proposed by Palla et al [13] and led to the development of CFinder [1]. Kumpula et al. then improved the clique percolation method by defining sequential order for edges added into a graph to detect k-cliques [9].

The problem of the clique percolation approach is in its chaining of cliques. There exist a possibility that nodes of resulting communities can have a high distance (number of hops) between themselves.

2.2 Local Expansion

This approach uses randomly picked nodes as seeds for the community and greedily expand the community to maximise a fitness function that evaluates the quality of a resulting community.

An example of such an approach is LFM [10].

$$f(c) = \frac{k_{in}^{c}}{(k_{in}^{c} + k_{out}^{c})^{\alpha}} \tag{1}$$

The algorithm picks a random node as a seed and expands it to maximize function in Equation 1 where $k_{in/out}^{c}c$ is external/internal degree of community c and α is resolution parameter controlling the size of the community. After the maximisation is finished, LFM picks a new random node from unassigned nodes as a seed for the next community.

The random seed selection leads to situations where the overlapping community is not detected because its nodes are already assigned to different communities. This approach also does not restrict the diameter of communities, which mean that nodes inside communities may be distanced.

2.3 Link Clustering

Another approach to community mining is to detect overlapping communities by clustering links. The idea is, that node can belong to multiple communities, but the link between nodes defines the relationship between these two objects. [3] By grouping similar edges into clusters, respective nodes can be assigned into

communities defined by these clustered edges. There exist several link clustering methods such as OCMiner [5] and DB Link Clustering [18].

The latter mentioned uses six steps to identify communities:

1. An index of the edges is generated from the graph.
2. An incidence matrix between edges and nodes is created from the adjacency matrix.
3. A similarity matrix between each pair of edges is computed. Modified Jaccard coefficient is used as a distance function.
4. For each yet unassigned edge in the graph, if this edge is a core edge, a new cluster is created, and neighbouring edges connected to the core edges are assigned to this cluster. Edge is core edge if the cardinality of the set of similar edges in the neighbourhood of studied edge is higher than core size parameter.
5. Unassigned edges that are not part of the core are checked, whether they can be assigned to some existing cluster of edges to get final link partitioning.
6. Final link partitioning is transformed into the node communities by collecting incident nodes.

From the selected approaches, the link clustering results in community structures that are the most coherent. The diameter is controlled by the usage of core links that need to be adjacent to each other and, as such, does not tend to grow too much. Because of this, the link clustering approach leads to the most structurally cohesive community structure.

3 Community Mining Process

We propose a two-step approach to discover cohesive communities within a reasonably sparse graph. First, we detect possible community candidates, so-called *cores*, which are densely interconnected sets of nodes. To find such cores, we take inspiration in the problem of frequent item-set mining, for which many different algorithms can be used. Second, we possibly enrich each discovered candidate community with its *surrounding* – a set of nodes that do not need to be mutually interconnected but are densely connected to the core. To find such surroundings, we define a core-to-surrounding distance function and search for suitable surrounding candidates by evaluating range queries, for which many different similarity-search algorithms exist. The enriched communities are finally refined to retain only communities with reasonable size and inter-community overlaps.

3.1 Preliminaries

We consider structured network data as a graph G consisting of a pair of sets (V, E), where V is a set of nodes and E is a set of undirected and unweighted edges. We define a community as a structure consisting of two types of nodes: *core* and *surrounding* nodes.

The core is a group of heavily interconnected nodes of sufficient size. The ideal core of the community is a fully connected subgraph, otherwise known as *clique*. Such core ensures that in the resulting community, the diameter is maximally three hops: one from a starting node to core, one across the core, and the last one from the core to an end node. Although the clique is ideal core, this requirement is too strict; thus, the selection of the right relaxation can dramatically improve the detection of communities over the whole graph.

Although cores of the communities are cohesive groups, the extension of such structure can result in better grouping. This can be caused by imperfections in the process of capturing relations between network nodes. As an example, two members of the community do not use the same communication platform as with the rest of the community. This is the reason why each core is extended by its surrounding. The surrounding is defined as a set of nodes that have "sufficient" interconnection into the core of the community but do not meet the criteria to become a core.

3.2 FIMSIM: Community Mining Algorithm

For the purpose of processing, a graph is assumed in the form of a list of neighbours. The first step is the detection of cores. To detect the suitable core of the community, we use the task of mining frequent item sets, which was originally introduced by Agrawal et al. [2]. This task is often referred to as a market basket analysis and served as a way to detect items commonly bought together. The frequent item-set mining searches a database containing sets of unique items for subsets. Any subset occurring in the database more frequently than *minimal threshold* θ is returned as a frequent set.

In the list of neighbours representation, each node can be viewed as a market basket and a set of neighbours as the bought items. We further extend the list of neighbours of a node by adding the node itself. Frequent sets FS obtained from this extended representation are a superset of the cliques and densely connected clusters provided with reasonable parameters. By selecting the parameter θ, we will obtain all the cliques of size θ and higher.

To prove this assumption, let C be set of nodes of the clique of size k and F_C set of nodes from graph, such as each set of neighbours of selected node $\forall f_C \in F_C$ contains all nodes of clique $C \subseteq f_C$. Then every node of clique C contributes to a frequency of C by one, and thus frequency must be at least $|C| = k$. If $k \geq \theta$, the clique will be part of frequent sets FS. There is one specific pathological case (as illustrated in Fig. 1), where a set of nodes is referenced by third party nodes frequently enough to appear as heavily connected. This case is finally eliminated in the refinement step.

After frequent sets are obtained, it is necessary to eliminate undesirable ones. Sets are filtered based on two parameters; their common occurrence must be higher than their size to eliminate part of cores referenced from third party nodes, and their size must be at least θ. This second condition removes small cores that are often the result of the bottom-up approach of frequent item-set mining.

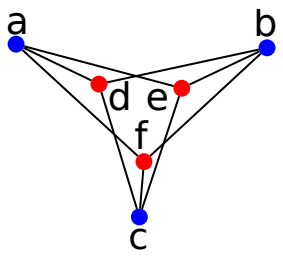

Fig. 1. Pathological case resulting in frequent sets {a, b, c} and {d, e, f} without an existing connection between nodes.

After obtaining the cores of the communities, a similarity range query is applied with the core as a query. For each node, there is then sequentially measured similarity to each core, and if the node is sufficiently similar, the node is marked with the respective core. There is a possibility to replace the range query with a K-NN query, which will allow better control over the resulting size of the community. The similarity is measured against the set of nodes of the core for each node in the neighbourhood of the core.

There is a number of standard distance functions for sets like the Jaccard coefficient that can be found in [17]. However, for comparison of node's neighbour list A with core B, we need a distance function that would result in zero in these two special cases:

1. $A \subseteq B$ – All neighbours of the node A are members of core B;
2. $B \subseteq A$ – Neighbour list of node A contains whole core B.

For this purpose, we propose to use the following distance function:

$$1 - max\left(\frac{|A \cap B|}{|A|}, \frac{|A \cap B|}{|B|}\right). \tag{2}$$

One parameter of the range query is distance radius r. The radius has a major impact on the density of the resulting communities. When distance is smaller, more edges are connected to the core of the communities. Standard measures often prefer this density, so values between 0 and 0.2 are preferred.

As a result of the range query, all nodes are marked with respective cores. By aggregating them into groups, raw communities are obtained. It is possible that the result contains duplicates and products of the pathological cases of a core. These can be eliminated by checking if the core is part of the community. Nodes from the pathological case have very low to zero common neighbours present in the core set and thus are not selected into the community by range query. After filtration, duplicates are removed.

After obtaining the communities, the results can be used as the new cores and search of the surrounding can be started again. This leads to communities with a potentially bigger distance between nodes and thus it is necessary that such approach is used only for suitable use cases. The pseudocode of the whole algorithm is depicted in Fig. 2.

```
def FIMSIM(G, θ, r):
    unfiltered_cores = FIM(G, θ); # frequent item-set mining
    cores = {};
    foreach (core in unfiltered_cores):
        if (core.frequency >= core.size and core.size >= θ):
            cores += core;
    assignments = {};
    foreach (core in cores):
        assignments += simsearch(G, core, r);
    aggregations = aggregate(assignments)
    duplicit_communities = {};
    foreach (candidate in aggregations):
        foreach (core in cores):
            if(candidate contains core):
                duplicit_communities += candidate;
    communities = deduplications(duplicit_communities)
    return communities;
```

Fig. 2. Pseudocode of the proposed FIMSIM algorithm for discovering communities.

4 Experiments

In the experimental part, we compared the proposed approach with the DB-Link Clustering. Both of these approaches share similar parameter space and results in similar graph structures, as can be seen in Fig. 3.

The prototype implementation of FIMSIM is developed with the analytical framework ADAMiSS [14], and for evaluation of similarity, MESSIF framework [4] is used. The former framework allows for a potential optimization based on the density of the analyzed graph.

4.1 Dataset

To evaluate whether this approach is eligible for community mining, we created a collaboration network from a pseudonymised dataset consisting of logged interactions of users with documents on a shared drive provided by Konica Minolta. Typically each document is created, modified and read by various users; thus, the dataset captures active collaboration of users on the document's content and passive interactions of users who only accessed the document.

The input data are in form of tuples: *date user_id*, *action_type*, *document_id*, *document_type*. The *action_type* consisted of six kategories: *Download*, *Previewed*, *Edit*, *Uploaded*, *Created* and *Item Shared*.

We created the user-user interaction network from this data, where two users are connected when they cooperated on the creation process of at least one document. Thus if there is an access log with *action_type Created*, *Uploaded* or *Edit* for both users with same *document_id*. The resulting network consists of 128 users with an average degree of 8.531.

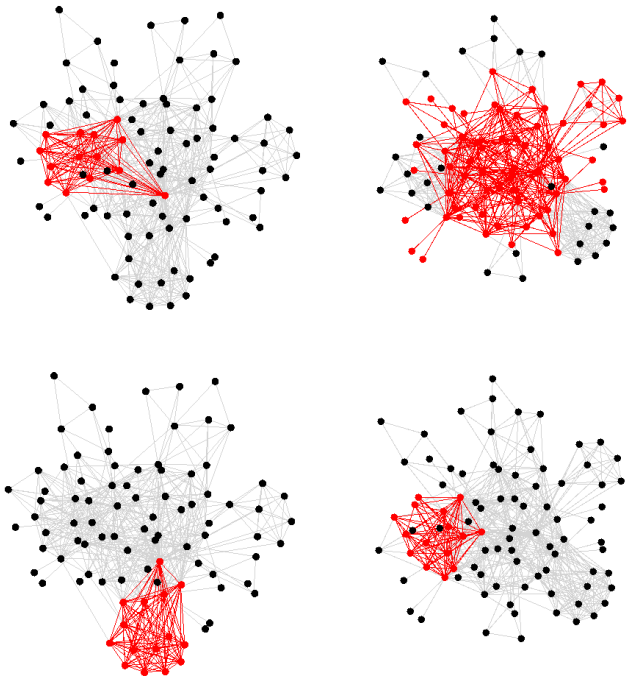

Fig. 3. Example of discovered communities: FIMSIM (Left) and DB-Link Clustering (Right). The example contains two biggest communities obtained from the best runs of both the algorithms. In particular, DB-Link Clustering uses $\theta = 6$ and $r = 0.4$, while FIMSIM uses $\theta = 12$ and $r = 0.1$.

This data represents a collaboration of workers in the organisation and, as such, can be analysed by managers. The discovered communities may function as decision support for managers to create and validate the matrix structure of the organisation.

4.2 Evaluation Criteria

The traditional way of evaluating the quality of graph partitioning is by *modularity*. [12]

$$Q = \frac{1}{2 \cdot |E|} \sum_{c \in C} \sum_{i,j \in V} \delta_{ci} \delta_{cj} \left(A_{i,j} - \frac{k_i k_j}{2 \cdot |E|} \right) \tag{3}$$

Modularity measures the number of edges that connects nodes in the same partitioning reduced by the expected amount of edges in a randomly wired network. This metric is primarily used for the evaluation of exclusive partitioning and thus penalizes overlapping communities.

Because of limitations of the modularity, an extension proposed by Chen et al. was used. [6]

$$Q = \frac{1}{2 \cdot |E|} \sum_{c \in C} \sum_{i,j \in V} \alpha_{ci} \alpha_{cj} \left(A_{i,j} - \frac{k_i k_j}{2 \cdot |E|} \right), \tag{4}$$

$$\alpha_{ci} = \frac{k_{ci}}{\sum_{c_2 \in C} k_{c_2 i}} \tag{5}$$

In this version of modularity, the Kronecker delta is replaced with a coefficient of how many communities the node is involved in. This allows for a decrease in the importance of the node involved in multiple communities and thus is not that penalising for higher density on the overlap. Even though this approach tries to solve the problem of overlaps, it still evaluates non-overlapping communities as having much higher quality than overlapping ones.

4.3 Evaluation

Although algorithms share parameter space, parameters do not match one to one. Because of that, it is important to watch best-achieved results over the whole searched space. The experiment showed that FIMSIM outperforms the quality of found communities in both metrics. The result of the comparison can be seen in Fig. 4 and Fig. 5.

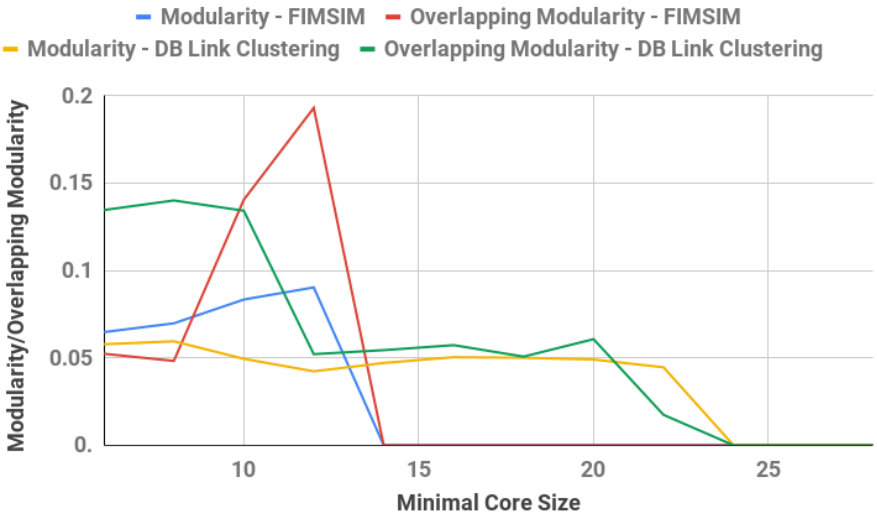

Fig. 4. Quality comparison

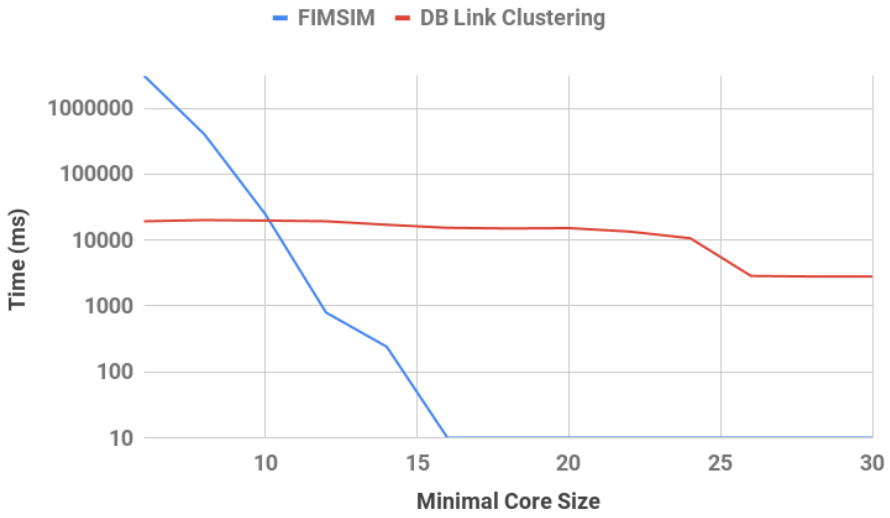

Fig. 5. Time comparison

The experiment also showed that our approach can be better used without knowledge of the correct parameters than DB Link clustering. Because of the nature of frequent item-set mining, when the size of the core is too huge, the algorithm stops almost immediately. Because of that, it can be beneficial to start from higher numbers and lower the core size parameter until a suitable amount of cores is found.

The disadvantage of our approach is that once the core size parameter is too small, the discovery of cores is computationally challenging. Our experiments showed that higher quality of communities is achieved at the higher value of the core size parameter.

5 Conclusion

In this paper, we proposed a different approach to community mining based on a combination of frequent item-set mining and similarity searching. The proposed method uses a two-step process of finding core community candidates and assigning surroundings to them. We proposed a distance function for the selection of suitable surroundings based on two extreme cases.

In cooperation with Konica Minolta, a collaboration network was created as a representation of real-world data, and a new approach was tested. With the usage of modularity and overlapping modularity, we showed that the proposed approach achieves higher-quality results than the state-of-the-art DB-Link Clustering approach that discovers a similar type of communities.

In several steps, we discussed the possibility of modifying the approach to achieve better flexibility in terms of the size of the community as well as the quality of the surrounding of the cores. We also discussed an iterative approach, where after obtaining communities, these are taken as the new cores and with the new range, surrounding for them can be chosen.

Due to the modular nature of the approach, there is a possibility to further improve the proposed approach in terms of optimizing the selection of appropriate frequent item-set algorithms, as well as employing some sort of indexing for assignment of the surroundings to the cores.

Acknowledgment

This research has been supported by the Czech Science Foundation project No. GA19-02033S.

References

1. Adamcsek, B., Palla, G., Farkas, I.J., Derényi, I., Vicsek, T.: Cfinder: locating cliques and overlapping modules in biological networks. Bioinformatics **22**(8), 1021–1023 (2006)
2. Agrawal, R., Srikant, R., et al.: Fast algorithms for mining association rules. In: Proc. 20th int. conf. very large data bases, VLDB. vol. 1215, pp. 487–499 (1994)
3. Ahn, Y.Y., Bagrow, J.P., Lehmann, S.: Link communities reveal multiscale complexity in networks. Nature **466**(7307), 761–764 (2010)
4. Batko, M., Novak, D., Zezula, P.: MESSIF: Metric similarity search implementation framework. In: International DELOS Conference. pp. 1–10. Springer (2007)
5. Bhat, S.Y., Abulais, M.: Ocminer: A density-based overlapping community detection method for social networks. Intelligent Data Analysis **19**(4), 917–947 (2015)
6. Chen, D., Shang, M., Lv, Z., Fu, Y.: Detecting overlapping communities of weighted networks via a local algorithm. Physica A: Statistical Mechanics and its Applications **389**(19), 4177–4187 (2010)
7. Fortunato, S.: Community detection in graphs. Physics reports **486**(3-5), 75–174 (2010)
8. Girvan, M., Newman, M.E.: Community structure in social and biological networks. Proceedings of the national academy of sciences **99**(12), 7821–7826 (2002)
9. Kumpula, J.M., Kivelä, M., Kaski, K., Saramäki, J.: Sequential algorithm for fast clique percolation. Physical review E **78**(2), 026109 (2008)
10. Lancichinetti, A., Fortunato, S., Kertész, J.: Detecting the overlapping and hierarchical community structure in complex networks. New journal of physics **11**(3), 033015 (2009)
11. Newman, M.E.: Communities, modules and large-scale structure in networks. Nature physics **8**(1), 25–31 (2012)
12. Newman, M.E., Girvan, M.: Finding and evaluating community structure in networks. Physical review E **69**(2), 026113 (2004)
13. Palla, G., Derényi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature **435**(7043), 814–818 (2005)

14. Peschel, J., Batko, M., Zezula, P.: Techniques for complex analysis of contemporary data. In: Proceedings of the 2020 International Conference on Pattern Recognition and Intelligent Systems. pp. 1–5 (2020)
15. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks. Proceedings of the national academy of sciences **101**(9), 2658–2663 (2004)
16. Schaeffer, S.E.: Graph clustering. Computer science review **1**(1), 27–64 (2007)
17. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity search: the metric space approach, vol. 32. Springer Science & Business Media (2006)
18. Zhou, X., Liu, Y., Wang, J., Li, C.: A density based link clustering algorithm for overlapping community detection in networks. Physica A: Statistical Mechanics and its Applications **486**, 65–78 (2017)