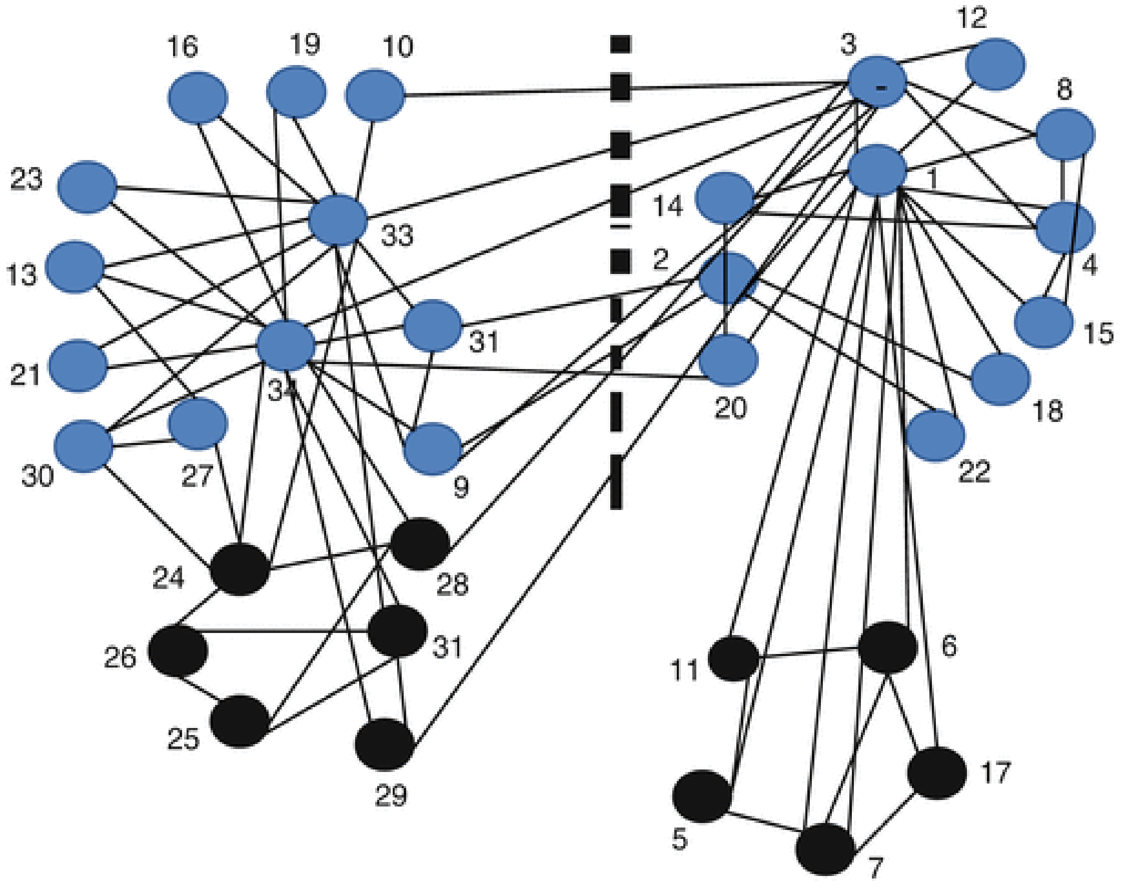
Community detection in DBLP

Community Detection: A community is defined as a group of nodes which are densely connected inside the group, while loosely connected with the nodes outside the group, i.e. a dense graph within a sparse graph.

Community detection in graphs aims to identify the modules and,possibly,their hierachical organization,using mainly the information encoded in graph topology.

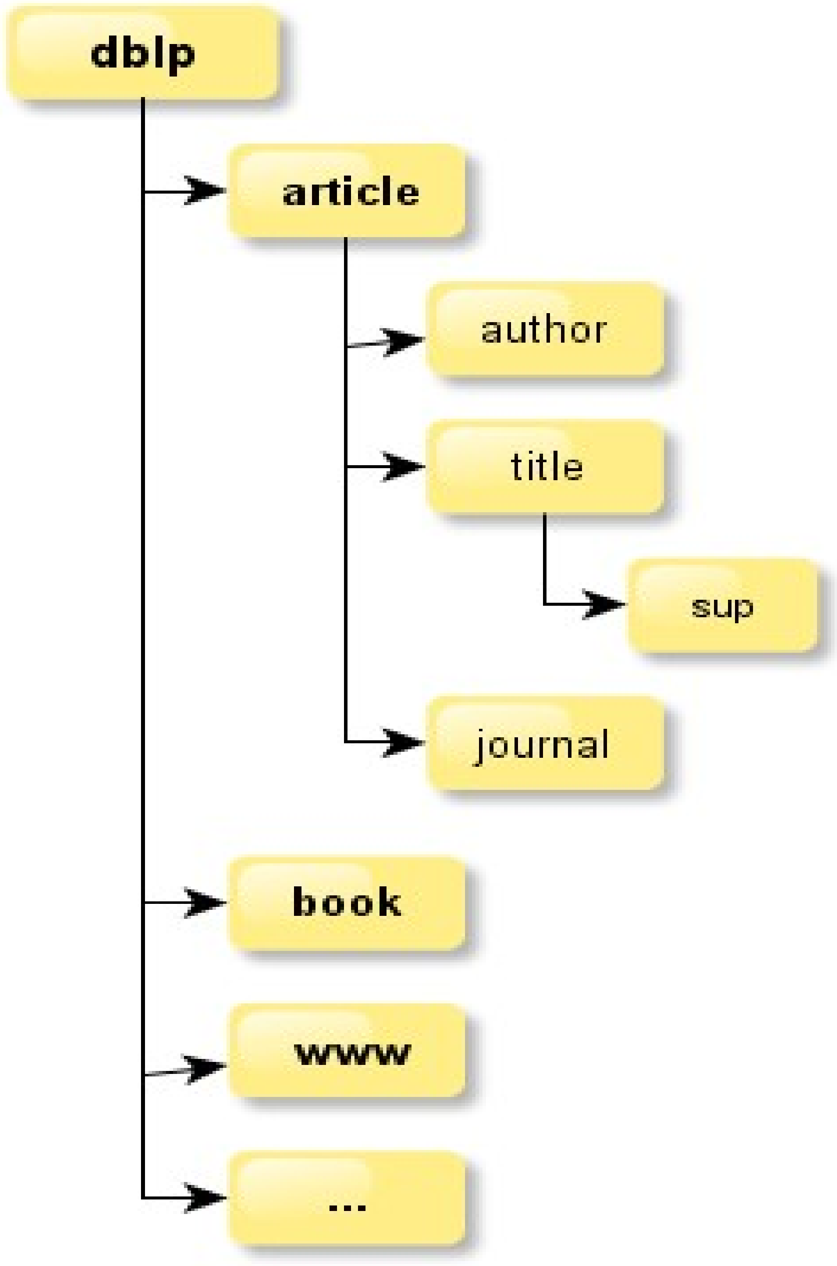
An example of community in a graph can be shown below which shows different communities in a graph:



In our project we had to work on **DBLP**(digital bibliography and library project) corpus.((It can be downloaded from http://dblp.uni-trier.de/xml/) (Download dblp.xml.tar.gz as well as dblp.dts and keep it in the same folder).An excerpt of the XML file looks like:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE dblp SYSTEM "dblp.dtd">
<dblp>

[...]

<article key="journals/cacm/Gentry10" mdate="2010-04-26">
<author>Craig Gentry</author>
<title>Computing arbitrary functions of encrypted data.</title>
<pages>97-105</pages>
<year>2010</year>
<volume>53</volume>
<journal>Commun. ACM</journal>
<number>3</number>
<ee>http://doi.acm.org/10.1145/1666420.1666444</ee>
<url>db/journals/cacm/cacm53.html#Gentry10</url>
</article>

[...]

<inproceedings key="conf/focs/Yao82a" mdate="2011-10-19">
<title>Theory and Applications of Trapdoor Functions (Extended Abstract)</title>
<author>Andrew Chi-Chih Yao</author>
<pages>80-91</pages>
<crossref>conf/focs/FOCS23</crossref>
<year>1982</year>
<booktitle>FOCS</booktitle>
<url>db/conf/focs/focs82.html#Yao82a</url>
<ee>http://doi.ieeecomputersociety.org/10.1109/SFCS.1982.45</ee>
</inproceedings>

[...]

<www mdate="2004-03-23" key="homepages/g/OdedGoldreich">
<author>Oded Goldreich</author>
<title>Home Page</title>
<url>http://www.wisdom.weizmann.ac.il/~oded/</url>
</www>

[...]
</dblp>
```

The DBLP computer science bibliography provides a comprehensive list of research papers in computer science. We construct a co-authorship network where two authors are connected if they publish at least one paper together. Publication venue, e.g, journal or conference, defines an individual ground-truth community; authors who published to a certain journal or conference form a community.
The tags in the corpus are:



- *article* – An article from a journal or magazine.
- *inproceedings* – A paper in a conference or workshop proceedings.
- *proceedings* – The proceedings volume of a conference or workshop.
- *book* – An authored monograph or an edited collection of articles.
- *incollection* – A part or chapter in a monograph.
- *phdthesis* – A PhD thesis.
- *mastersthesis* – A Master's thesis. There are only very few Master's theses in dblp.
- *www* – A web page. It contains all the aliasing of the authors

Problem statement:

Comparing different algorithm for community detection on DBLP corpus.Basically we need to find the communities of authors.

Steps involved are:

1. **Parsing:** The first part Parsing involved generating an authors relationship going through different tags and assigning weights according to the priority,i.e lesser priority tags are given lesser weights,for example, based on similarity of the booktitle weights were given according to the similarity in the title whereas authors who worked on the same journals were given more weight.We then added two edges in the final graph and assigned the weight according to the relationship present between them.We chose SAX parser for parsing the data,We chose it over DOM parser because DOM parser take whole data at a time while SAX parser takes chunks of data at a time and thus more efficient.
2. After that we used three different algorithm on the computed weighted graph:(a) Newman-Girvan (b) Louvain (c) CPM

   <u>Newman-Girvan:</u>

   It is a **divisive method** that detect edges that connect different communities and remove them until clusters are disconnected.It involves four mahor steps:

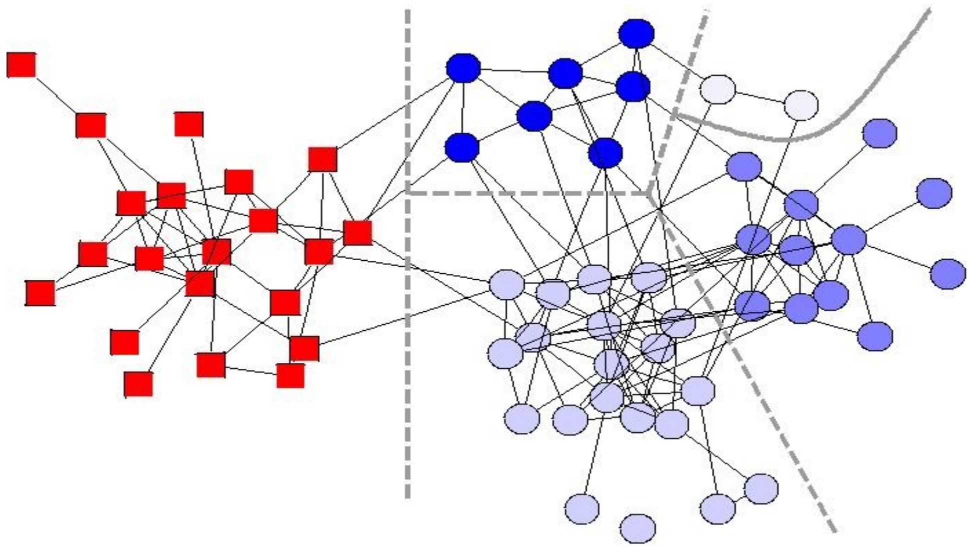
   1. Compute Edge centrality
   2. Remove the edge with the highest centrality
   3. Update Centralities
   4. If |Edges| > 0 , go to step 2

Instead of trying to construct a measure which tells us which edges are most central to communities, we focus instead on those edges which are least central , If a network contains communities or groups that are only loosely connected by a few inter-group edges, then all shortest paths between different communities must go along one of these few edges

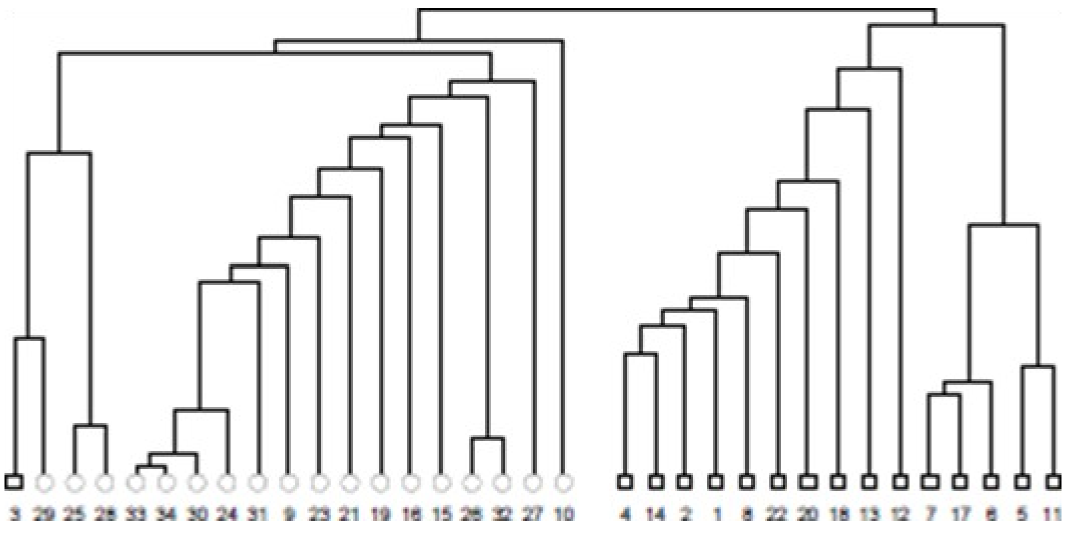
• **Edge Betweenness**:It is defined to be the number of shortest paths between vertex pair that run along the edges,summed over all vertex pairs.This can be calculated in O(mn) on the graph where m is the no of edges and n is the no of vertices.

The algorithm can be thought of as progressing from the root of the dendrogram to the leaves, rather than the other way round, the branches of the tree rep-resenting the order of splitting of the network as edges are removed, and the communities are taken to be the components of the graph, as in the single linkage cluster-ing method. Horizontal cross-sections of the dendrogram represent possible community divisions with a larger or smaller number of communities depending on the posi-tion of the cut.

Communities:



Dendogram -> output of newman-girvan



There are two principal disadvantages of the algorithm of Girvan and Newman:

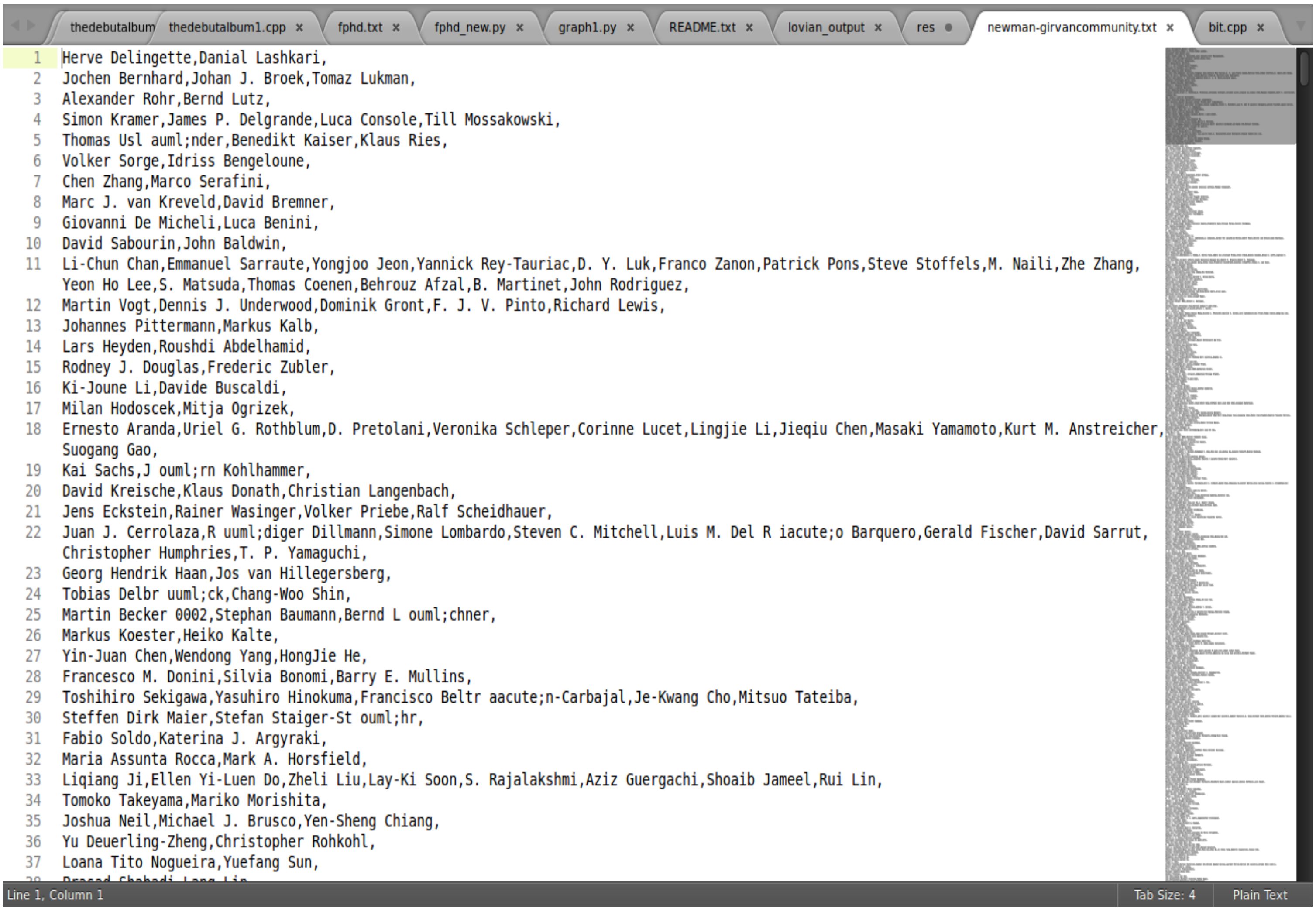
1. It provides no guide to how many communities a network should be split into. To address this problem, Newman and Girvan proposed that the divisions the algorithm generates be evaluated using a measure they call modularity, which is a numerical index of how good a particular division is. Then the modularity is defined to be

$$Q = \frac{1}{2m}\sum_{vw}\sum_{r}\left[A_{vw} - \frac{k_v k_w}{2m}\right] S_{vr}S_{wr} = \frac{1}{2m}\mathrm{Tr}(\mathbf{S}^{\mathrm{T}}\mathbf{BS}),$$

   It lies between [-1/2, 1). Higher the modularity better the classification of communities

2. The other main disadvantage of the algorithm of Gir-van and Newman is that it is slow. Since there are m edges to be removed in total and each iteration of the al-gorithm takes O(mn) time, the worst-case running time of the algorithm is $O(m^2n)$, or $O(n^3)$ on a sparse graph.

Snapshot of the result obtained:



```
Herve Delingette,Danial Lashkari,
Jochen Bernhard,Johan J. Broek,Tomaz Lukman,
Alexander Rohr,Bernd Lutz,
Simon Kramer,James P. Delgrande,Luca Console,Till Mossakowski,
Thomas Usl auml;nder,Benedikt Kaiser,Klaus Ries,
Volker Sorge,Idriss Bengeloune,
Chen Zhang,Marco Serafini,
Marc J. van Kreveld,David Bremner,
Giovanni De Micheli,Luca Benini,
David Sabourin,John Baldwin,
Li-Chun Chan,Emmanuel Sarraute,Yongjoo Jeon,Yannick Rey-Tauriac,D. Y. Luk,Franco Zanon,Patrick Pons,Steve Stoffels,M. Naili,Zhe Zhang,
Yeon Ho Lee,S. Matsuda,Thomas Coenen,Behrouz Afzal,B. Martinet,John Rodriguez,
Martin Vogt,Dennis J. Underwood,Dominik Gront,F. J. V. Pinto,Richard Lewis,
Johannes Pittermann,Markus Kalb,
Lars Heyden,Roushdi Abdelhamid,
Rodney J. Douglas,Frederic Zubler,
Ki-Joune Li,Davide Buscaldi,
Milan Hodoscek,Mitja Ogrizek,
Ernesto Aranda,Uriel G. Rothblum,D. Pretolani,Veronika Schleper,Corinne Lucet,Lingjie Li,Jieqiu Chen,Masaki Yamamoto,Kurt M. Anstreicher,
Suogang Gao,
Kai Sachs,J ouml;rn Kohlhammer,
David Kreische,Klaus Donath,Christian Langenbach,
Jens Eckstein,Rainer Wasinger,Volker Priebe,Ralf Scheidhauer,
Juan J. Cerrolaza,R uuml;diger Dillmann,Simone Lombardo,Steven C. Mitchell,Luis M. Del R iacute;o Barquero,Gerald Fischer,David Sarrut,
Christopher Humphries,T. P. Yamaguchi,
Georg Hendrik Haan,Jos van Hillegersberg,
Tobias Delbr uuml;ck,Chang-Woo Shin,
Martin Becker 0002,Stephan Baumann,Bernd L ouml;chner,
Markus Koester,Heiko Kalte,
Yin-Juan Chen,Wendong Yang,HongJie He,
Francesco M. Donini,Silvia Bonomi,Barry E. Mullins,
Toshihiro Sekigawa,Yasuhiro Hinokuma,Francisco Beltr aacute;n-Carbajal,Je-Kwang Cho,Mitsuo Tateiba,
Steffen Dirk Maier,Stefan Staiger-St ouml;hr,
Fabio Soldo,Katerina J. Argyraki,
Maria Assunta Rocca,Mark A. Horsfield,
Liqiang Ji,Ellen Yi-Luen Do,Zheli Liu,Lay-Ki Soon,S. Rajalakshmi,Aziz Guergachi,Shoaib Jameel,Rui Lin,
Tomoko Takeyama,Mariko Morishita,
Joshua Neil,Michael J. Brusco,Yen-Sheng Chiang,
Yu Deuerling-Zheng,Christopher Rohkohl,
Loana Tito Nogueira,Yuefang Sun,
```

Louvain:

The problem of community detection requires the partition of a network into communities of densely connected nodes, with the nodes belonging to different communities being only sparsely connected. Precise formulations of this optimization problem are known to be computationally intractable. Several algorithms have therefore been proposed to find reasonably good partitions in a reasonably fast way.

The quality of the partitions resulting from these methods is often measured by the so-called modularity of the partition. The modularity of a partition is a scalar value between -1 and 1 that measures the density of links inside communities as compared to links between communities. In the case of weighted networks (weighted networks are networks that have weights on their links, such as the number of communications between two mobile phone users), it is defined as

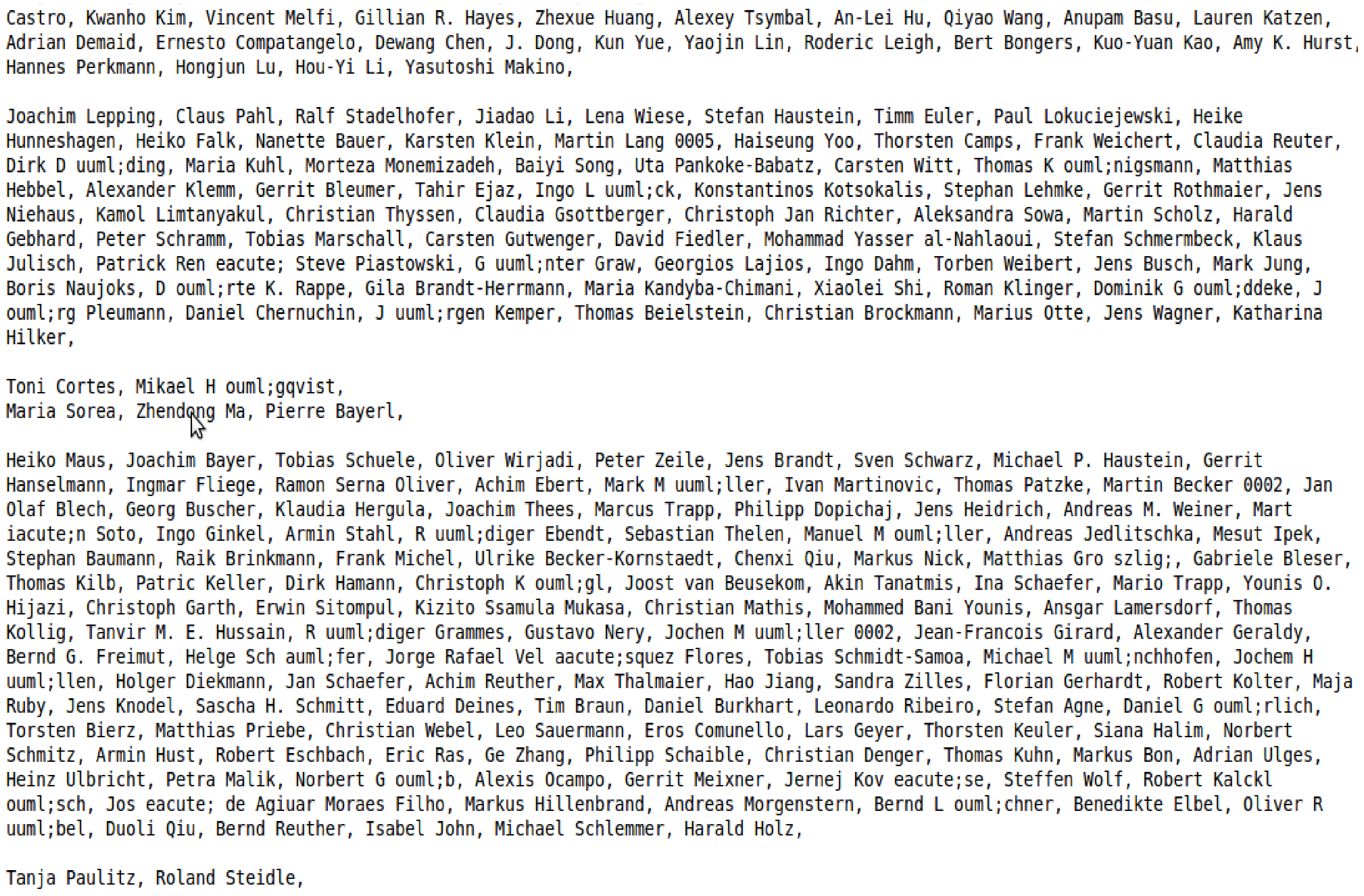
$$Q = \frac{1}{2m}\Sigma_{ij}\left[A_{ij} - \frac{k_i k_j}{2m}\right]\delta(c_i, c_j)$$

Our algorithm is divided in two phases that are repeated iteratively.
Assume that we start with a weighted network of N nodes.
1. We assign a different community to each node of the network. So, in this initial partition there are as many communities as there are nodes.
2. For each node i we consider the neighbours j of i and we evaluate the gain of modularity that would take place by removing i from its community and by placing it in the community of j.
3. The node i is then placed in the community for which this gain is maximum (in case of a tie we use a breaking rule), but only if this gain is positive. If no positive gain is possible, i stays in its original community.
This process is applied repeatedly and sequentially for all nodes until no further improvement can be achieved and the first phase is then complete.

This simple algorithm has several advantages. First, its steps are intuitive and easy to implement, and the outcome is unsupervised. Moreover, the algorithm is extremely fast, i.e., computer simulations on large ad-hoc modular networks suggest that its complexity is linear on typical and sparse data. This is due to the fact that the possible gains in modularity are easy to compute with the above formula and that the number of communities decreases drastically after just a few passes so that most of the running time is concentrated on the first iterations.

Limitation and Time complexity:
The limitation of the method for the experiments that we performed was the storage of the network in main memory rather than the computation time. By construction, the algorithm unfolds a complete hierarchical community structure for the network, each level of the hierarchy being given by the intermediate partitions found at each pass. In this algorithm, however, only the verification of the accuracy of the top level of this hierarchy has been done, namely the final partition found by the algorithm, and the accuracy of the intermediate partitions has still to be shown. Louvain typically executes in **nlogn**, although this is an experimental result and it's not possible to prove it.

Snapshot of the result after running on DBLP dataset:



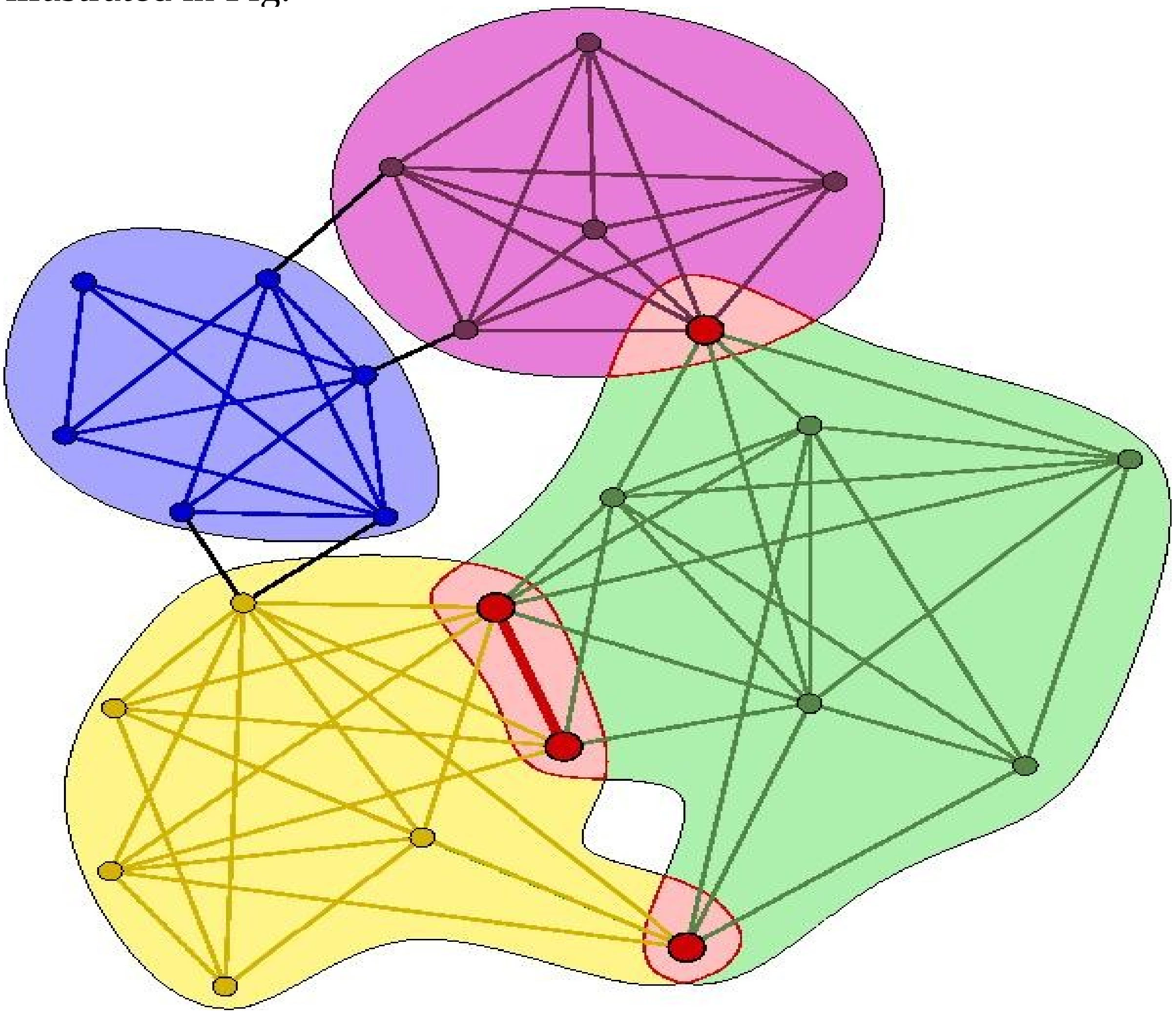
```
Castro, Kwanho Kim, Vincent Melfi, Gillian R. Hayes, Zhexue Huang, Alexey Tsymbal, An-Lei Hu, Qiyao Wang, Anupam Basu, Lauren Katzen,
Adrian Demaid, Ernesto Compatangelo, Dewang Chen, J. Dong, Kun Yue, Yaojin Lin, Roderic Leigh, Bert Bongers, Kuo-Yuan Kao, Amy K. Hurst,
Hannes Perkmann, Hongjun Lu, Hou-Yi Li, Yasutoshi Makino,

Joachim Lepping, Claus Pahl, Ralf Stadelhofer, Jiadao Li, Lena Wiese, Stefan Haustein, Timm Euler, Paul Lokuciejewski, Heike
Hunneshagen, Heiko Falk, Nanette Bauer, Karsten Klein, Martin Lang 0005, Haiseung Yoo, Thorsten Camps, Frank Weichert, Claudia Reuter,
Dirk D uuml;ding, Maria Kuhl, Morteza Monemizadeh, Baiyi Song, Uta Pankoke-Babatz, Carsten Witt, Thomas K ouml;nigsmann, Matthias
Hebbel, Alexander Klemm, Gerrit Bleumer, Tahir Ejaz, Ingo L uuml;ck, Konstantinos Kotsokalis, Stephan Lehmke, Gerrit Rothmaier, Jens
Niehaus, Kamol Limtanyakul, Christian Thyssen, Claudia Gsottberger, Christoph Jan Richter, Aleksandra Sowa, Martin Scholz, Harald
Gebhard, Peter Schramm, Tobias Marschall, Carsten Gutwenger, David Fiedler, Mohammad Yasser al-Nahlaoui, Stefan Schmermbeck, Klaus
Julisch, Patrick Ren eacute; Steve Piastowski, G uuml;nter Graw, Georgios Lajios, Ingo Dahm, Torben Weibert, Jens Busch, Mark Jung,
Boris Naujoks, D ouml;rte K. Rappe, Gila Brandt-Herrmann, Maria Kandyba-Chimani, Xiaolei Shi, Roman Klinger, Dominik G ouml;ddeke, J
ouml;rg Pleumann, Daniel Chernuchin, J uuml;rgen Kemper, Thomas Beielstein, Christian Brockmann, Marius Otte, Jens Wagner, Katharina
Hilker,

Toni Cortes, Mikael H ouml;gqvist,
Maria Sorea, Zhendong Ma, Pierre Bayerl,

Heiko Maus, Joachim Bayer, Tobias Schuele, Oliver Wirjadi, Peter Zeile, Jens Brandt, Sven Schwarz, Michael P. Haustein, Gerrit
Hanselmann, Ingmar Fliege, Ramon Serna Oliver, Achim Ebert, Mark M uuml;ller, Ivan Martinovic, Thomas Patzke, Martin Becker 0002, Jan
Olaf Blech, Georg Buscher, Klaudia Hergula, Joachim Thees, Marcus Trapp, Philipp Dopichaj, Jens Heidrich, Andreas M. Weiner, Mart
iacute;n Soto, Ingo Ginkel, Armin Stahl, R uuml;diger Ebendt, Sebastian Thelen, Manuel M ouml;ller, Andreas Jedlitschka, Mesut Ipek,
Stephan Baumann, Raik Brinkmann, Frank Michel, Ulrike Becker-Kornstaedt, Chenxi Qiu, Markus Nick, Matthias Gro szlig;, Gabriele Bleser,
Thomas Kilb, Patric Keller, Dirk Hamann, Christoph K ouml;gl, Joost van Beusekom, Akin Tanatmis, Ina Schaefer, Mario Trapp, Younis O.
Hijazi, Christoph Garth, Erwin Sitompul, Kizito Ssamula Mukasa, Christian Mathis, Mohammed Bani Younis, Ansgar Lamersdorf, Thomas
Kollig, Tanvir M. E. Hussain, R uuml;diger Grammes, Gustavo Nery, Jochen M uuml;ller 0002, Jean-Francois Girard, Alexander Geraldy,
Bernd G. Freimut, Helge Sch auml;fer, Jorge Rafael Vel aacute;squez Flores, Tobias Schmidt-Samoa, Michael M uuml;nchhofen, Jochem H
uuml;llen, Holger Diekmann, Jan Schaefer, Achim Reuther, Max Thalmaier, Hao Jiang, Sandra Zilles, Florian Gerhardt, Robert Kolter, Maja
Ruby, Jens Knodel, Sascha H. Schmitt, Eduard Deines, Tim Braun, Daniel Burkhart, Leonardo Ribeiro, Stefan Agne, Daniel G ouml;rlich,
Torsten Bierz, Matthias Priebe, Christian Webel, Leo Sauermann, Eros Comunello, Lars Geyer, Thorsten Keuler, Siana Halim, Norbert
Schmitz, Armin Hust, Robert Eschbach, Eric Ras, Ge Zhang, Philipp Schaible, Christian Denger, Thomas Kuhn, Markus Bon, Adrian Ulges,
Heinz Ulbricht, Petra Malik, Norbert G ouml;b, Alexis Ocampo, Gerrit Meixner, Jernej Kov eacute;se, Steffen Wolf, Robert Kalckl
ouml;sch, Jos eacute; de Agiuar Moraes Filho, Markus Hillenbrand, Andreas Morgenstern, Bernd L ouml;chner, Benedikte Elbel, Oliver R
uuml;bel, Duoli Qiu, Bernd Reuther, Isabel John, Michael Schlemmer, Harald Holz,

Tanja Paulitz, Roland Steidle,
```

CPM:

The **clique percolation method** is a popular approach for analyzing the overlapping community structure of networks.

The clique percolation method builds up the communities from k-cliques, which correspond to complete (fully connected) sub-graphs of k nodes. (E.g., a k-clique at $k = 3$ is equivalent to a triangle). Two k-cliques are considered adjacent if they share $k - 1$ nodes. A community is defined as the maximal union of k-cliques that can be reached from each other through a series of adjacent k-cliques. Such communities can be best interpreted with the help of a k-clique template (an object isomorphic to a complete graph of k nodes). Such a template can be placed onto any k-clique in the graph, and rolled to an adjacent k-clique by relocating one of its nodes and keeping its other $k - 1$ nodes fixed. Thus, the k-clique communities of a network are all those sub-graphs that can be fully explored by rolling a k-clique template in them, but cannot be left by this template.

This definition allows overlaps between the communities in a natural way, as illustrated in Fig.



showing four *k*-clique communities at $k = 4$. The communities are color-coded and the overlap between them is emphasized in red. The definition above is also local: if a certain sub-graph fulfills the criteria to be considered as a community, then it will remain a community independent of what happens to another part of the network far away. In contrast, when searching for the communities by optimizing with respect to a global quantity, a change far away in the network can reshape the communities in the unperturbed regions as well. Furthermore, it has been shown that global methods can suffer from a resolution limit problem, where the size of the smallest community that can be extracted is dependent on the system size. A local community definition such as here circumvents this problem automatically.

Time Complexity:

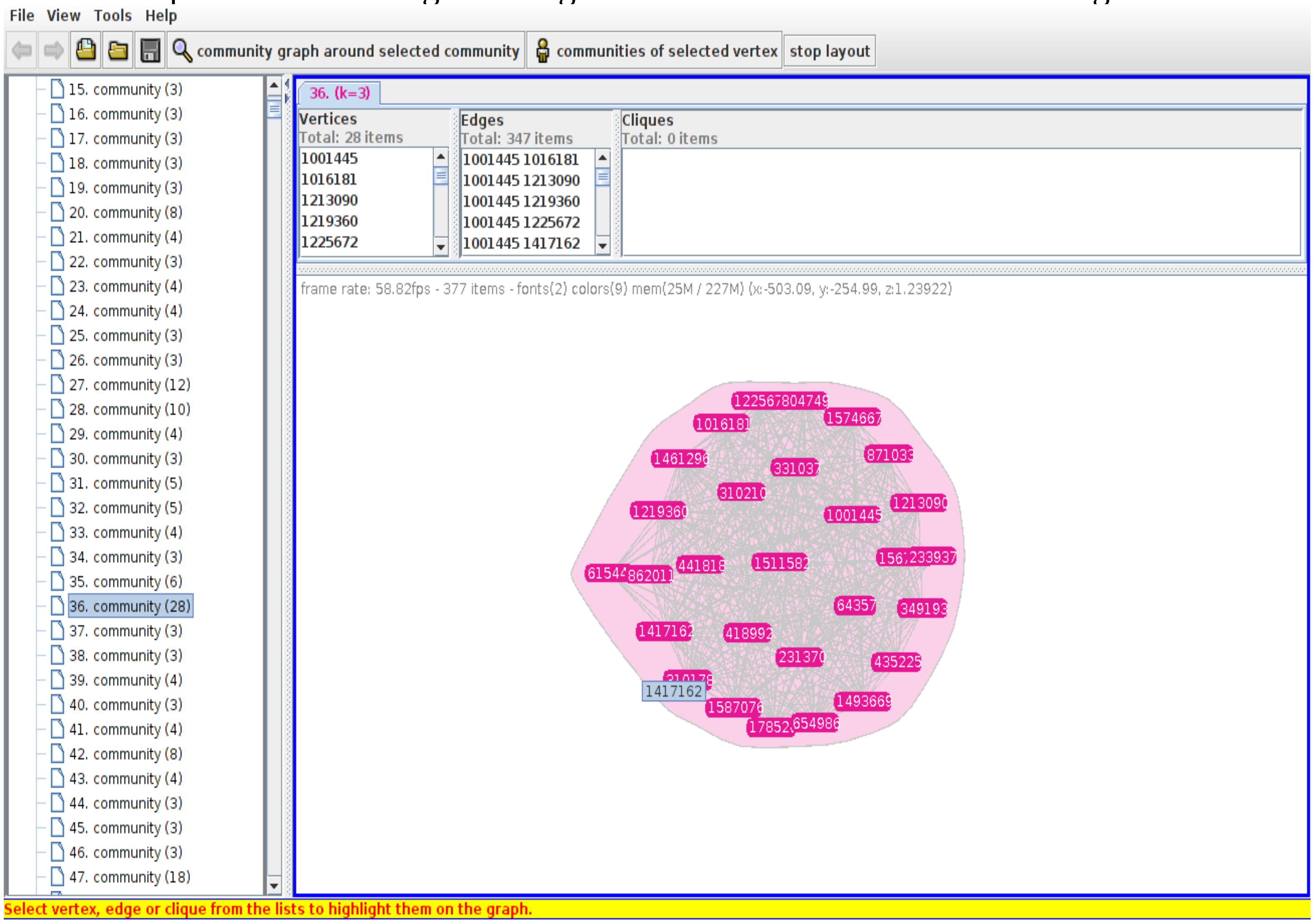
Since even small networks can contain a vast number of *k*-cliques, the implementation of this approach is based on locating the maximal cliques rather than the individual *k*-cliques.Thus, the complexity of this approach in practice is equivalent to that of the NP-complete maximal clique finding, (in spite that finding *k*-cliques is polynomial). This means that although networks with few million nodes have already been analyzed successfully with this approach,no prior estimate can be given for the runtime of the algorithm based simply on the system size.

Disadvantage:

It has several disadvantage as well:

1. Fail to give meaningful covers for graph with few cliques.
2. With too many cliques,might give a trivial community structure.
3. Subgraphs containing many cliques is equal to no of communi-ties
4. It is difficult to define value of k to give meaningful structure

snapshot after running CPM algo on the DBLP dataset with k=3 we get:

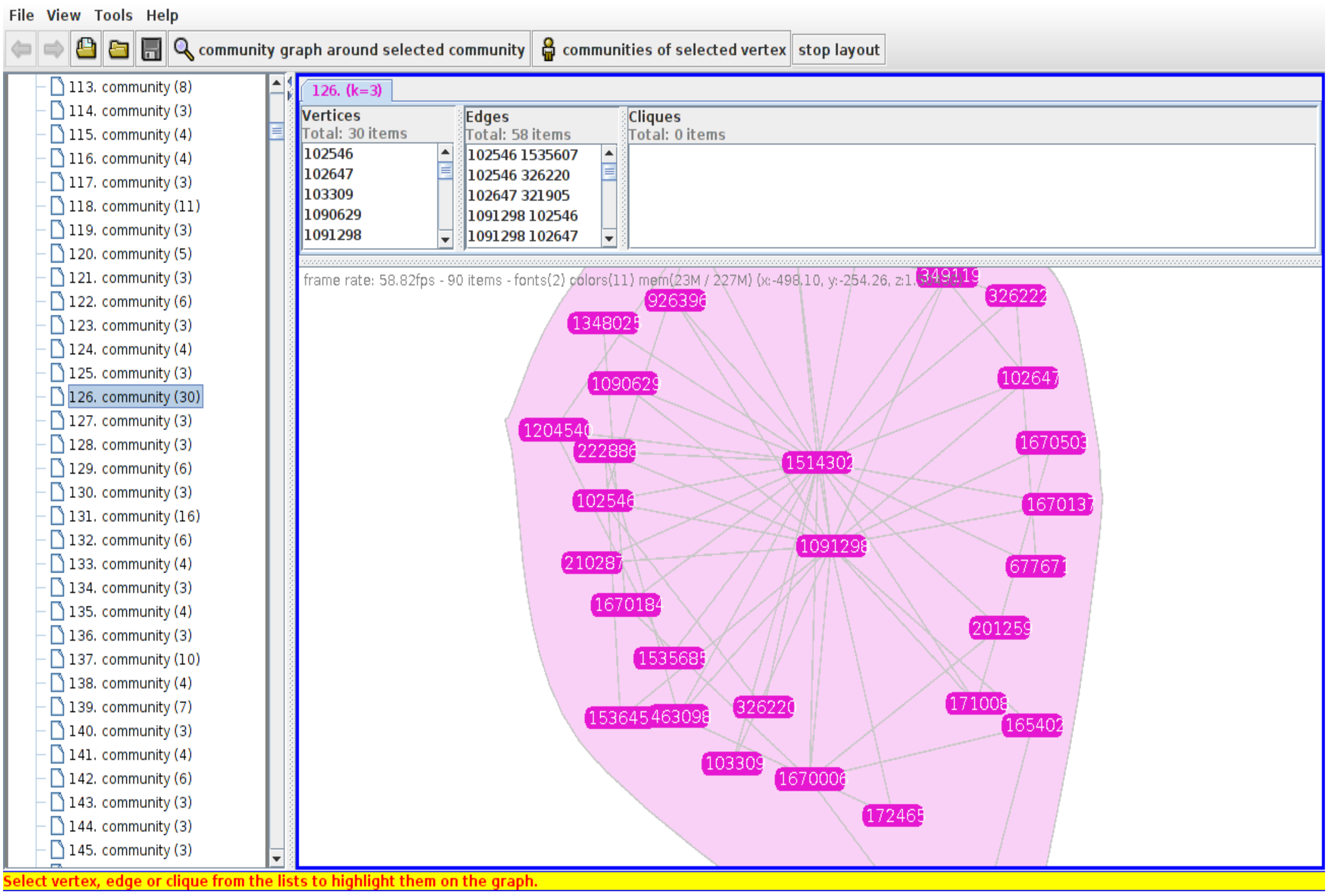


15. community (3)
16. community (3)
17. community (3)
18. community (3)
19. community (3)
20. community (8)
21. community (4)
22. community (3)
23. community (4)
24. community (4)
25. community (3)
26. community (3)
27. community (12)
28. community (10)
29. community (4)
30. community (3)
31. community (5)
32. community (5)
33. community (4)
34. community (3)
35. community (6)
36. community (28)
37. community (3)
38. community (3)
39. community (4)
40. community (3)
41. community (4)
42. community (8)
43. community (4)
44. community (3)
45. community (3)
46. community (3)
47. community (18)

Vertices	Edges	Cliques
Total: 28 items	Total: 347 items	Total: 0 items
1001445	1001445 1016181	
1016181	1001445 1213090	
1213090	1001445 1219360	
1219360	1001445 1225672	
1225672	1001445 1417162	

frame rate: 58.82fps - 377 items - fonts(2) colors(9) mem(25M / 227M) (x:-503.09, y:-254.99, z:1.23922)

Select vertex, edge or clique from the lists to highlight them on the graph.


File View Tools Help

community graph around selected community | communities of selected vertex | stop layout

113. community (8)
114. community (3)
115. community (4)
116. community (4)
117. community (3)
118. community (11)
119. community (3)
120. community (5)
121. community (3)
122. community (6)
123. community (3)
124. community (4)
125. community (3)
126. community (30)
127. community (3)
128. community (3)
129. community (6)
130. community (3)
131. community (16)
132. community (6)
133. community (4)
134. community (3)
135. community (4)
136. community (3)
137. community (10)
138. community (4)
139. community (7)
140. community (3)
141. community (4)
142. community (6)
143. community (3)
144. community (3)
145. community (3)

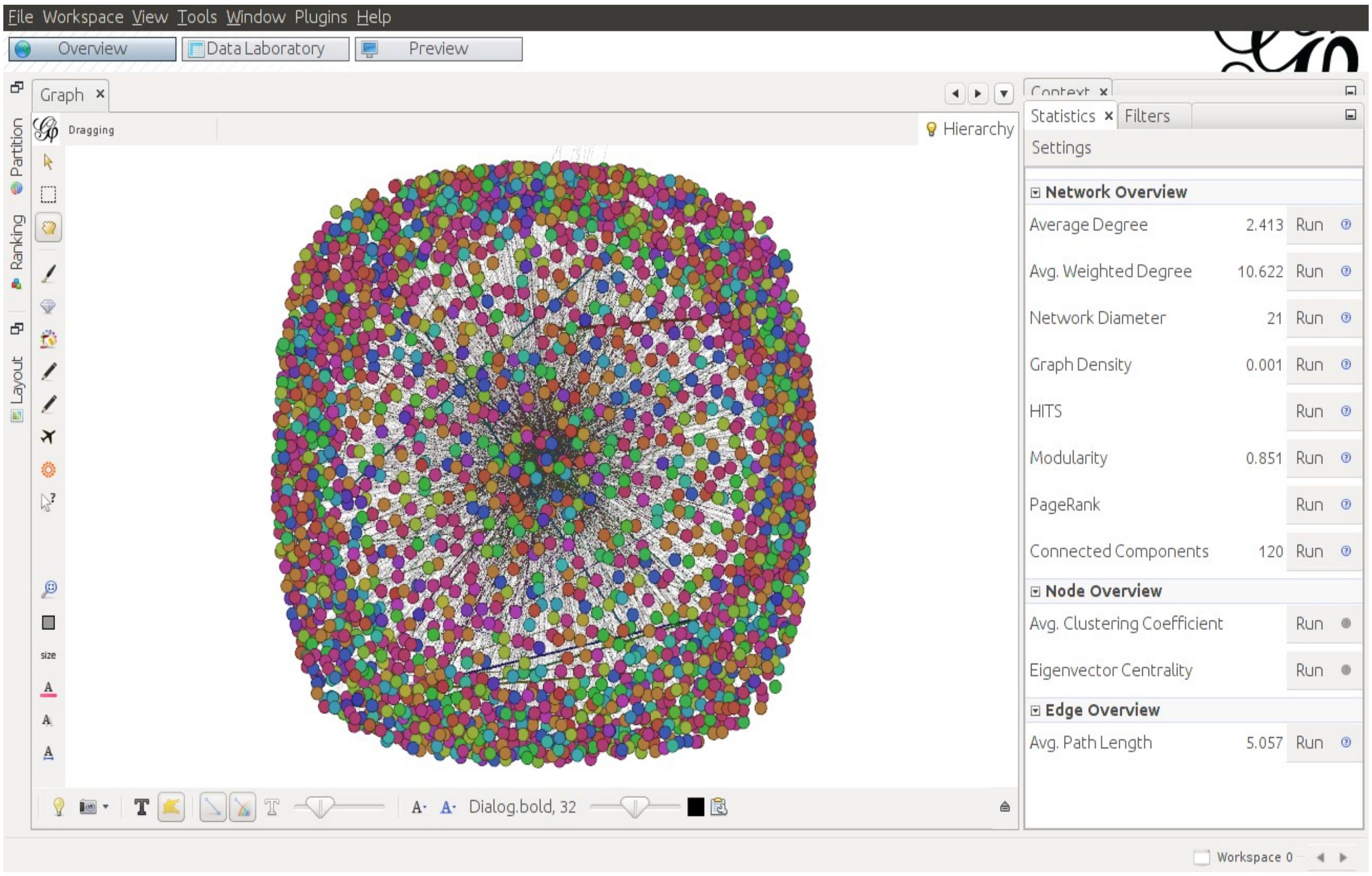
126. (k=3)

Vertices Total: 30 items	Edges Total: 58 items	Cliques Total: 0 items
102546	102546 1535607	
102647	102546 326220	
103309	102647 321905	
1090629	1091298 102546	
1091298	1091298 102647	

frame rate: 58.82fps - 90 items - fonts(2) colors(11) mem(23M / 227M) (x:-498.10, y:-254.26, z:1.

Select vertex, edge or clique from the lists to highlight them on the graph.

Modularity calculated using **Gephi is 0.851.**
Snapshot of the result is:



Network Overview

Average Degree	2.413	Run
Avg. Weighted Degree	10.622	Run
Network Diameter	21	Run
Graph Density	0.001	Run
HITS		Run
Modularity	0.851	Run
PageRank		Run
Connected Components	120	Run

Node Overview

Avg. Clustering Coefficient		Run
Eigenvector Centrality		Run

Edge Overview

Avg. Path Length	5.057	Run

Where nodes with same colour represents authors in the same community.

Major Challenges:

1. **Noisy Data:** At some places name of the authors contained special character that need to be removed.Also same authors may have different name in his different publications.We need to remove that.
2. **Using Networkx Library:**NetworkX doesn't have pre-compiled C libraries, which makes it quite slow on larger graphs. In fact, some of the simple link prediction algorithms that we tried to work with - Jaccard similarity, negated shortest path - required about a day for complete execution
3. **Lack of computation Power:**One of the major issues we faced was that we were unable to to execute all the steps of our algorithm quickly. So, if we changed the first part of the algorithm, it could take up to several hours to finish execution. It also made it difficult to work on much larger datasets of authors, which could have improved our community detection accuracy Girvan-Newman community

detection,Louvain,CPM were some of the algorithms that took huge amount of time due to the lack of processing power.

4. **Parsing dataset:** There were different tags in the corpus,also the size of the corpus was large.So it took long time to parse tags as we need to process whole corpus to retrieve some information.The DBLP corpus uses the ISO-8859-1 encoding and thus it was difficult to handle character such as ampersand(&). We tried changing the encoding to Unicode but it turned out that it couldn't be modified.

Conclusion:

In this project we learnt about Community Detection and the various algorithms which can be used for it. These algorithms can also be extended to finding communities in other types of networks mainly social networks. We also got to know about a few methods useful for computing text similarity. Furthermore, the project taught us a lot about team work and the importance of efficient communication.

Useful Links

Slideshare Link : https://www.slideshare.net/secret/xbEB3MnI9RcLkb

Github Project Page : http://modestboy.github.io/ire_major_team31/

Github Repository : https://github.com/modestboy/ire_major_team31

Youtube Video : https://youtu.be/WvOVSEoqmOE

Dropbox : https://www.dropbox.com/sh/9vfun1t2wbsn09k/AADYbwmYJP7smj7fnDtqG76sa?dl=0

References

[1.] https://perso.uc**louvain**.be/vincent.blondel/research/**louvain**.html

[2.] http://www.ismll.uni-hildesheim.de/lehre/cmie-11w/script/lecture5.pdf

[3.] www.comp.nus.edu.sg/.../W11-Eugene-**Clique-Percolation-Method**.pptx

[4.] www.github.com/gephi/gephi/releases

[5.] www.**dblp**.uni-trier.de/

[6.] https://docs.python.org/2/library/**difflib**.html

[7.] https://www.youtube.com/watch?v=lU1QEUH0nNc&list=PLriUvS7IljvkBLqU4nPOZtAkp7rgpxjg1&index=9