

## Online clustering with CFinder

András Barta · Gergely Palla · Péter  
Pollner · Tamás Vicsek

Received: date / Accepted: date

**Abstract** Due to the constant increase of the complexity and size of the complex networks studied, the need for distributed implementations of network clustering methods is getting more urgent each year. Here we present the online version of CFinder, which can locate overlapping communities in directed, weighted or undirected networks based on the clique percolation method (CPM). Due to the local nature of the CPM, the computation can be distributed among several CPU-s or computers. This implementation uses an enhanced, grid based computational backend and provides resources for the CPU and RAM intensive tasks of scientific research.

**Keywords** Networks · Clustering · Grid Computing

### 1 Introduction

Many complex systems in nature and society can be successfully represented in terms of networks capturing the intricate web of connections among the units they are made of [1]. In recent years, several large-scale properties of real-world webs have been uncovered, e.g. a low average distance combined with a high average clustering coefficient[2], the broad (scale-free) distribution of node degree (number of links of a node) [3,4] and various signatures of hierarchical and modular organization[5].

Beside the mentioned global characteristics, there has been a quickly growing interest in the local structural units of networks as well. Small and well defined sub-graphs consisting of a few vertices have been introduced as motifs[6], whereas somewhat larger

---

András Barta  
Dept. of Biological Physics, Eötvös University, 1117 Budapest, Pázmány s 1/A, Hungary

Gergely Palla  
Statistical and Biological Physics Res. Grp. of HAS, 1117 Budapest, Pázmány s 1/A, Hungary

Peter Pollner  
Statistical and Biological Physics Res. Grp. of HAS, 1117 Budapest, Pázmány s 1/A, Hungary  
E-mail: pollner@hal.elte.hu

Tamas Vicsek  
Statistical and Biological Physics Res. Grp. of HAS 1117 Budapest, Pázmány s 1/A, Hungary  
and Dept. of Biological Physics, Eötvös University 1117 Budapest, Pázmány s 1/A, Hungary

units, associated with more highly interconnected parts are usually called *communities*, clusters, cohesive groups, or modules[7–10]. These structural sub-units can correspond to multi-protein functional units in molecular biology[5,11], a set of tightly coupled stocks or industrial sectors in economy[12], groups of people[13], cooperative players[14], etc. The location of such building blocks can be crucial to the understanding of the structural and functional properties of the systems under investigation.

The complexity and the size of the investigated data sets are increasing every year. Therefore, the hardware resources of every-day-usage computers are not always enough to perform the computation needed for analysis of the structures in the studied systems. Centralized mainframe computers, where such computations can be done, are a possible solutions in these cases. Along this line, here we introduce the online version of CFinder[15], suitable for finding and visualizing overlapping clusters in large networks. This application is based on the earlier, stand-alone version of CFinder, which turned out to be a quite popular network clustering program. The advantage of the online version is that it can help users lacking enough computer resources for their task.

The paper is organized as follows. In Section 2 we give a summary of the Clique Percolation Method (CPM). The Section 3 presents the implementation details, how we distribute the computation among several CPUs or PCs. The last Section encloses our concluding remarks.

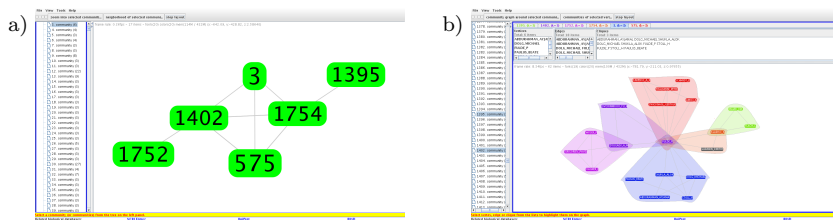
## 2 The CPM method

Communities are usually defined as dense parts of the studied networks, and the majority of the community finding approaches separate these regions from each other by a relatively small number of links. However, in reality communities may even overlap as well, (in this case the nodes in the overlap are members of more than one communities). A recently introduced, link density-based community finding technique allowing community overlaps is given by the CPM.

In this approach a community is built up from adjacent blocks of the same size  $k$ . These blocks correspond to  $k$ -cliques, that are subgraphs with the highest possible density: each of the  $k$  members of the  $k$ -clique is linked to every other member. Two blocks are considered adjacent if they overlap with each other as strongly as possible, i.e., they share  $k - 1$  nodes. (Note that removing one link from a  $k$ -clique leads to two adjacent  $k - 1$ -cliques sharing  $k - 2$  nodes).

A community is a set of blocks that can be reached from one to the other through a sequence of adjacent blocks. Note, that any block belongs always to exactly one community, however, there can be nodes belonging to several communities at the same time, (e.g., if blocks overlap only in a single node). A consequence of the above definition is that the communities contain only nodes that are densely connected. Nodes with only a few connections or nodes that do not participate in a densely connected subgraph are not classified into any community.

The CPM method is robust against removal or insertion of a single link. Due to the local nature of this approach, such perturbations can alter only the communities containing at least one of the end points of the link. (In contrast, for global methods that optimize a homogeneously defined quantity the removal or insertion of a single link can result in the change of the overall community structure.)



**Fig. 1** a) The network of communities found by CPM algorithm in the cond-mat coauthorship data set ( $k = 3$ ). b) The zoom view of the community network, where individual nodes are visible. The Java based visualization program maps the communities or the connections between communities (network of communities) in a similar way as the stand-alone CFinder.

Finally, we note that the CPM will find the same communities in a given subgraph irrespective to the fact whether the subgraph is linked to a larger network or not. Therefore, a heterogeneous network can be analyzed by first dividing it into homogeneous parts, and applying the method to these subnetworks separately.

### 3 Web interface and the grid version of CPM

A server based web-application for locating overlapping communities of densely interconnected nodes is provided by WebCFinder[15]. This application is meant as an additional tool to supplement the existing stand-alone CFinder program, as it can be employed by potential users lacking enough computer resources for their task (e.g., not having enough RAM in their PC). Furthermore, WebCFinder is also ideal for users wanting to test our method without downloading or installing any software.

#### 3.1 Web interface

Stand-alone, generic graph visualization and analysis programs ([16]) are frequently used for the layout and structural analysis of networks. Recent software platforms([17]), on the other hand, enable the user to integrate many different types of data, e.g., PPI, expression levels and annotation information. These softwares can work either with local files or with remote databases. In any case, however, they need to be installed by the user, and the analysis is limited by the computational power and the amount of available main memory. In contrast, WebCFinder[15] needs only some standard programs to be available on the users computer and an Internet connection from where our server can be reached. Our program reads a list of binary interactions uploaded by the user, performs a search for dense subgraphs (groups), and provides an URL where users can download the results.

The *input of WebCFinder* is a file containing strings and numbers ordered into three columns; in each row the first two strings correspond to the two end points of a link and the third item is the weight of this link. The input file can be uploaded to our server after a login process. The web interface is designed for graphical browsers and for text-only browsers as well. The text-only interface mainly serves computational backend for a text mining and trend analysis toolkit[18], but a simplified form is publicly available.

The computational core of WebCFinder was implemented in C++, while the visualization components were written in Java and work as a webstart application at

the client side. The *search algorithm* can use three version of the Clique Percolation Method (undirected CPM, directed CPM and weighted CPM see [8,19,20]) to locate the *k-clique percolation clusters* of the network that we interpret as communities.

Note that larger values of  $k$  correspond to a higher stringency during the identification of dense groups and provide smaller groups with a higher density of links inside them. For both local and global analyzes in a network, we suggest using such a value of  $k$  that provides the user with the richest group structure. In the presence of link weights WebCFinder can apply lower and upper cutoff values to keep only the set of connections judged to be significant by the user. The cutoff values, the appropriate clustering algorithm and the  $k$  value (which is important only in the case of the weighted version of CPM) are changeable on a configuration page.

The resulting set of data can be downloaded in two formats. The user may start the visualization program directly in their web browser, and examine the results of the clustering process. The other option is to download the results in a compressed file that can be later analyzed or can be visualized by the stand-alone CFinder program.

The *graphical visualization interface* of WebCFinder offers several views of the analyzed network and its module structure. (see Fig.1) Alternative views currently available are “Communities” (displaying the identified modules), “Cliques”, “Stats” (statistics of, e.g., module and overlap sizes) and “Graph of communities”. A wide variety of visualization settings can be adjusted in the “Tools” menu.

### 3.2 Computation engine

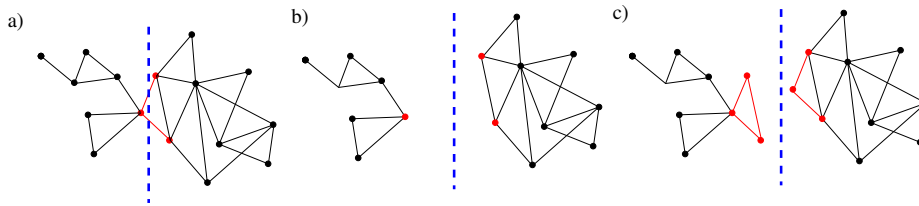
The search engine of WebCFinder is built up from two parts: one subprocess prepares the network for distributed processing and collects the results, the other subprocess performs the clustering (in a distributed way if needed). As mentioned in the previous section, the CPM method is a local method, therefore, the network can be split up into small pieces that fit into the memory, find the communities in each parts, and finally merge the results.

When locating the communities in a distributed way, the splitting of the analyzed network has to satisfy the following conditions:

- each part must be small enough so that the clustering process can fit into the memory of the computer, where the given subnetwork is processed,
- in case the splitting might divide the (to be located) communities as well, the split borders have to appear in both subnetworks, so that the global community structure can be reconstructed from the local communities of the subnetworks, (see Fig.2.)
- the computational overhead due to the overlapping subnetworks has to be as small as possible

The first and the third condition are contradictory: if one optimizes for CPU time, than the overlaps must be minimal. But if one optimizes for memory usage on a single processing host, the network has to be split into numerous tiny subnetworks. More subnetworks, however, contain more overlaps.

To decide which condition to favor at the expense of the other let us consider the following question. Is it easier to add more CPUs to an existing cluster, or is it easier to add more RAM to every computer where we plan to run our software? Today CPU is relatively cheap: either you can buy new nodes or you can use one of the existing



**Fig. 2** Splitting of a network in two pieces before the exploration of the communities. a) The original network: the links highlighted in red are selected as cut-links, their end-nodes define the boundary region between the two pieces. b) After the link removal the two separate subnetworks. c) We re-insert the boundary region into each subnetwork separately. This way the  $k$ -cliques of the boundary region will be found in both pieces, enabling the reconstruction of the original communities.

computer clouds. Therefore, CPU usage is scalable up to a very large amount. Although RAM is getting cheap as well, the size of the RAM modules that can be built into to the computers has an upper limit. This means that the size of the available RAM is scalable only in a limited range and adding further RAM modules above this limit becomes very expensive or even impossible. In conclusion, cutting the network into many small subnets seems to be the better solution.

The second condition, which requires the ability to reconstruct the global community structure from the locally found communities (and community parts) is satisfied as described in the following. For simplicity let us suppose that we would like to split the investigated network into two parts, as shown in Fig.2. First we select a set of links (indicated in red), whose removal cuts the network unto two unconnected subnetworks. The end-nodes of these links define the boundary region of the subnetworks. We split the network into two pieces by removing the selected links, and for each subnetwork we separately insert back all nodes and links in the boundary region, (including links between boundary nodes that were not cut-links), which means that the boundary region is duplicated. As a result, the  $k$ -cliques located in the boundary region of the original network will appear in both subnetworks, thus, the communities found in the two pieces will overlap in these  $k$ -cliques, enabling the reconstruction of the original communities.

The resulting unconnected graphs can be clustered independently, therefore, the calculation can be distributed among several computers. The distributed computation is aided by the Condor Queueing System[21], so we can submit as many processes as we like. The queue system distributes the jobs among free processors and computers we have access to. The results of every subprocess are sent to our main server, where a database engine collects the results. If every subprocess is ready and we have the communities for every parts of the network, the communities are merged together within the database. So the size of the processable network is bounded not by the available RAM of our server but by the available disk space, which is measured in terabytes today.

## 4 Conclusion

We have presented a new implementation of the CFinder algorithm. We have shown, that due to the local nature of the underlying clique percolation method[8], the com-

putation can be distributed among several CPU-s of computers. The presented implementation uses an enhanced, grid based computational backend. This provides the possibility to perform CPU and RAM intensive tasks, that were not solvable on single desktop personal computers.

**Acknowledgements** This work was supported by the Hungarian National Science Fund (OTKA K68669 and T049674), the National Research and Technological Office (NKFP.07\_A2 (2007)/TEXTREND) and the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

## References

1. Albert R and Barabási A-L, Statistical mechanics of complex networks, *Rev. Mod. Phys.*, 74, 47-97 (2002)
2. D. J. Watts and S. H. Strogatz, Collective dynamics of 'small-world' networks, *Nature*, 393, 440-442, (1998)
3. A.-L. Barabási and R. Albert, Emergence of scaling in random networks, *Science*, 286, 509-512, (1999)
4. S. Boccaletti and V. Latora and Y. Moreno and M. Chavez and D.-U. Hwang, Complex networks: Structure and dynamics, *Physics Reports*, 424, 175-308, (2006)
5. E. Ravasz and A. L. Somera and D. A. Mongru and Z. N. Oltvai and A.-L. Barabási, Hierarchical organization of modularity in metabolic networks, *Science*, 297, 1551-1555, (2002)
6. R. Milošević and S. Shen-Orr and S. Itzkovitz and N. Kashtan and D. Chklovskii and U. Alon, Network Motifs: Simple Building Blocks of Complex Networks, *Science*, 298, 824-827, (2002)
7. Reichardt J and Bornholdt S, Detecting fuzzy community structures in complex networks with a Potts model, *Phys. Rev. Lett.*, 93, 218701, (2004)
8. Palla G and Derényi I and Farkas I and Vicsek T, Uncovering the overlapping community structure of complex networks in nature and society, *Nature*, 435, 814-818, (2005)
9. M. Blatt and S. Wiseman and E. Domany, Super-paramagnetic clustering of data, *Phys. Rev. Lett.*, 76, 3251-3254, (1996)
10. M. Girvan and M. E. J. Newman, Community structure in social and biological networks, *Proc. Natl. Acad. Sci. USA*, 99, 7821-7826, (2002)
11. V. Spirin and K. A. Mirny, Protein complexes and functional modules in molecular networks, *Proc. Natl. Acad. Sci. USA*, 100, 12123-12128, (2003)
12. T. Heimo and J. Saramäki and J.-P. Onnela and K. Kaski, Spectral and network methods in the analysis of correlation matrices of stock returns, *Physica A-Statistical Mechanics and its Applications*, 383, 147-151, (2007)
13. J. Scott, *Social Network Analysis: A Handbook*, Sage Publications, London, (2000)
14. G. Szabó and G. Fáth, Evolutionary games on graphs, *Physics Reports-Review Section of Physics Letters*, 446, 97-216, (2007)
15. <http://www.cfinder.org>
16. Batagelj, V. and Mrvar, A., Pajek - program for large network analysis, *Connections*, 21, 47-57, (1998)
17. Shannon, P et al., Cytoscape: A software environment for integrated models of biomolecular interaction networks, *Genome Research*, 13, 2498-2504, (2003)
18. <http://www.textrend.org>
19. G. Palla, I. J. Farkas, P. Pollner, I. Derényi, T. Vicsek, Directed network modules, *New J. Phys.*, 9, 186, (2007)
20. I. J. Farkas, D. Ábel, G. Palla, T. Vicsek, Weighted network modules, *New J. Phys.*, 9, 180, (2007)
21. Douglas Thain, Todd Tannenbaum, and Miron Livny, "Distributed Computing in Practice: The Condor Experience" *Concurrency and Computation: Practice and Experience*, Vol. 17, No. 2-4, pages 323-356, February-April, 2005.