Structural Cohesion: Visualization and Heuristics for Fast Computation with NetworkX and matplotlib

Jordi Torrents[§*], Fabrizio Ferraro[‡]

https://www.youtube.com/watch?v=K8RFIdG3g9Y

Abstract—The structural cohesion model is a powerful sociological conception of cohesion in social groups, but its diffusion in empirical literature has been hampered by computational problems. We present useful heuristics for computing structural cohesion that allow a speed-up of one order of magnitude over the algorithms currently available. Both the heuristics and the exact algorithm have been implemented on NetworkX by the first author. Using as examples three large collaboration networks (co-maintenance of Debian packages, co-authorship in Nuclear Theory, and co-authorship in High-Energy Theory) we illustrate our approach to measure structural cohesion in relatively large networks. We also introduce a novel graphical representation of the structural cohesion analysis to quickly spot differences across networks. It is implemented using matplotlib.

Index Terms—Network Analysis, Sociology, Structural Cohesion, NetworkX, matplotlib

Introduction

Group cohesion is a central concept that has a long and illustrious history in sociology and organization theory, although its precise characterization has remained elusive. Its use in most sociological research has been ambiguous at best. This is largely because, as [moody2003] argued, it is often based on sloppy definitions of cohesion, grounded mostly in intuition and common sense. Network analysis has provided a large number of solutions to this problem. From classical work in the graph-theoretic sociological tradition on cliques, clans, clubs, *k*-plexes, *k*-cores and lambda sets [wasserman1994], to the more recent contribution of physicists and computer scientists on community analysis [fortunato2010], network theorists have provided researchers with a wide range of measures of cohesion in social networks.

However, neither the classical approaches nor new developments in community analysis are well-enough suited to address many of the common uses of group cohesion in the sociological literature, and thus fall short when used in empirical analysis. The structural cohesion model ([white2001], [moody2003]) has strong mathematical foundations, and captures many of the features of the concept group cohesion. Despite this, it has not been widely used in empirical analysis because it is not possible to perform the required computations for networks with more than a few thousands nodes and edges in a reasonable time frame. Moreover, there are very few implementations available to researchers.

In this paper we present a set of heuristics to compute the connectivity structure of a given network. We implemented them, along with the exact algorithm, on top of NetworkX [hagberg2008], a Python Library for Network Analysis. We also suggest a novel graphical representation of the results, implemented using Matplotlib [hunter2007]. The rest of the paper is organized as follows: we start by discussing the main features which a cohesive subgroup formalization should have from a sociological perspective, and then discuss in depth the structural cohesion model. We then describe the exact algorithm and introduce our proposed heuristics. We go on to report our findings from applying the structural cohesion analysis to three large collaboration networks, which allows us to illustrate the novel graphical representation of the connectivity structure. Finally we conclude with some implications for future research.

Cohesion in social networks

[doreian1998] argue that group cohesion can be divided analytically into an *ideational* component, which is based on the members' identification with a collectivity, and a *relational* component, which is based on connections among members. These connections are, at least in part, observable, and thus the relational approach seems more appropriate for theory building and empirical research. But, despite its attractiveness, the relational component has received much less attention than the ideational component in sociological literature. Social network analysis has been the exception, and since the beginning, its proponents formalized group cohesion in relational terms, that is, they defined the boundaries of subgroups in a community starting from the patterns of relations among actors.

Unfortunately most of the existing formalizations of cohesive subgroups do not capture some key properties of the concept of cohesive groups. First, a cohesive subgroup should be *robust*, in the sense that its qualification as a group should not be dependent on the actions of a single individual, or any small set of individuals that belong to the group. This implies, on the one hand, that no actor, or small set of actors, should be able to dissolve the cohesive subgroup by abandoning it; while, on the other hand, all actors in a group should be related to all other actors by multiple direct or indirect connections in order to pull it together [moody2003]. Therefore, cohesive subgroups should also be relatively invariant to changes outside the group [brandes2005].

* Corresponding author: jordi.t21@gmail.com
§ University of Barcelona
‡ IESE Business School

Second, actual social groups tend to *overlap* in the sense that some actors are likely to be part of more than one cohesive subgroup. As [freeman1992] notes, formalizations of subgroups that overlap a lot are not well suited to capturing the concept of groups because their sociological use is not focused on individuals but on contexts, such as productive relations, friendship relations, or family ties, to name a few. Thus if groups are defined around a highly specific context the overlap is likely to be small. Therefore the formalization of subgroups often assumed non-overlapping subgroups. However, there is always overlap among cohesive subgroups in actual social groups; and this overlap might be both empirically and theoretically relevant.

Third, following a typical distinction in the social network literature, cohesive groups have both a *structural* and a *positional* dimension. In the former, cohesive subgroups are defined in terms of the global patterns of relations, and the focus is on the groups and the network as a whole. In the latter, the focus is on the identification of actors who, because of their network position, obtain preferential access to information or resources that flow through the network. Cohesive subgroup formalizations should help address both structural and positional questions.

Last but by no means least, cohesive subgroups are likely to display a *hierarchical structure* in the sense that highly cohesive subgroups are nested inside less cohesive ones. This notion of hierarchy is grounded on Simon's definition: *a system that is composed of interrelated subsystems, each of the latter being, in turn, hierarchic in structure until we reach some lowest level of elementary subsystem* [simon1962]. A hierarchical conception of cohesive subgroups implies that there is a relevant organization at all scales of the network, and that cohesive groups are a mesolevel structure that is not reducible to neither macro nor micro level phenomena and dynamics.

The structural cohesion model

Structural cohesion is a powerful explanatory factor for a wide variety of interesting empirical social phenomena. It can be used to explain, for instance: the likelihood of building alliances and partnerships among biotech firms [powell2005]; how positions in the connectivity structure of the Indian inter-organizational ownership network are associated with demographic features (age and industry); and differences in the extent to which firms engage in multiplex and high-value exchanges [mani2014]. Social cohesion can also help us understand degrees of school attachment and academic performance in young people, as well as the tendency of firms to enroll in similar political activity behaviors [moody2003]. It offers insight, also, into emerging trust relations among neighborhood residents or the hiring relations among top level US graduate programs [grannis2009]. In addition to social solidarity and group cohesion, the model can equally fit many relevant theoretical issues, such as conceptualizing structural differences among fields and organizations [white2004], explaining the role of highly connected subgroups in boosting diffusion in social networks without a high rate of decay [moody2004], or highlighting the complexity and diversity of the structure of real world markets beyond stylized one-dimensional characterizations of the market [mani2014].

The structural cohesion approach to subgroup cohesion ([white2001], [moody2003]) is grounded on two mathematically equivalent definitions of cohesion that are based on commonly used concepts of cohesion in the sociological literature. On the one hand, the ability of a collectivity to hold together independently of the will of any individual. As set out by the formal definition, *a group's structural cohesion is equal to the minimum number of actors who, if removed from the group, would disconnect the group*. Yet, on the other hand, a cohesive group has multiple independent relational paths among all pairs of members. According to the formal definition *a group's structural cohesion is equal to the minimum number of independent paths linking each pair of actors in the group* [moody2003]. These two definitions are mathematically equivalent in terms of the graph theoretic concept of node connectivity[1] as defined by Menger's Theorem [white2001], which can be formulated locally: *The minimum node cut set* $\kappa(u,v)$ *separating a nonadjacent* u,v *pair of nodes equals the maximum number of node-independent* $u-v$ *paths*; and globally: *A graph is k-connected if and only if any pair of nodes* u,v *is joined by at least k node-independent* $u-v$ *paths*. Thus Menger's theorem links with an equivalence relation the connectivity based on cut sets with the number of node independent paths among pairs of different nodes. This equivalence relation has a deep sociological meaning because it allows for the definition of structural cohesion in terms of the difficulty to pull a group apart by removing actors and, at the same time, in terms of multiple relations between actors that keep a group together.

The starting point of cohesion in a social group is a state where every actor can reach every other actor through at least one relational path. The emergence of a giant component --a large set of nodes in a network that have at least one path that links any two nodes-- is a minimal condition for the development of group cohesion and social solidarity. [moody2003] argue that, in this situation, the removal of only one node can affect the flow of knowledge, information and resources in a network because there is only one single path that links some parts of the network. Thus, if a network has actors who are articulation points[2], their role in keeping the network together is critical; and by extension the network can be disconnected by removing them. [moody2003] convincingly argue that biconnectivity provides a baseline threshold for strong structural cohesion in a network because its cohesion does not depend on the presence of any individual actor and the flow of information or resources does not need to pass through a single point to reach any part of the network. Therefore, the concept of robustness is at the core of the structural cohesion approach to subgroup cohesion.

Note that the bicomponent structure of a graph is an exact partition of its edges, which means that each edge belongs to one, and only one, bicomponent; but this is not the case for nodes because k-components can overlap in $k-1$ nodes. In the case of bicomponents, articulation points belong to all bicomponents that they separate. Thus, this formalization of subgroup cohesion allows limited horizontal overlapping over k-components of the same k. On the other hand, the k-component structure of a network is inherently hierarchical because k-components are nested in terms of connectivity: a connected graph can contain several 2-components, each of which can contain one or more tricomponents, and so forth.

However, one shortcoming of classifying cohesive subgroups only in terms of node connectivity is that k-components of the same k are always considered equally cohesive despite the fact

1. See http://networkx.readthedocs.org/en/latest/reference/generated/networkx.algorithms.connectivity.connectivity.node_connectivity.html .

2. See http://networkx.readthedocs.org/en/latest/reference/generated/networkx.algorithms.components.biconnected.articulation_points.html .

that one of them might be very close to the next connectivity level, while the other might barely qualify as a component of level k (i.e. removing a few edges could reduce the connectivity level to $k-1$). To deal with this shortcoming, we propose using another connectivity-based metric to obtain a continuous and more granular measure of cohesion. [beineke2002] propose the measure of *average node connectivity* of G[3], denoted $\bar{\kappa}(G)$, defined as the sum of local node connectivity between all pairs of different nodes of G divided by the number of distinct pairs of nodes. Or put more formally:

$$\bar{\kappa}(G) = \frac{\sum_{u,v} \kappa_G(u,v)}{\binom{n}{2}}$$

Where n is the number of nodes of G. In contrast to node connectivity κ, which is the minimum number of nodes whose removal disconnects some pairs of nodes, the average connectivity $\bar{\kappa}(G)$ is the expected minimal number of nodes that must be removed in order to disconnect an arbitrary pair of nodes of G. For any graph G it holds that $\bar{\kappa}(G) \geq \kappa(G)$. As [beineke2002] show, average connectivity does not increase only with the increase in the number of edges: graphs with the same number of nodes and edges, and the same degree for each node can have different average connectivity.

Despite all its merits, the structural cohesion model has not been widely applied to empirical analysis because it is not practical to compute it for networks with more than a few thousands nodes and edges due to its computational complexity. What's more, it is not implemented in most popular network analysis software packages. In the next section, we will review the existing algorithm to compute the k-component structure for a given network, before introducing our heuristics to speed up the computation.

Existing algorithms for computing k-component structure

[moody2003] provide an algorithm for identifying k-components in a network[4], which is based on the [kanevsky1993] algorithm for finding all minimum-size node cut-sets of a graph[5]; i.e. the set (or sets) of nodes of cardinality k that, if removed, would break the network into more connected components. The algorithm consists of 4 steps:

1) Identify the node connectivity, k, of the input graph using flow-based connectivity algorithms.
2) Identify all k-cutsets at the current level of connectivity using the Kanevsky's algorithm.
3) Generate new graph components based on the removal of these cutsets (nodes in the cutset belong to both sides of the induced cut).
4) If the graph is neither complete nor trivial, return to 1; otherwise end.

As the authors note, one of the main strengths of the structural cohesion approach is that it is theoretically applicable to both small and large groups, which contrasts with the historical focus of the literature on small groups when dealing with cohesion. But the fact that this concept and the algorithm proposed by the authors, are theoretically applicable to large groups does not mean that this would be a practical approach for analyzing the structural cohesion on large social networks.

The equivalence relation established by Menger's theorem between node cut sets and node independent paths can be useful to compute connectivity in practical cases but both measures are almost equally hard to compute if we want an exact solution. However, [white2001b] proposed a fast approximation algorithm for finding good lower bounds of the number of node independent paths between two nodes[6]. This algorithm is based on the idea of searching paths between two nodes, marking the nodes of the path as *used* and searching for more paths that do not include nodes already marked. But instead of trying all possible paths without order, this algorithm considers only the shortest paths: it finds node independent paths between two nodes by computing their shortest path, marking the nodes of the path found as *used* and then searching other shortest paths excluding the nodes marked as *used* until no more paths exist. Because finding the shortest paths is faster than finding other kinds of paths, this algorithm runs quite fast, but is not exact because a shortest path could use nodes that, if the path were longer, may belong to two different node independent paths [white2001b].

Heuristics for computing k-components and their average connectivity

The logic of the heuristics presented here is based on repeatedly applying fast algorithms for k-cores[7] [batagelj2011] and biconnected components[8] [tarjan1972] in order to narrow down the number of pairs of different nodes over which we have to compute their local node connectivity for building the auxiliary graph in which two nodes are linked if they have at least k node independent paths connecting them. We follow the classical insight that, *:math:'k'-cores can be regarded as seedbeds, within which we can expect highly cohesive subsets to be found* [seidman1983]. More formally, our approach is based on Whitney's theorem [white2001], which states an inclusion relation among node connectivity $\kappa(G)$, edge connectivity $\lambda(G)$ and minimum degree $\delta(G)$ for any graph G:

$$\kappa(G) \leq \lambda(G) \leq \delta(G)$$

This theorem implies that every k-component is nested inside a k-edge-component, which in turn, is contained in a k-core. This approach does not require computing node independent paths for all pairs of different nodes as a starting point, thus saving an important amount of computation. Moreover it does not require recursively applying the same procedure over each subgraph. In our approach we only have to compute node independent paths among pairs of different nodes in each biconnected part of each k-core, and repeat this procedure for each k from 3 to the maximal core number of a node in the input network.

The aim of the heuristics presented here is to provide a fast and reasonably accurate way of analyzing the cohesive structure of empirical networks of thousands of nodes and edges. As we

3. See http://networkx.readthedocs.org/en/latest/reference/generated/networkx.algorithms.connectivity.connectivity.average_node_connectivity.html .

4. See http://networkx.readthedocs.org/en/latest/reference/generated/networkx.algorithms.connectivity.kcomponents.k_components.html .

5. See http://networkx.readthedocs.org/en/latest/reference/generated/networkx.algorithms.connectivity.kcutsets.all_node_cuts.html .

6. See http://networkx.readthedocs.org/en/latest/reference/generated/networkx.algorithms.approximation.connectivity.node_connectivity.html .

7. See http://networkx.readthedocs.org/en/latest/reference/generated/networkx.algorithms.core.k_core.html .

8. See http://networkx.readthedocs.org/en/latest/reference/generated/networkx.algorithms.components.biconnected.biconnected_components.html .

have seen, k-components are the cornerstone of structural cohesion analysis. But they are very expensive to compute. Our approach consists of computing extra-cohesive blocks of level k for each biconnected component of a k-core. Extra-cohesive blocks are a relaxation of the k-component concept in which not all node independent paths among pairs of different nodes have to run entirely inside the subgraph. Thus, there is no guarantee that an extra-cohesive block of level k actually has node connectivity k. We introduce an additional constraint to the extra-cohesive block concept in order to approximate k-components: our algorithm computes extra-cohesive blocks of level k that are also k-cores by themselves in G. Furthermore, extra-cohesive blocks maintain high requirements in terms of multiconnectivity and robustness, thus conserving the most interesting properties from a sociological perspective on the structure of social groups.

Combining this logic with three observations about the auxiliary graph H allows us to design a new algorithm[9] for finding extra-cohesive blocks in each biconnected component of a k-core, that can either be exact but slow ---using flow-based algorithms for local node connectivity [brandes2005] --- or fast and approximate, giving a lower bound with certificate of the composition and the connectivity of extra-cohesive blocks ---using [white2001b] approximation for local node connectivity. Once we have a fast way to compute extra-cohesive blocks, we can approximate k-components by imposing that the induced subgraph of the nodes that form an extra-cohesive block of G have to also be a k-core in G.

Let H be the auxiliary graph in which two nodes are linked if they have at least k node independent paths connecting them in each of the biconnected components of the core of level k of original graph G (for $k > 2$). The first observation is that complete subgraphs in H (H_{clique}) have a one to one correspondence with subgraphs of G in which each node is connected to every other node in the subgraph for at least k node independent paths. Thus, we have to search for cliques in H in order to discover extra-cohesive blocks in G.

The second observation is that an H_{clique} of order n is also a core of level $n-1$ (all nodes have core number $n-1$), and the degree of all nodes is also $n-1$. The auxiliary graph H is usually very dense, because we build a different H for each biconnected part of the core subgraph of level k of the input graph G. In this kind of network big clusters of almost fully connected nodes are very common. Thus, in order to search for cliques in H we can do the following:

1) For each core number value c_{value} in each biconnected component of H:
2) Build a subgraph $H_{candidate}$ of H induced by the nodes that have *exactly* core number c_{value}. Note that this is different than building a k-core, which is a subgraph induced by all nodes with core number *greater or equal than* c_{value}.
3) If $H_{candidate}$ has order $c_{value}+1$ then it is a clique and all nodes will have degree $n-1$. Return the clique and continue with the following candidate.
4) If this is not the case, then some nodes will have degree $< n-1$. Remove all nodes with minimum degree from $H_{candidate}$.
5) If the graph is trivial or empty, continue with the following candidate. Or otherwise recompute the core number for each node and go to 3.

Finally, the third observation is that if two k-components of different order overlap, the nodes that overlap belong to both cliques in H and will have core numbers equal to all other nodes in the bigger clique. Thus, we can account for possible overlap when building subgraphs $H_{candidate}$ (induced by the nodes that have *exactly* core number c_{value}) by also adding to the candidate subgraph the nodes in H that are connected to all nodes that have *exactly* core number c_{value}. Also, if we sort the subgraphs $H_{candidate}$ in reverse order (starting from the biggest), we can skip checking for possible overlap for the biggest.

Based on these three observations, our heuristics for approximating the cohesive structure of a network and the average connectivity of each individual block, consists of:

Let G be the input graph. Compute the core number of each node in G. For each k from 3 to the maximum core number build a k-core subgraph G_{k-core} with all nodes in G with core level $\geq k$.

For each biconnected component of G_{k-core}:

1) Compute local node connectivity $\kappa(u,v)$ between all pairs of different nodes. Optionally store the result for each pair. Either use a flow-based algorithm (exact but slow) or White and Newman's approximation for local node connectivity (approximate but a lot faster).
2) Build an auxiliary graph H with all nodes in this bicomponent of G_{k-core} with edges between two nodes if $\kappa(u,v) \geq k$. For each biconnected component of H:
3) Compute the core number of each node in $H_{bicomponent}$, sort the values in reverse order (biggest first), and for each value c_{value}:
    a) Build a subgraph $H_{candidate}$ induced by nodes with core number *exactly* equal to c_{value} plus nodes in H that are conected with all nodes with core number equal to c_{value}.
        i) If $H_{candidate}$ has order $c_{value}+1$ then it is a clique and all nodes will have degree $n-1$. Build a core subgraph $G_{candidate}$ of level k of G induced by all nodes in $H_{candidate}$ that have core number $\geq k$ in G.
        ii) If this is not the case, then some nodes will have degree $< n-1$. Remove all nodes with minimum degree from $H_{candidate}$. Build a core subgraph $G_{candidate}$ of level k of G induced by the remaining nodes of $H_{candidate}$ that have core number $\geq k$ in G.
            A) If the resultant graph is trivial or empty, continue with the following candidate.
            B) Else recompute the core number for each node in the new $H_{candidate}$ and go to (i).
    b) The nodes of each biconnected component of $G_{candidate}$ are assumed to be a k-component of the input graph if the number of nodes is greater than k.
    c) Compute the average connectivity of each detected k-component. Either use the value of

9. See http://networkx.readthedocs.org/en/latest/reference/generated/networkx.algorithms.approximation.kcomponents.k_components.html .

$\kappa(u,v)$ computed in step 1 or recalcualte $\kappa(u,v)$ in the induced subgraph of candidate nodes.

Notice that because our approach is based on computing node independent paths between pairs of different nodes, we are able to use these computations to calculate both the cohesive structure and the average node connectivity of each detected k-component. Of course, computing average connectivity comes with a cost: either more space to store $\kappa(u,v)$ in step 1, or more computation time in step 3.c if we did not store $\kappa(u,v)$. This is not possible when applying the exact algorithm for k-components proposed by [moody2003] because it is based on repeatedly finding k-cutsets and removing them, thus it does not consider node independent paths at all.

The output of these heuristics is an approximation to k-components based on extra-cohesive blocks. We find extra-cohesive blocks and not k-components because we only build the auxiliary graph H one time on each bicoennected component of a core subgraph of level k from the input graph G. Local node connectivity is computed in a subgraph that might be larger than the final $G_{candidate}$ and thus some node independent paths that shouldn't could end up being counted.

Accuracy can be improved by rebuilding H from the pairwise node connectivity in $G_{candidate}$ and following the remaining steps of the heuristics at the cost of slowing down the computation. There is a trade-off between speed and accuracy. After some tests we decided to compute H only once and lean towards the speed pole of the trade-off. Our goal is to have an usable procedure for analyzing networks of thousands of nodes and edges in which we have substantive interests. Following this goal, the use of [white2001b] approximation algorithm for local node connectivity in step 3.b is key. It is almost on order of magnitude faster than the exact flow-based algorithms. As usual, speed comes with a cost in accuracy: [white2001b] algorithm provides a strict lower bound for the local node connectivity. Thus, by using it we can miss an edge in H that should be there.

Our tests reveal that the use of [white2001b] approximation does indeed underestimate the order of some k-components, particularly in not very sparse networks. One approach to mitigate this problem is to relax the strict cohesion requirement of $H_{candidate}$ being a clique. Following the network literature on cliques, we can relax its cohesion requirements in terms of degree, coreness and density. We did some experiments and found that a good relaxation criteria is to set a density threshold of 0.95 for $H_{candidate}$.

Case study: Structural cohesion in collaboration networks

The structural cohesion model can be used to explain cooperation in different kinds of collaboration networks; for instance, co-authorship networks ([moody2004] , [white2004]) and collaboration among biotech firms [powell2005]. Most collaboration networks are modeled as bipartite graphs, in which nodes can be divided in two disjoin sets, and edges only connect nodes from opposite sets. In the case of co-authorship networks, one node set represents authors and the other papers. Each author has edges that link her to all papers she authored. The usual practice to deal with bipartite networks is focus the analysis only on unipartite projections. That is, a new network only with the nodes that represent authors from the original bipartite network, where two authors are linked by an edge if they co-authored a paper together.

However, recent literature on bipartite networks strongly suggests that it is necessary to analyze bipartite networks directly to get an accurate picture ([uzzi2007], [opsahl2011], [latapy2008]). We show that this is also the case for the k-component structure of collaboration networks. This kind of analysis has been conducted very rarely on bipartite networks, and only on very small ones [white2004]. Its limited diffusion can be readily explained by the fact that bipartite networks are usually quite a lot bigger than their unipartite counterparts, and the computational requirements, once again, stifled empirical research in this direction.

The heuristics for structural cohesion presented here allow us to analyze relatively large networks (up to tens of thousands of nodes and edges) quickly enough to be practical. To illustrate this we use data on collaboration among software developers in one organization (the Debian project) and scientists publishing papers in the arXiv.org electronic repository in two different scientific fields: High Energy Theory and Nuclear Theory. We built the Debian collaboration network by linking each software developer with the packages (i.e. programs) that she uploaded to the package repository of the Debian Operating System during a complete release cycle. We analyze the Debian Operating System version 5.0, codenamed *Lenny*, which was developed from April 8, 2007, to February 1, 2009. Scientific networks are built using all the papers uploaded to the arXiv.org preprint repository from January 1, 2006, to December 31, 2010, for High Energy Physics Theory and Nuclear Theory. In these networks each author is linked to the papers that she has authored during the time period analyzed. Unipartite projections consist of scientists linked together if they have co-authored a paper, and developers linked together if they have worked on the same program. Table 1 presents some details on those networks (which are available, see[10]).

In the remaining part of this section we perform two kinds of analysis to illustrate how the structural cohesion model can help us understand the structure and dynamics of collaboration networks. First, we present a tree representation of the k-component structure ---which is also named cohesive blocks structure in the literature ([white2001], [moody2003], [white2004], [mani2014])--- for our bipartite networks and their unipartite projections, both for actual networks and for their random counterparts. Finally, we present a novel graphic representation of the structural cohesion of a network, based on three-dimensional scatter plot, using average node connectivity as a fine-grained measure of cohesion of each k-component.

For the first analysis we do need to generate null models in order to discount the possibility that the observed structure of actual networks is just the result of randomly mixing papers and scientists or packages and developers. The null models used in this paper are based on a bipartite configuration model [newman2003], which consists of generating networks by randomly assigning papers/programs to scientists/developers but maintaining constant the distribution of papers per scientists and scientists by paper observed in the actual networks. For unipartite projections, we generated bipartite random networks, and then performed the unipartite projection.

10. You can download the networks used in this section in graphml format. Nodes have an attribute named *bipartite*, with values 0 and 1, which indicates the node set to which each node belongs. Note that this is the convention used in NetworkX's bipartite package (see https://networkx.github.io/documentation/latest/reference/algorithms.bipartite.html):

- Debian Lenny: http://dx.doi.org/10.6084/m9.figshare.1472938
- Nuclear Theory: http://dx.doi.org/10.6084/m9.figshare.1472940
- High Energy Theory: http://dx.doi.org/10.6084/m9.figshare.1472939

	Bipartite				Unipartite			
Network	# nodes	# edges	Av. degree	Time(s)	# nodes	# edges	Av. degree	Time(s)
Debian Lenny	13,121	20,220	3.08	1,105.2	1,383	5,216	7.54	204.7
High Energy (theory)	26,590	37,566	2.81	3,105.7	9,767	19,331	3.97	7,136.0
Nuclear Theory	10,371	15,969	3.08	1,205.2	4,827	14,488	6.00	3,934.1

TABLE 1: *Collaboration networks analyzed from science and from software development. See text for details on their content. Time refers to the execution of our heuristics on each network expressed in seconds.*

So let's start with the tree representation of the cohesive blocks structure. As proposed by [white2004], we can represent the k-component structure of a network by drawing a tree whose nodes are k-components; two nodes are linked if the k-component of higher level is nested inside the k-component of lower level (see pp. 1643, 1651 from [mani2014] for this kind of analysis on the Indian firm ownership network). This representation of the connectivity structure can be built during the run time of the exact algorithm. However, because our heuristics are based on finding node independent paths, we have to compute first the k-components hierarchy, and then construct the tree that represents the connectivity structure of the network.

Figures 1 (a) and 1 (c) show the connectivity structure of Nuclear Theory collaboration networks represented as a tree, the former for the bipartite network and the latter for the unipartite one. As we can see, both networks display non-trivial structure. The bipartite network has up to an 8-component, but most nodes are in k-components with $k < 6$. Up to $k = 3$ most nodes are in giant k-components, but for $k = \{4,5\}$ there are many k-components of similar order. Figure 1 (c), which corresponds to the unipartite projection, has a lot more connectivity levels. In this network, the maximum connectivity level is 46; the four long legs of the plot correspond to 4 cliques with 47, 31, 27 and 25 nodes. Notice that each one of these 4 cliques are already a separated k-component at $k = 7$ It is at this level of connectivity ($k = \{7,8\}$) where the giant k-components start to dissolve and many smaller k-components emerge.

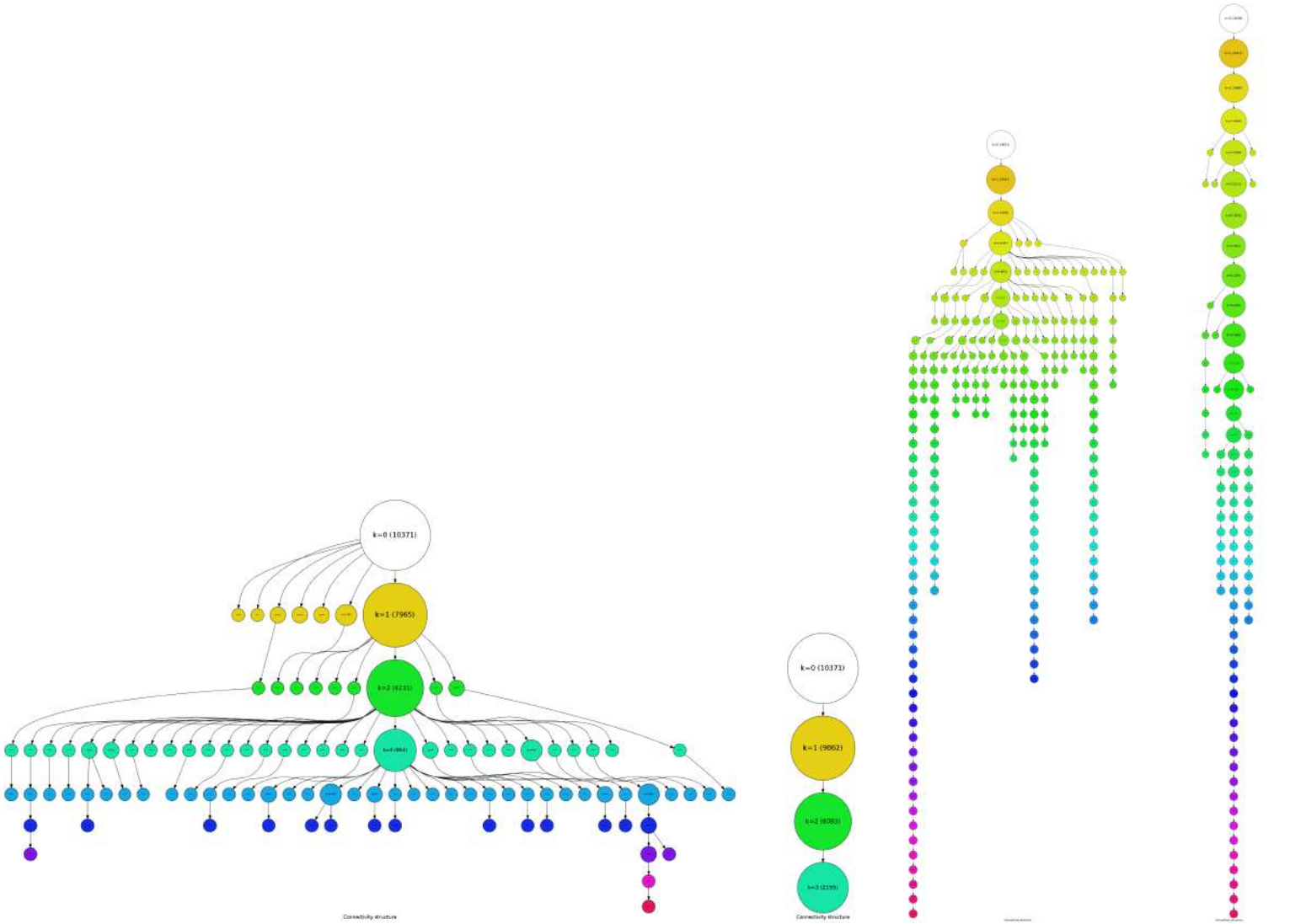
In order to be able to assess the significance of the results obtained, we have to compare the connectivity structure of actual networks with the connectivity structure of a random network that maintains some constraints observed in the empirical networks. In this case, we compare actual networks with only one random network. We obtained it by generating 1000 random networks and choosing one randomly. Figures 1 (b) and 1 (d) show the connectivity structure of the random counterparts for Nuclear Theory collaboration networks. For the bipartite network, instead of the differentiated connectivity structure displayed by the actual bipartite network, there is a flatter connectivity structure, where the higher level k-component is a tricomponent. Moreover, instead of many small k-components at high connectivity levels, the random bipartite network has only giant k-components where all nodes with component number k are. In this case, the unipartite network is also quite different from its random counterpart. There are only giant k-components up until $k = 15$, where the four cliques observed in the actual network separate from each other to form distinct k-components.

Going one step beyond classical structural cohesion analysis, as proposed above, we can deepen our analysis by also considering the average connectivity of the k-components of these networks. By analogy with the k-component number of each node, which is the maximum value k of the deepest k-component in which that node is embedded, we can establish the average k-component number of each node as the value of average connectivity of the deepest k-component in which that node is embedded. Notice that, unlike plain node connectivity, average node connectivity is a continuous measure of cohesion. Thus it provides a more granular measure of cohesion because we can rank k-components with the same k according to their average node connectivity.

Figure 2 graphically represent the three networks with three-dimensional scatter plots produced with the powerful Matplotlib library [hunter2007]. In these graphs, each dot corresponds to a node of the network, for bipartite networks nodes represent both scientists/developers and papers/programs. The Z axis (the vertical one) is the average k-component number of each node, and the X and Y axis are the result of a 2 dimensional force-based layout algorithm implemented by the *neato* program of Graphviz. The two dimensional layout is computed by constructing a virtual physical model and then using an iterative solver procedure to obtain a low-energy configuration. Following [kamada1989], an ideal spring is placed between each pair of nodes (even if they are not connected in the network). The length of each spring corresponds to the geodesic distance between the pair of nodes that it links. The final node positioning in the layout approximates the path distance among pairs of nodes in the network.

This novel graphic representation of cohesion structure is inspired by the approximation technique developed by [moody2004] for plotting the approximate cohesion contour of large networks to which is not practical to apply Moody & White (2003) exact algorithm for k-components. Moody's technique is based on the fact that force-based layouts algorithms tend to draw nodes within highly cohesive subgroups near each other. Then we have to divide the surface of the two-dimensional plane in squares of equal areas and compute node independent paths on a sample of pairs of nodes inside each square so as to obtain an approximation for the node connectivity in that square. Then we can draw a surface plot using a smoothing probability density function. However, in order to obtain a nice smooth surface plot, we have to use heavy smoothing in the probability density function, and carefully choose the area of the squares (mostly by trial and error). Moreover, this technique strongly relies on the force-based layout algorithm to put nodes in highly cohesive subgroups near each other ---something which is not guaranteed because they are usually based in path distance and not directly on node connectivity. Because we are able to compute the k-component structure with our heuristics for large networks, the three-dimensional scatter plot only relies on the layout algorithm for setting the X and Y positions of the nodes, while the Z position (average node connectivity) is computed directly from the network. Moreover, we don't have to use a smoothed surface plot because we have a value of average connectivity for each node, and thus we can plot each node as a dot on the plot. This gives a more accurate picture of the actual cohesive structure of a network.

This representation of cohesive structures can help researchers visualize the presence of different organizational mechanisms in



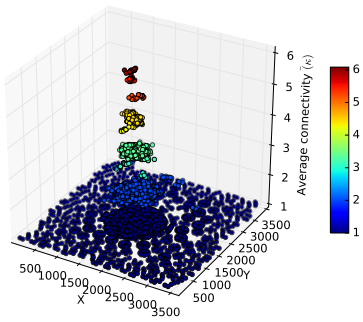
(a) Actual bipartite network

(b) Random bipartite network

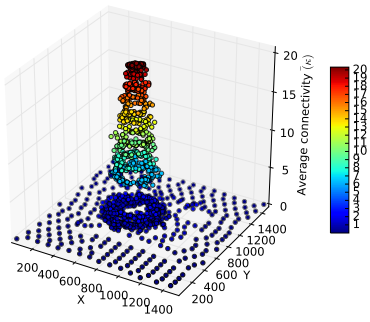
(c) Actual unipartite network

(d) Random unipartite network

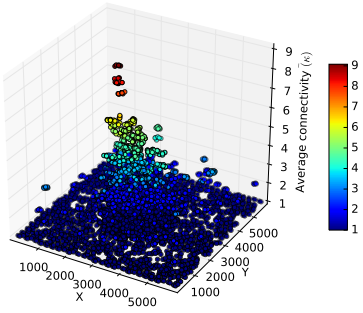
Fig. 1: *Cohesive blocks for bipartite and unipartite Nuclear Theory collaboration networks, and for their random counterparts. Random networks were generated using a bipartite configuration model. We built 1000 random networks and chose one randomly, see text for details. For lower connectivity levels we have removed some small k-components to improve the readability: we do not show 1-components with less than 20 nodes, 2-components with less than 15 nodes, or tricomponents with less than 10 nodes.*



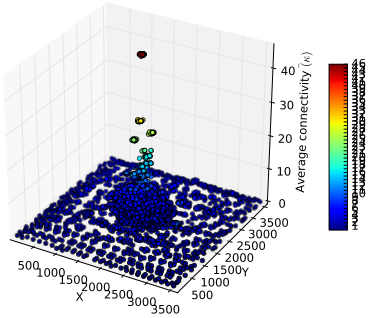
(a) Bipartite Debian Lenny network



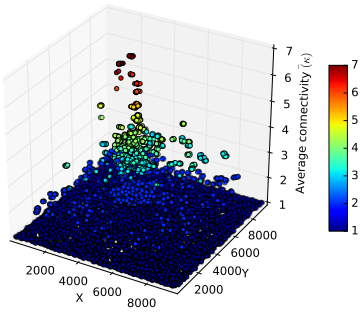
(b) Unipartite Debian Lenny network



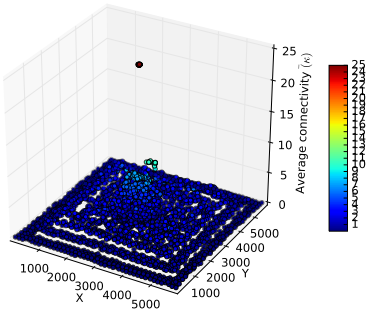
(c) Bipartite Nuclear Theory network



(d) Unipartite Nuclear Theory network



(e) Bipartite High Energy Theory network



(f) Bipartite High Energy Theory network

Fig. 2: *Average connectivity three-dimensional scatter plots. X and Y are the positions determined by the Kamada-Kawai layout algorithm. The vertical dimension is average connectivity. Each dot is a node of the network and bipartite networks contain both papers/programs and scientists/developers.*

different kinds of collaboration networks. The difference between the Debian and the scientific collaboration networks is striking. In figure 2 (a) we can see the scatter plot for a Debian bipartite network. We can observe a clear vertical separation among nodes in different connectivity levels. This is because almost all nodes in each connectivity level are in a giant k-component and thus they have the same average connectivity. In other words, developers in Debian show different levels of engagement and contribution, with a core group of developers deeply nested at the core of the community. This pattern is the result of formal and informal rules of collaboration that evolved over the years [ferraro2007] into a homogeneous hierarchical structure, where there is only one core of highly productive individuals at the center. Not surprisingly, perhaps, the Debian project has been particularly resilient to developers' turnover and splintering factions.

Scientific collaboration networks show a rather different structure of collaboration. The bipartite science collaboration networks (figures 2 (c) and 2 (e)) display a continuous hierarchical structure in which there are nodes at different levels of average connectivity for each discrete plain connectivity level. This is because science collaboration networks have a complex cohesive block structure where there are a lot of independent k-components in each plain connectivity level, for $k \geq 3$. Each small cohesive block has a different order, size and average connectivity; thus, when we display them in this three-dimensional scatter plot we observe a continuous hierarchical structure that contrasts with the almost discrete structure of Debian collaboration networks.

One explanation why we observe this heterogeneous connectivity structure is that scientific collaborations cluster around a variety of different aims, methods, projects, and institutional environments. Therefore as the most productive scientists collaborate with each other, hierarchies naturally emerge. However, we are less likely to observe one single hierarchical order as we did in the Debian network, as more than one core of highly productive scientists is likely to emerge.

If we compare the bipartite networks with their unipartite projections using this graphical representation (see figures 2 (b), 2 (d), and 2 (f)) we can see that, again, they look quite different. While bipartite average connectivity structure for the Debian network is characterized by clearly defined and almost discrete hierarchical levels, its unipartite counterpart shows a continuous hierarchical structure. However, this is not caused by the presence of many small k-components at the same level k, as in the case of bipartite science networks discussed above, but by the close succession of hierarchy levels with almost the same number of nodes in a chain-like structure.

For collaboration science networks, the three-dimensional scatter plots of unipartite projections are also quite different than their original bipartite networks. They have a lot more hierarchy levels than bipartite networks but most nodes are at lower connectivity levels. Only a few nodes are at top levels of connectivity, and they all form part of some clique, which are the groups in the long *legs* of the cohesive block structure depicted in figure 1 (c). Thus, the complex hierarchical connectivity structure of bipartite collaboration networks gets blurred when we perform unipartite projection. An important consequence of the projection is that only a few nodes embedded in big cliques appear at top connectivity levels and all other nodes are way down in the connectivity structure. This could lead the risk of overestimating the importance of those nodes in big cliques and to underestimate the importance of nodes that, despite being at high levels of the bipartite connectivity structure, appear only at lower levels of the unipartite connectivity structure.

Conclusions

We developed heuristics to compute the k-components structure, along with the average node connectivity for each k-component, based on the fast approximation to compute node independent paths [white2001b]. These heuristics allow for the computing of the approximate value of group cohesion for moderately large networks in a reasonable time frame. We showed that these heuristics can be applied to networks at least one order of magnitude bigger than the ones manageable by the exact algorithm proposed by [moody2003]. To ensure reproducibility and facilitate diffusion of these heuristics we provided an implementation of both the exact algorithm and the heuristics on top of NetworkX [hagberg2008]. These implementations are included in the recently released 1.10 version of NetworkX.

We analyzed three large collaboration networks and showed that the heuristics and the novel visualization technique for cohesive network structure help us capture important differences in the way collaboration is structured. Future research could leverage the tools we provide to systematically measure those structures. For instance, sociologists of science often compare scientific disciplines in terms of their collaborative structures [moody2004] and their level of controversies [bearman2010]. The measures and the visualization technique we proposed could nicely capture these features and compare them across scientific disciplines. This would make it possible to further our understanding of the social structure of science, and its impact in terms of productivity, novelty and impact. Social network researchers interested in organizational robustness would also benefit from leveraging the structural cohesion measures to detect sub-groups that are more critical to the organization's resilience, and thus prevent factionalization. Exploring the consequences of different forms of cohesive structures will eventually help us further our understanding of collaboration and the role that cohesive groups play in linking micro-level dynamics with macro-level social structures.

References

[batagelj2011] Batagelj, V. and M. Zaveršnik (2011). Fast algorithms for determining (generalized) core groups in social networks. Advances in Data Analysis and Classification 5(2), 129–145.

[bearman2010] Shwed, U. and P. Bearman (2010). The temporal structure of scientific consensus formation. American sociological review 75(6), 817–840.

[beineke2002] Beineke, L., O. Oellermann, and R. Pippert (2002). The average connectivity of a graph. Discrete mathematics 252(1-3), 31–45.

[brandes2005] Brandes, U. and T. Erlebach (2005). Network analysis: methodological foundations, Volume 3418. Springer Verlag.

[doreian1998] Doreian, P. and T. Fararo (1998). The problem of solidarity: theories and models. Routledge.

[freeman1992] Freeman, L. (1992). The sociological concept of "group": An empirical test of two models. American Journal of Sociology, 152–166.

[fortunato2010] Fortunato, S. (2010). Community detection in graphs. Physics Reports, 486(3), 75-174.

[grannis2009] Grannis, R. (2009). Paths and semipaths: reconceptualizing structural cohesion in terms of directed relations. Sociological Methodology 39(1), 117–150.

[hagberg2008] Hagberg, A., Schult, D. A., & Swart, P. (2008). Exploring network structure, dynamics, and function using NetworkX. In Proceedings of the 7th Python in Science Conferences (SciPy 2008) (Vol. 2008, pp. 11-16).

[hunter2007] Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. Computing In Science & Engineering 9(3), 90–95.

[kamada1989] Kamada, T. and S. Kawai (1989). An algorithm for drawing general undirected graphs. Information processing letters 31(1), 7–15.

[kanevsky1993] Kanevsky, A. (1993). Finding all minimum-size separating vertex sets in a graph. Networks 23(6), 533–541.

[latapy2008] Latapy, M., C. Magnien, and N. Vecchio (2008). Basic notions for the analysis of large two mode networks. Social Networks 30(1), 31–48.

[mani2014] Mani, D. and J. Moody (2014). Moving beyond stylized economic network models: The hybrid world of the indian firm ownership network. American Journal of Sociology 119(6), pp. 1629–1669.

[moody2004] Moody, J. (2004). The structure of a social science collaboration network: Disciplinary cohesion from 1963 to 1999. American Sociological Review 69(2), 213–238.

[moody2003] Moody, J., & White, D. R. (2003). Structural cohesion and embeddedness: A hierarchical concept of social groups. American Sociological Review, 103-127.

[newman2003] Newman, M. (2003). The structure and function of complex networks. SIAM Review 45, 167.

[ferraro2007] O'Mahony, S. and F. Ferraro (2007). The emergence of governance in an open source community. The Academy of Management Journal 50(5), 1079–1106.

[opsahl2011] Opsahl, T. (2011). Triadic closure in two-mode networks: Redefining the global and local clustering coefficients. Social Networks 34.

[powell2005] Powell, W., D. White, K. Koput, and J. Owen-Smith (2005). Network dynamics and field evolution: The growth of interorganizational collaboration in the life sciences. American Journal of Sociology 110(4), 1132–1205.

[simon1962] Simon, H. A. (1962). The architecture of complexity. Proceedings of the American philosophical society 106(6), 467–482.

[seidman1983] Seidman, S. (1983). Network structure and minimum degree. Social networks 5(3), 269–287.

[tarjan1972] Tarjan, R. (1972). Depth-first search and linear graph algorithms. In Switching and Automata Theory, 1971., 12th Annual Symposium on, pp. 114–121. IEEE.

[uzzi2007] Uzzi, B., L. Amaral, and F. Reed-Tsochas (2007). Small-world networks and management science research: a review. European Management Review 4(2), 77–91.

[wasserman1994] Wasserman, S., & Faust, K. (1994). Social network analysis: Methods and applications (Vol. 8). Cambridge university press.

[white2004] White, D., J. Owen-Smith, J. Moody, and W. Powell (2004). Networks, fields and organizations: micro-dynamics, scale and cohesive embeddings. Computational & Mathematical Organization Theory 10(1), 95–117.

[white2001b] White, D. and M. Newman (2001). Fast approximation algorithms for finding node-independent paths in networks. Santa Fe Institute Working Papers Series.

[white2001] White, D. R., & Harary, F. (2001). The cohesiveness of blocks in social networks: Node connectivity and conditional density. Sociological Methodology, 31(1), 305-359.