Overlapping community detection on hypergraphs

BSc Thesis

Botond Simon
Eötvös Loránd University
Physics BSc

Supervisors: Dr. Gergely Palla, Bianka Kovács
ELTE, Department of Biological Physics

2023, Budapest

Contents

Introduction

Over the past decades, a variety of complex systems have been successfully described as networks whose interacting pairs of nodes are connected by edges. These interactions were usually formed as pairwise relations until recently, when more and more attention has been devoted to the higher-order architecture of real complex systems. In my thesis, I will discuss different frameworks, including the most advanced of all, the hypergraphs to describe higher-order systems and to represent the higher-order interactions in a compact and understandable way, while – along the way – highlighting the links between well-known representations, such as simple graphs and this higher-order concept. I will review different measures characterizing the structure of these systems, examining the parameters and community configuration of these interactions, accounting for the vast load of information carried.

In the first part of Sect. 1, the more basic and well-known representations of these systems will be discussed, as part of the Network Theory, serving as a starting point for later comparisons between the representations. Then, the higher-order interactions will be introduced, as a solution for representing the complexity of real-life systems and networks more faithfully. The most advanced, compact, and understandable representation of these networks of higher-order interactions is the hypergraph representation. As going through the unique features and measures of hypergraphs, the algorithm of the **h-ABCD** (**H**ypergraph **A**rtificial **B**enchmark for **C**ommunity **D**etection) hypergraph generating model, and its tuneable parameters will be discussed.

The interactions of a hypergraph can be mapped to a system of pairwise interactions, meaning that a hypergraph can be represented as a simple graph, but when doing so it is evident that the difference in the amount and form of information carried by the representations should be also taken into account. The enhancement of the simple graph representation – described in Sect. 2 – through the application of link weights and thresholds within the model will help me to show the interoperability between the different formalisms. My thesis will build on the differences in community structure between the one composed by the **h-ABCD** hypergraph generating model, serving as a baseline for community detection, and an algorithm implemented by me, the base of which lies within the so-called k-clique percolation method.

Finally, in Sect. 3 I will discuss the phase transition – a well-known phenomenon within networks built on pairwise relations, known to take place as a result of the change of some external conditions – that can be observed in the simple graph representations of generated hypergraphs as a result of applying weight threshold levels and clearing out certain, weak edges from the network. I will search for an optimal k value for my implementation of the community detection and weight threshold level, by studying the critical point of a phase transition within the hypergraphs, representations of higher-order interactions.

1 Higher-order interactions in networks

1.1 Networks in general

A network [1] is a structure containing various objects that are – in some form or another – connected. The objects are mathematical abstractions often called nodes (denoted by N), and they are connected with lines, or curves that we will call edges (denoted by E). The edge between two nodes can be directed or undirected, symbolizing symmetric or asymmetric relations. This way it is possible to distinguish the networks with directed edges as directed networks, and the ones with undirected edges – undirected networks, or for now, simply **Networks**, as I am not going to study directed networks during the course of my thesis.

Networks can be used to model many types of relations and processes in physical, biological, social, and information systems. They can represent numerous practical problems, so they have been widely used in branches like:

- Computer science
- Linguistics
- Natural sciences such as Physics, Chemistry, Biology, or Mathematics
- Social sciences

A complex network is a network with non-trivial topological features, those which do not occur in simple systems such as lattices or random networks. Such features include a heavy tail in the degree distribution, a high clustering coefficient, hierarchical structure, and community structure which I will study further in 2.

With the growing interest in graph theory, more extensive and abstract problems have been studied with the use of Complex Networks [2, 3]. These can represent either

- an electric network
- a social network
- a narrative network
- a biological network, etc.

The simplest mathematical representations of networks, the graphs [4], are structures used to model pairwise relations. They consist of nodes and edges, the edges resembling the connections between the nodes. By the direction of the connections we can differentiate directed and undirected graphs, and by the edge relations, simple and multigraphs, the latter of which – consisting of self-edges or loops (an edge connecting a node to itself) as well – will not be studied during my thesis. Although self-edges are not allowed in simple graphs, nodes without connections are, so a node may exist in a graph and not belong to an edge. Also by the number of connections, we can highlight the fully connected graphs where the number of edges is at the maximum possible value, as complete graphs. A graph can be called complete if each pair of its nodes are connected by an edge. Sub-graphs are worth to be defined as well, as they will emerge later in Sect. 2. A sub-graph is a graph whose nodes and edges are subsets of another graph.

I am going to work with undirected simple graphs, the edges of which resemble undirected pairwise interactions between the nodes.

1.2 Simplicial complexes and Hypergraphs

Taking the issue a step further, the usage of graphs that are easily understandable and yield more information in a compact way is the goal. We need to get rid of certain obligations. The most deterrent of these obligations is that graphs can only work with pair interactions, but in reality, problems are far more complex than this.

In mathematics, a simplicial complex [5] is a set composed of points, line segments, triangles, and their n-dimensional counterparts; or in a collective expression simplexes (or simplices). In geometry, a simplex [6] is a generalization of the notion of a triangle or tetrahedron to arbitrary dimensions. It represents the simplest possible polytope in any given dimension:

- A point in 0 dimensions,
- a line segment in 1 dimension,
- a triangle in 2 dimensions,
- and a tetrahedron in 3 dimensions, etc.

A set of connected simplexes must satisfy the following rules:

- Every face of a simplex from $\mathcal{K}$ is also in $\mathcal{K}$,
- The non-empty intersection of any two simplexes denoted by $\sigma_1, \sigma_2 \in \mathcal{K}$ is a face for both σ_1 and σ_2

to create a simplicial complex $\mathcal{K}$. A simplicial k-complex $\mathcal{K}$ is a simplicial complex where the largest dimension of any simplex in $\mathcal{K}$ equals k. For instance, a simplicial 2-complex must contain at least one triangle, and must not contain any tetrahedra or higher-dimensional simplices. Although simplicial complexes are more advanced and applicable in terms of information carried than simple graphs are, the obligatory rules do form a kind of barrier for their usage. With simplicial complexes, there is an additional requirement we need to obey if we are representing a network like so. That is if a second-order interaction (i, j, k) exists, then the three corresponding first-order interactions (i, j), (i, k), and (j, k) must also exist [7]. Take a third-order simplicial complex as an example. It has to have an edge for every pair that appears in a triple. Generally for each n-order interaction, we ought to have $\binom{n}{n-k}$ number of $(n-k)$-order interactions, for $k = (1 \dots n)$.

With this rule, although the amount of information and the compactness of the model is elevated, the representation itself will get more complicated and still, in some sense, restricted.

With the generalization of a simplicial complex as a hypergraph, we are free of these requirements and able to form an even more compact representation for a network, than we are with simplicial complexes [8].

A hypergraph [9] is an object, in which we have the freedom to make an edge join more than two nodes, a rule to which we have to obey in ordinary graphs as ordinary graphs can only model pairwise relations, but, unlike in simplicial complexes, we are not obliged to have smaller order interactions as a consequence of higher order ones [10]. With this freedom we can express the relationship information presented by the edges of a network in a more complex way, as more data can be represented by the set of edges and nodes, but also in a more compressed way, as we got free from most of our obligations stated by the various representations of networks discussed in Sects. 1.1 and 1.2.

1.3 Basic characteristics of hypergraphs

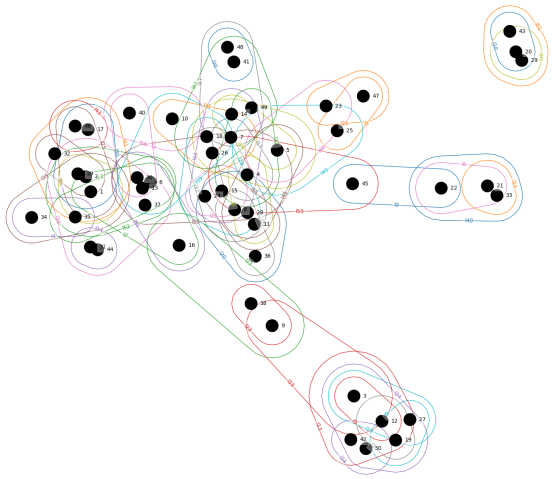
As the hypergraph is one of the most compact generalizations of a network, mentioned earlier in Sect. 1.2, we have some matching attributes and functions between the two. Within hypergraphs, we still have our nodes (denoted by N), and edges (denoted by E), just like in our simple graph. The nodes represent some kind of a mathematical abstraction, an object to be simple, while the edges connecting the nodes are the representations of some relationship between the nodes, now it being a relationship of a minimum of 2 nodes, while in simple graphs it was exactly 2 nodes (not counting self edges, which, of course, can occur in hypergraphs, but I am getting rid of those for now). The degree of a node is equivalent to the number of hyperedges it is part of, just like in simple graphs it is the number of edges by which it is connected to other nodes; but now in the case of hypergraphs – as the edges are not limited to 2 nodes – we have a new measure, hyperedge cardinality (denoted by C), which is equal to the number of nodes within the hyperedge $C_{E_i} = |N|$ (where E_i is the i-th edge and $N \in E_i$).

Just like with simple graphs, hypergraphs can be differentiated by their characteristics and properties [11]. Some of the various hypergraph types:

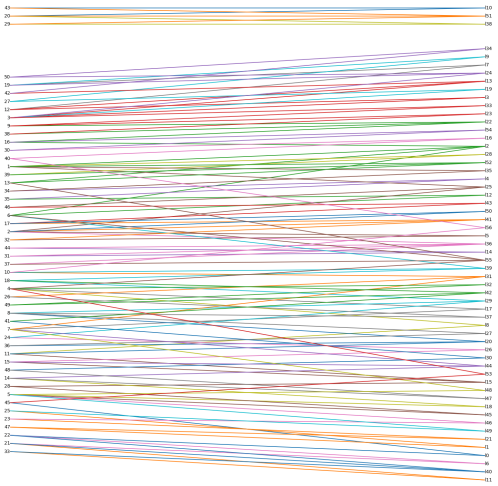
- Empty: one without edges
- Simple: no loops (hyperedges with a single node) or repeated edges (hyperedges containing exactly the same nodes)
- Non-simple or Multiple (or Multi-): contains loops and/or repeated edges
- d-regular: every node has degree d, so all of them are contained by exactly d hyperedges
- k-uniform: each hyperedge contains precisely k nodes, so the each hyperedge has cardinality k

Figure 1a shows a visualization for a 50-node hypergraph, which is a stochastically generated object with the use of the h-ABCD generator [12, 13], the model is described in Sect. 1.4. Figure 1b shows its two-column counterpart and Fig. 1c presents its simple graph representation.

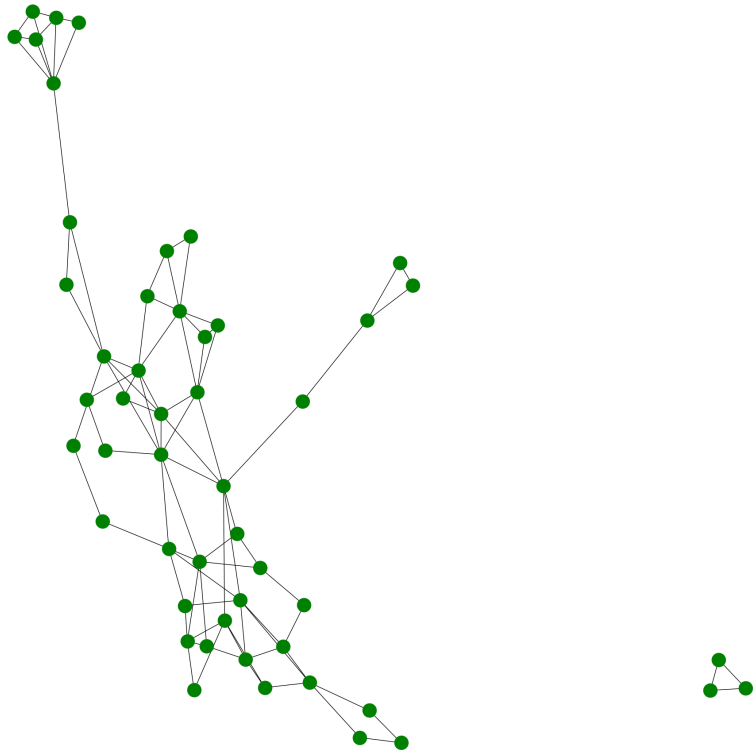
The bipartite representation of a network shows information in a unique, easily understandable way [14]. In the mathematical field of graph theory, a bipartite graph (or bi-graph) is a graph whose nodes can be divided into two disjoint and independent sets E and N – the sets being the same like in simple graphs discussed in Sect. 1.1 – so that every edge connects a node in E to one or more in N. In my case, in the bipartite representation of a hypergraph the E set corresponds to the edges, and the N set corresponds to the nodes in the hypergraph. If we consider the bipartite representation of a simple graph, one node in E is connected to two nodes in N (as a simple graph represents pair-interactions), but if we consider the bipartite representation of a hypergraph, as seen in Fig. 1b, it can be seen clearly, that one hyperedge does not have to oblige to the rule of connecting two nodes, and one node in E (for example E_i, as the i-th hyperedge) is connected to C_{E_i} nodes in N (in my case, not permitting self-edges, with $C_{E_i} \geq 2$), C_{E_i} being the cardinality of the i-th hyperedge.



(a) **Hypergraph**



(b) **Bipartite graph representation**



(c) **Simple graph representation**

Figure 1: A hypergraph is seen in panel 1a. In panel 1b, the bipartite representation provides an easy way to check which nodes (left column) do a hyperedge (right column) contain, or vice versa, which hyperedges do a node belong to. Finally, seen in panel 1c is the simple graph representation. As the nodes between one hyperedge have a connection with each other it means that each hyperedge can also be represented as a complete subgraph of the whole network.

As seen in Fig. 1, a hypergraph can be represented as a simple graph as well, as we are able to represent the set of hyperedges with a set of simple edges while keeping the nodes intact. But of course, one simple edge is not equivalent to one hyperedge. The sheer size of the set is just more vast within the simple graph, and more compact within a hypergraph, as the same information carried by a hyperedge can only be represented through a "unique" edge for every two nodes within said hyperedge (a hyperedge with $C = n$ can be represented through $(n-1)!$ number of simple edges).

If we consider our network for example as one representing a real-life social network, do we think of the relationship between the people of a big group to be the same level as the relationship of people in a considerably smaller group? It seems evident that the nodes being part of a hyperedge with big cardinality C will have a kind of weaker connection in between them than the nodes of a hyperedge with a smaller cardinality. Also, when studying hyperedges, not only the cardinalities, but overlapping edges too, need to be taken into consideration. This means that a node – or even more nodes, as overlappings are not 'restricted' to one node – can be part of more than one hyperedge (with possibly different cardinality for each hyperedge they are a part of). If we take the number of overlappings and the cardinality measure into consideration, a 'weight' attribute can also be added to the simple graph representation, to show the closeness of the nodes or the level of relationship between them. The weight of an edge in the simple graph representation of a hypergraph can be determined based on the number of nodes n within the corresponding hyperedge (so the cardinality of the hyperedge), considering also the overlaps through a summation as:

$$w_{i,j} = \sum_{i,j \in E} \frac{1}{C_N - 1}, \tag{1}$$

where i, j are the nodes the edge connects, N is the hyperedge i, j are part of, and C_N is the cardinality of those hyperedge or hyperedges.

With this implementation, we can evaluate the level of relationship between the nodes with our simple graph representation, to give a detailed view about which edge (or pair of nodes) is carrying information more important for our complex network, than other pairs of nodes.

I have visualized my implementation of edge weights, seen in Fig. 2 – the calculation of which is explained in Eq. 1 – for the simple graph representation of the hypergraphs seen in Fig. 1a; while the implementation of a certain threshold with its effect on the weighted simple graph (the removal of edges falling under the threshold, creating fallout components and nodes along the way) can be seen in Fig. 3 .

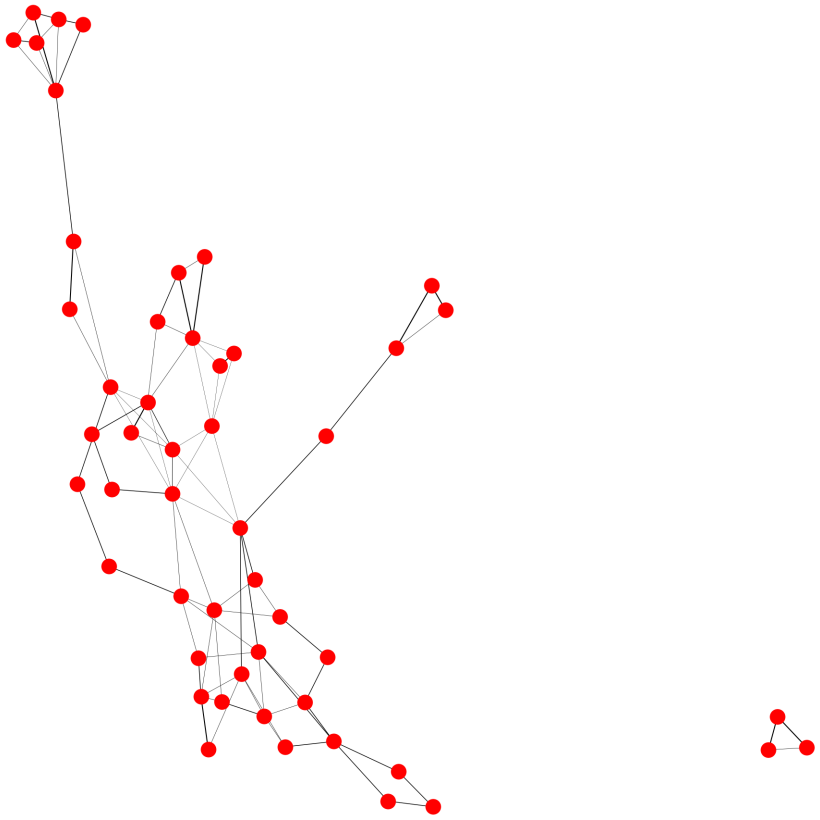


Figure 2: **Weighted graph representation of the hypergraph plotted in Fig. 1a.** With the implemented solution for the edge weighting, it is visible how the carried information compares within a hyper-, a simple (seen in Fig. 1), and a weighted simple edge if we consider the cardinality of the hyperedges and their overlappings. I have used an embedded format, a force-directed representation of the network treating edges as springs holding nodes close, while treating nodes as repelling objects, sometimes called an anti-gravity force [15].

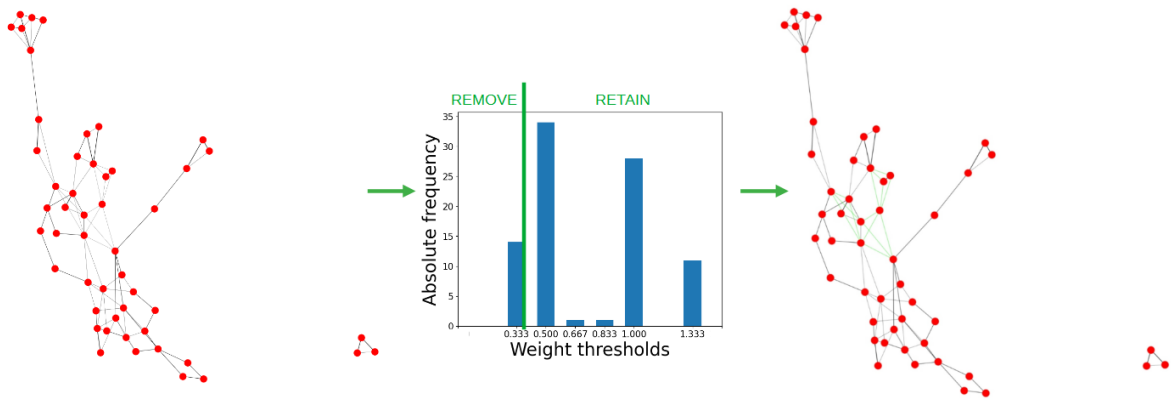


Figure 3: **Weight thresholding in the weighted graph representation of a hypergraph.** On the left side we can clearly see the simple graph with the implemented edge weights as in Fig. 2, and the frequency of each individual weights calculated from Eq. 1 in the middle. This will be important to determine the importance of each edge in the network, as an edge with a higher weight is generally considered to be more essential than others of smaller weights. A weight threshold level means that I am clearing out the edges under or at the said threshold level, in this case, if an edge has 0.333 or lower weight, it is deleted from the graph. This process disassembles the graph, leaving us edges with more weight – so communities with closer relationships, and some stranded nodes, outliers as well. On the right side the same weighted simple graph is visible, after the implementation of said weight threshold. The edges, which have been cut off from the model have a distinct green colour, differentiating them from the edges being kept.

1.4 Hypergraph Artificial Benchmark for Community Detection

Generative models exist for networks, offering various methods with tuneable parameters to generate a desired network or its simple-, hypergraph counterpart.

To generate hypergraphs I have used the Hypergraph Artificial Benchmark for Community Detection generative model, or the h-ABCD generator [12, 13]. The h-ABCD model is a recently introduced and implemented solution, producing similar results as the **LFR** (**L**ancichinetti, **F**ortunato, **R**adicchi) model of graphs describing pairwise interactions [16], but at the same time can easily be tuned to make a smooth transition between two extremes, pure disjoint graphs and random graph with no community structure.

Let me start with the introduction of the parameters of the model. The model generates a hypergraph with a set of n nodes. The degrees are between values (δ, D), following a power-law distribution with exponent γ ($P(k) \sim k^{-\gamma}$). Community sizes are between values (s, S) with a similar power-law distribution, but with an exponent β. The model also has a built-in variable for the level of noise, denoted as ζ. The vector q contains the desired probability of generating hyperedges of each cardinality. Last but not least, a matrix W specifies how many nodes from its own defined community should a community hyperedge have. The ranges of these tuneable parameters are described in Table 1 with a brief description.

Parameter	Range	Description
n	$\mathbb{N}$	Number of Nodes
γ	(2, 3)	Power-law exponent of degree distribution
δ	$\mathbb{N}$	Minimum degree at least δ
D	$\mathbb{N}$	Maximum degree at max D
β	(1, 2)	Power-law exponent of distribution of community sizes
s	$\mathbb{N}/[\delta]$	Community sizes at least s
S	$\mathbb{N}/[s-1]$	Community sizes at most S
ζ	(0, 1)	Level of noise
q	vector of [0, 1]	Distribution of hyperedge sizes (Vector)
W	matrix of [0, 1]	Distribution of hyperedge composition (Matrix)

Table 1: **The tuneable parameters and range of the parameters within the h-ABCD model.**

The degree distribution of nodes and the community sizes of h-ABCD is following a power-law distribution with γ and β exponents respectively. In order to generate a simple hypergraph (not multi-, so loops and parallel edges are not allowed) with n nodes a straightforward generalization of the classical random graph model is used, known as the *configuration model* or the *pairing model*, first introduced by Béla Bollobás [17]. The program starts with a set P of $vol(N) = \sum_{i=1}^{n} w_i$ points (where $vol(N)$ is the sum of degrees over all nodes in the hypergraph, so one w_i point resembles one degree for node n_i) that are partitioned into $n_1, n_2, ...n_n$ buckets, so bucket n_i will consist of w_i points. The set P is partitioned into disjoint P_i sets of various sizes so that $P = P_1 \cup P_2 \cup ...P_n$. Then the hypergraph $P(w)$ from set P is constructed as follows: the nodes are the buckets $n_1, n_2, ...n_n$ and the set P_i – consisting of w_i points from different n_i buckets – corresponds to a hyperedge $e_i \subseteq N$ in $P(w)$. Let me demonstrate the procedure with a smaller hypergraph and these overlapping sets shown in Fig. 4.

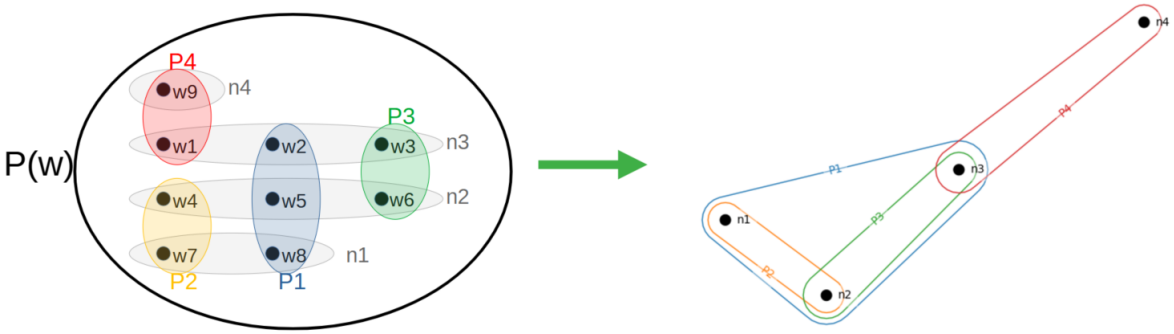


Figure 4: **The generating procedure of the h-ABCD algorithm.** On the left part the set P is seen containing $vol(N) = \sum_{i=1}^{n} w_i$ points, in my case, $w_1, w_2 \dots w_9$. Each of these points resembles a degree for a node within the hypergraph, and the disjoint sets n_1, n_2, n_3, n_4 are the nodes themselves. The P_1, P_2, P_3, P_4 disjoint sets (as of $P = P_1 \cup P_2 \cup P_3 \cup P_4$) are resembling the edges in the hypergraph, as one hyperedge is responsible for one degree on a node it contains.

When generating the hypergraphs, the h-ABCD model creates communities, and assigns the nodes to these communities, as seen in Fig. 8. The nodes are considered one by

one going from the node with the smallest degree to the one with the largest degree in the generated graph. Node v_i is assigned to a group with a probability proportional to the number of available spots left in the community. The community sizes are generated randomly following a power-law distribution with variables (β, s, S), as β is the power-law exponent, and (s, S) are the minimum and maximum community sizes respectively. Every node will belong to one group (leaving no nodes without a community), but to one group only, making the groups disjoint. These communities composed by the h-ABCD algorithm will be our baseline for the community detection, later expounded in Sect. 2, as it gives us a standpoint for each generated hypergraph, with every node within the graph assigned to a community.

The level of noise, guided by the ζ parameter, will affect the number of neighboring nodes of node v (nodes that are part of the same hyperedge as v is) to be in the same community (generated by the h-ABCD model, seen in Fig. 8) as node v. So if $\zeta = 0$, each hyperedge will be a so-called *community hyperedge*, meaning that the majority of its nodes will belong to one community, and by raising the value given for this parameter, the number of *community hyperedges* will decrease. The modularity and homogeneity of the *community hyperedges* can also be tweaked with the W parameter.

The matrix $\mathbf{W}$ parameter has 3 standard options implemented in the code, called the *majority*, the *linear*, and the *strict* model, but it can also be tuned to meet a unique preference. The distribution of the desired composition of the hyperedges is reflected by the elements of matrix $\mathbf{W}$; $\mathbf{W}_{c,d}$ – with $d \in |E|$ (c being the communities and $|E|$ being the number of edges). For my hypergraphs, I have used the implemented *linear* option. Generally for the same level of noise, generating *strict* hyperedges yields higher modularity, while the *majority* model yields the smallest, finally, the *linear* model lies in-between.

The modularity Q – first introduced by Newman and Girvan [18] – is used to measure the strength of a community structure in a network. It is formulated as $Q = \sum_{s=1}^{m} [\frac{l_s}{|E|} - (\frac{d_s}{2|E|})^2]$, where m is the number of modules (or in our case, communities), ls is the number of edges inside module s, $|E|$ stands for the number of edges in the network, and d_s denotes the total degree of the nodes in module s. The algorithm uses the formula as a base, modified for enabling its use on hypergraphs instead of simple graphs. Despite the difference in modularity for the 3 implemented models, generally the same number of *community hyperedges* are generated between them for the same level of noise, so the frequency of *community hyperedges* does not, but their homogeneity differs. The homogeneity of a hyperedge resembles the fraction of nodes that belong to one community within that hyperedge. The *linear* model works with modularity and homogeneity levels between the *majority* and the *strict* models, with its community size distribution closely following a power-law distribution (better than the other implemented models), seen in Fig. 7.

Vector q contains probability values for generating hyperedges of cardinality C_i, with $C_i = 1, 2, ..., |q|$ being the cardinality of a hyperedge and $i = 1, 2, ..., |q|$ being the index of the probability value within q. So the index of the value within q gives the probability (percentage) of generating a hyperedge with the said number of nodes within, so in simple terms the *1st* value within q gives the probability of a hyperedge with a cardinality of 1, the *2nd* value within q gives the probability of a hyperedge with a cardinality of 2 and so on. Of course, as the values within vector q resemble probabilities, for the generator to work we have a rule of $\sum_i q_i = 1$, as the total probability must be 1 for these exclusive events.

In Figs. 5–7, I present the basic properties of the h-ABCD networks that can be tuned via the model parameters, and in Fig. 8 I show an example layout indicating the community labels in a small network generated by the h-ABCD model.

In Figs. 5–7, I averaged the results over 10 h-ABCD networks generated with the said parameters and their values being:

- the degree values, namely δ and D, which are the least and most degree sizes. I have used 3 and 30 respectively;
- the least and most community sizes, s and S, 10 and 100 respectively;
- the level of noise, ζ, for which I have used 0.15;
- the distribution of hyperedge sizes, vector q contains 1+8 values discussed in Eq. 2;
- W matrix, the distribution of hyperedge composition, is *linear*;
- and the power-law distribution exponents, γ and β, of degree distribution and of distribution of community sizes respectively, are 2.5 and 1.5.

The vector q for generating the distribution of hyperedge cardinality C, was composed of values from a slowly decreasing distribution. The first element of the vector (corresponding to the probability of hyperedges containing 1 node) has been fixed to 0.0, so no self-edges will be generated.

$$q = 0.0, 0.1786, 0.1633, 0.1479, 0.1327, 0.1173, 0.1021, 0.0867, 0.0714 \tag{2}$$

As seen in Fig. 5, the generator does have a freedom of choice regarding the variables it is given and working with, because a simple hypergraph should be generated, so loops and parallel edges are not allowed. To be precise, instead of the distribution for vector q shown in Eq. 2, the model yielded a distribution of

$$q = 0.0, 0.3897, 0.2188, 0.1418, 0.0950, 0.0653, 0.0437, 0.0280, 0.0177 \tag{3}$$

for the average of 10 generated hypergraphs. Although the algorithm has a bit of room to generate proper hypergraphs,for example the actual vector q – seen in Eq 2 – is in parallel to the values I used as an input – seen in Eq 3. It can be said to the other tuneable parameters as well. The hyperedge cardinality distribution, seen in the left panel of Fig. 5, follows a power-edge distribution, with minimum and maximum values at 2 and 9 respectively; and the community size distribution, seen in Fig. 7, follows a similar power-law distribution (with a lower exponent), with minimum and maximum values at 3 and 30 respectively, in correspondence with the input values seen in the list above 1.4. In the left panel of Fig. 7, the degree distribution of the hypergraphs is visible, having a 'tail' at the lower end of the spectrum, in match with the degree distribution of the simple graph representations seen in the left panel of Fig. 6. The right panel of Fig. 6 shows the distribution of the implemented edge wights, calculated by Eq. 1.

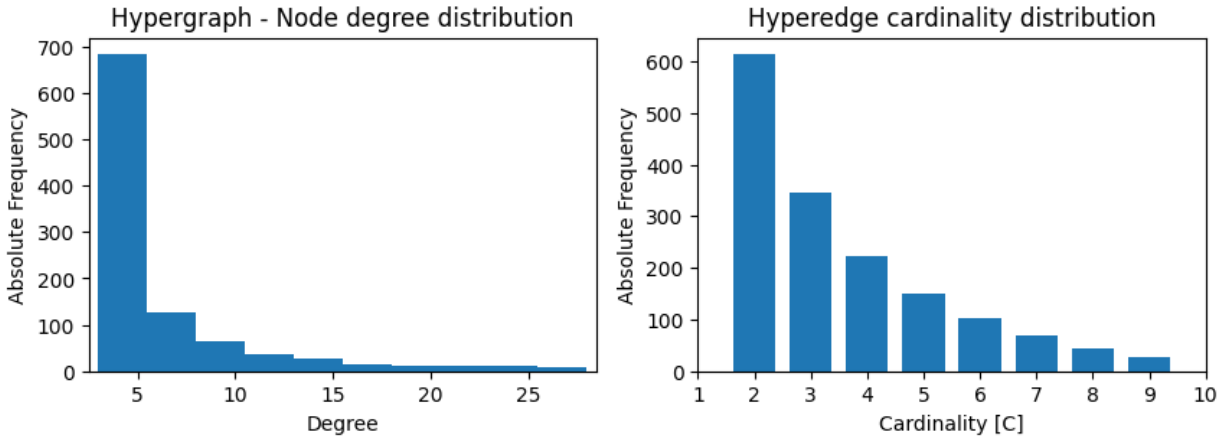


Figure 5: **Directly adjustable hypergraph properties in the h-ABCD model.** In the left panel, the distribution of the hyperedge node degrees is visible, with a 'tail' at the lower end of the spectrum, making most of the nodes low-degree. In the right panel the distribution of the hyperedge cardinality **C** is visible, which is a tuneable property via the vector q seen in Eq. 2. As it is visible, the model did follow a decreasing distribution for the vector q as given, but has a degree of freedom when generating the network (as seen in the actual average cardinalities for 10 networks in Eq. 3).

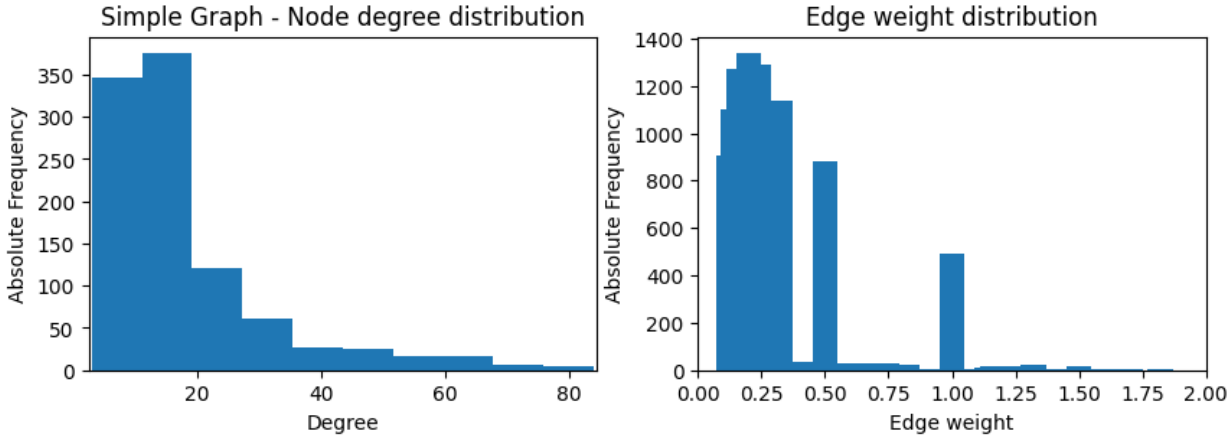


Figure 6: **Simple graph properties.** In the left panel the node degree distribution is visible for the simple graph, having a close resemblance to the node degree distribution for the hypergraph seen in Fig. 5, with a 'tail' at the lower degree nodes. In the right histogram, we can see the edge weight distribution where (with my implementation of the weight calculations seen in Eq. 1) I could reach an average number of more than 150 distinct weights in the networks generated for 1000 nodes, implying the various relationship levels between the objects of my network. It will help me to find the more important edges within the network, eventually resulting in stronger relationships and additional fallout nodes.

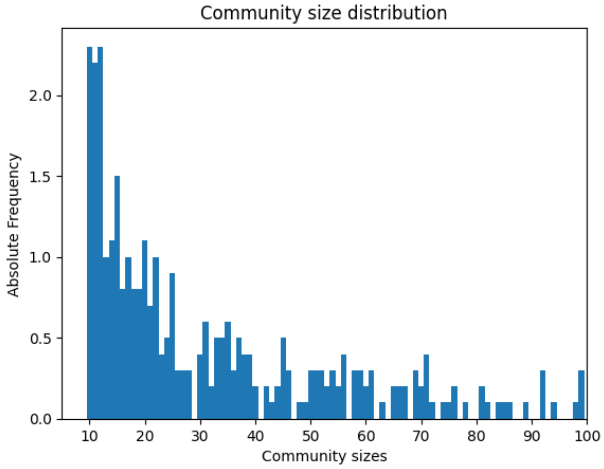


Figure 7: **The distribution of the community sizes averaged for 10 h-ABCD networks.** As it is seen in the figure the community sizes do follow a power-law distribution closely, with the minimum community size being 10 and the maximum being 100, the same as the parameters used by me, seen in the list above Fig. 5

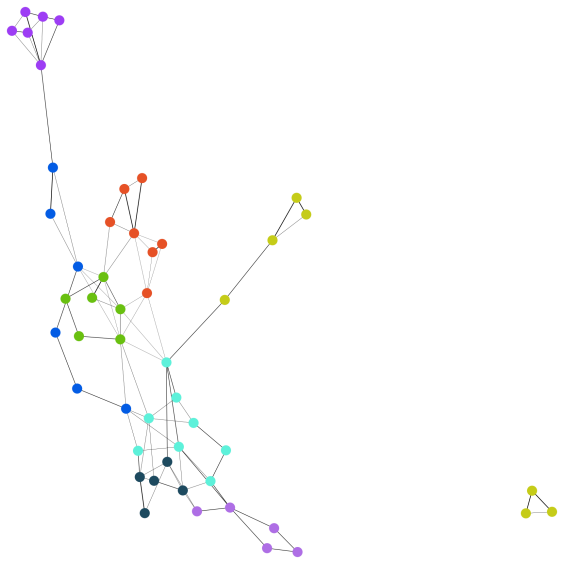


Figure 8: **The weighted graph representation of the h-ABCD hypergraph seen in Fig. 1 with a colouring that shows the planted communities.** This hypergraph has been generated with the use of the h-ABCD model, with the following parameter values; The hypergraph has $n = 50$ number of nodes, the degree values, namely δ and D, are 4 and 7 respectively; the least and most community sizes, s and S, 3 and 10 respectively; the level of noise, ζ, for which I have used 0.0; the distribution of hyperedge sizes, vector q is $q = (0.0, 0.4, 0.3, 0.2, 0.1)$; the W matrix, the distribution of hyperedge composition, is *linear*; and finally, the power-law distribution exponents, γ and β, of degree distribution and of distribution of community sizes in this order are 2.5 and 1.5 respectively. I have used this 50-node hypergraph for visualizing the hypergraph and its bipartite- and simple graph counterpart in Fig. 1 and also for illustrating the groups (seen in Figs. 9 –11) found by the k-clique communities algorithm expounded in Sect. 2.1.

2 Detecting overlapping communities in weighted networks

Real networks typically contain parts in which the nodes are 'more connected' or are in a stronger relationship with each other than they are with the rest of the network. These more densely connected units – usually referred to as communities – are considered to be the essential structural units of real networks.

2.1 Community detection with k-clique percolation

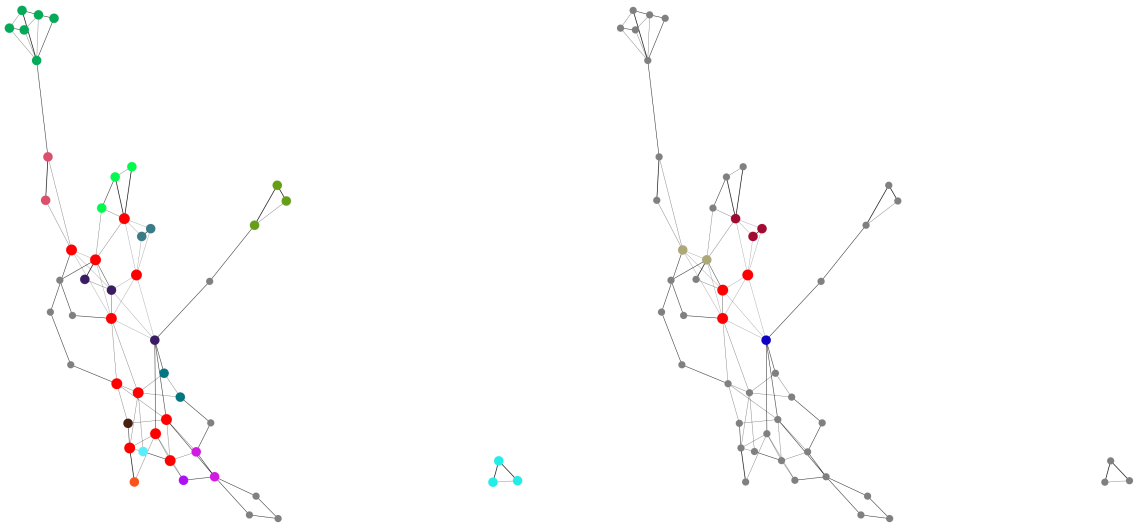
Clustering, or community detection is one of the most basic and ubiquitous methods to analyze data. Traditionally clustering meant separating the data elements into clusters, disjoint groups of close to equal sizes, but the problems and complications with this simplistic picture are becoming more prevalent, as new, more nuanced clustering methods surface, that reveal heterogeneous cluster size distributions, overlaps, and hierarchical structure.

I have used an implementation that is a more effective and deterministic method for the identification of communities and their overlapping within larger networks; the k-clique percolation model [19].

The central objects for the method are k-cliques, which are defined as complete sub-graphs [20], consisting of exactly k nodes. The algorithm considers these fully connected subsets of nodes to be in a more close relationship with each other than they are with other nodes. 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, or in other words if we can percolate through the system, hence the name k-clique percolation.

This is a "local" method: 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. A local community definition such as here circumvents this problem automatically.

If we look closely at our hypergraph and its simple graph counterpart, we quickly realize that one hyperedge completely corresponds to a k-clique, as all of the nodes within the hyperedge will be connected to each other within the simple graph representation, as seen in Fig. 1, and therefore form a complete sub-graph. Nevertheless, according to its definition, the community detection method builds on one single value of k at once, which has to be chosen by the user. Naturally, the method yields different community structures at different settings of k. As an example, in Fig. 9, the k-clique communities of two different k values are visualized for the unweighted simple graph representation for the 50-node hypergraph (the basic visualization of which is shown by Fig. 1).



(a) **k-clique communities with k=3** (b) **k-clique communities with k=4**

Figure 9: **Communities found with the k-clique percolation algorithm at different settings of the k parameter.** The coloring represents each community calculated by the k-clique community detection with a distinct colour for each group. The nodes which do not belong to a group – the 'fallouts' – are coloured grey and have a slightly decreased size, while the ones being in multiple groups – the communities they are part of are overlapping – are marked with colour red and are being represented with a slightly increased size compared to other nodes. By comparison, the groups generated by the h-ABCD model are shown in Fig. 8.

In its simplest form, the k-clique percolation algorithm, which will be used in my thesis, composes the cliques relying solely on the connections between the nodes, and not incorporating the weights of the edges. But of course, when looking at relationships, I have mentioned earlier in Sect. 1.3 that not only the connections themselves are important but also the 'level' of these connections, which can be represented through edge weights. An edge with more weight, calculated with my implementation seen in Eq. 1, based on the hyperedge size and the overlappings, is considered to represent a stronger relationship between the nodes it connects. So with these weight levels implemented, and by raising the minimum weight above which we consider our edges to be 'important' in the role of a network, we can potentially check which cliques will have the most closure, or which connections can be considered to be the strongest and which nodes or even connected components will be fallouts, with an edge weight insufficient to be considered important in the state of the network. This way the k-clique percolation model is available and ready to be applied to the simple graph representation of the hypergraphs at hand.

Although in its simplest form, the percolation itself is indifferent regarding the edge weights, as it only searches for completeness in the form of k-cliques, but as part of our processing of each group's importance for the whole network, the implemented weight levels and thresholds will play a major role. So after applying a certain weight threshold, and dropping the edges from our graph that fall below that level, our groups found by the k-clique percolation will be affected by both the edge weights and the size of the cliques we are searching for. As an example, Fig. 10 shows the output of the k-clique community

detection method at $k = 3$ when applied after weight thresholding. Compared to the number of communities depicted in Fig. 9a – that were found on the original graph of Fig. 8, considering all its edges –, the total number of groups visible in Fig. 10 – so found on the graph after the weight threshold has been implemented – are less, because of certain edges being cut off from a network, resulting in fewer connections and even making some nodes fallouts.

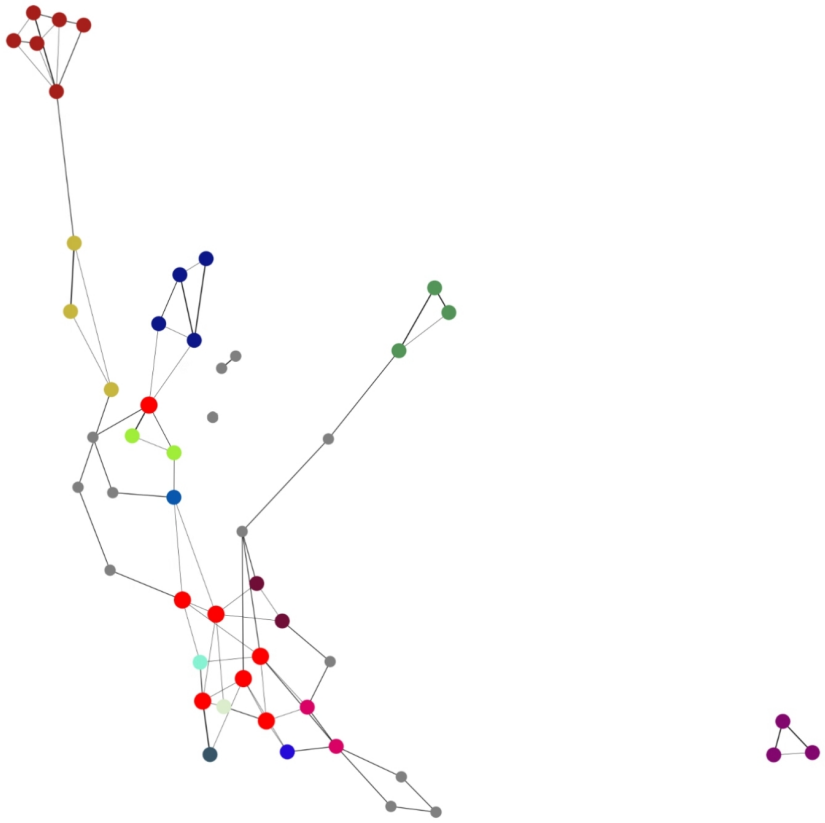


Figure 10: **The k-clique communities found at $k = 3$ after a weight thresholding.** Each composed group has a distinct node colour, while nodes in overlapping communities have a distinct red colouring with an increased node size. Un-grouped nodes on the other hand have a slightly smaller size with grey colouring. After implementing the weight levels and thresholds seen in Fig. 3, certain edges – those that fall below the threshold – will be dropped from the network, resulting in fewer connections. The number of groups found by the k-clique percolation will be less than in the unweighted version of the graph seen in Fig. 9a, but here we can consider the nodes within the cliques to be in an even stronger connection with each other than the ones seen in the unweighted graph, because of the minimum connection level, or edge weight they have to have between themselves.

But when we are considering splitting our network into smaller parts, reliant on the size of the groups, overlappings, and edge weights, how do we know if we are achieving a similar result, compared with which is embedded in the generator already, so we can state that our community detection was successful? The answer lies in the question, by calculating the similarities of the generated groups, with an implemented procedure presented in Sect. 2.2.

2.2 Evaluation of community detection performance

The most fundamental step towards understanding, evaluating, and leveraging identified clusterings is to quantitatively compare them. As there are numerous methods to effectively compare clusterings and withdraw information from the comparisons, it must be realized for each and every method, similarity measures must account for many other aspects of clusterings as well, such as the number of clusters, the size distribution of those clusters, multiple element memberships when clusters overlap, and scaling relations between levels of hierarchical clusterings. Arguably, these biases are maintained or exacerbated by extensions to accommodate overlapping or hierarchical clusterings, suggesting that most of the existing frameworks for clustering similarity are not adequate for comparing overlapping and hierarchically structured clusterings.

The element-centric framework for clustering similarity [21] naturally incorporates overlaps and hierarchy. The elements are compared based on the relationships induced by the cluster structure, in contrast to the traditional cluster-centric philosophy. This change in perspective resolves many of the aforementioned difficulties and avoids the common biases induced by irregular cluster structure.

The element-centric clustering similarity approach captures cluster-induced relationships between the elements through the cluster affiliation graph. Specifically, a cluster affiliation graph is constructed for a clustering $\mathbf{K}$ of labeled elements – the elements are the nodes of my graph, and the labels are the group labels from the community detection procedure – as a bipartite graph $\mathrm{B}(N \cup \mathbf{K}, \mathbf{R})$, where one node set corresponds to the original elements (the nodes of the graph) N and the other node set corresponds to the cluster set C. An undirected edge ($\subseteq R$) is placed between the element and the cluster if the node is part of the group. An element's membership in multiple overlapping clusters can be directly incorporated with multiple edges in the cluster affiliation graph.

This way, element-centric similarity can provide detailed insights into how two clusterings differ because the similarity is calculated at the level of individual elements. Specifically, the individual element-wise scores directly measure how similar the clusterings appear from the perspective of each element. The element-wise similarity S of an element n_i (resembling the i-th node in my network) in two clusterings $\mathbf{A}$ and $\mathbf{B}$ is found by comparing the stationary probability distributions $p_i^{\mathbf{A}}$ and $p_i^{\mathbf{B}}$, induced by the **PPR** processes on the two cluster-induced element graphs.

$$S_i(\mathbf{A}, \mathbf{B}) = 1 - \frac{1}{2\alpha} \sum_{j=1}^{N} \left| p_{ij}^{\mathbf{A}} - p_{ij}^{\mathbf{B}} \right| \tag{4}$$

Value α controls the influence of overlapping and hierarchical clusters with shared lineages. $p_{ij}^{\mathbf{A}}$ and $p_{ij}^{\mathbf{B}}$ are the **PPR** (**P**ersonalized **P**age**R**ank) affinites from element n_i to all elements n_j – found as the stationary distribution of a diffusion process – for clusterings $\mathbf{A}$ and $\mathbf{B}$. The final element-centric similarity score $S(A, B)$ of two clusterings $\mathbf{A}$ and $\mathbf{B}$ is the average of the element-wise similarities, calculated as

$$S(\mathbf{A}, \mathbf{B}) = \frac{1}{N} \sum_{i=1}^{N} S_i(\mathbf{A}, \mathbf{B}). \tag{5}$$

The possible results range from 0 (no matching) to 1 (same clusters element-wise).

The two clusterings to be compared are the ones coming from the h-ABCD generator, and the other being detected by the k-clique percolation model.

As explained in Sect. 1.4, the h-ABCD model distributes the nodes into groups by the given parameters. The assignment of a node into a community will be called admissible if the assignments (minimum and maximum community size, power-law exponent for the distribution, and node number) are satisfied. Each and every node will belong to a community but one community only, so overlappings and fallout nodes are not permitted.

The k-clique percolation on the other hand searches for complete subgraphs with k nodes within our graph, leaving some nodes without groups, while others will belong to multiple ones, generating overlappings within the groups along the way. Luckily, the element-centric clustering similarity approach works fine with overlapping clusters, but we still have the issue of the nodes without groups, the fallouts. As the similarity measure compares two clusterings element-wise, it is a must that the two clusterings have to have the same number of elements to be comparable. In our case, this means that we need to assign a group label to each node, the fallouts as well.

The two simplest approaches to this problem would be the following: the distinct nodes could have a group of their own, or the other way being the nodes left without groups will be joined as one big group of fallouts. However, neither packing the fallout nodes into one big group – even those that do not have any connection after the weight thresholds are implemented – nor stating that a fallout node is in its own group alone seems like the proper solution for composing a community structure. A more exact and complex solution is that after certain edges have been dropped from the graph and the groups have been found, we consider each connected component of the remaining nodes and edges a group of their own, individually.

All things considered, I compose the communities of a network in three phases, the procedure explained in Alg 1.

Algorithm 1 Three-step community detection

1. I use a weight threshold to remove the edges that are considered as 'weak' connections. Then groups are searched for in the remaining graph with the k-clique method.
2. All the nodes which were either dropped after implementing the weight threshold or were left out of the k-clique communities from Step 1. will form a Sub-graph of the original graph. The groups within this sub-graph will be determined by the connected components.
3. If there are nodes, which were not assigned to any groups in steps 1. and 2., then they will form a group of their own, individually, as they are treated as complete fallouts.

This way, instead of having a single vast group of outliers, they can be put into distinct groups, to make the clustering more precise, now accounting for both the edge weights and the direct connection between the nodes. As an example, Fig. 11 presents the output of this three-step community detection procedure in the case of the network of Fig. 8.

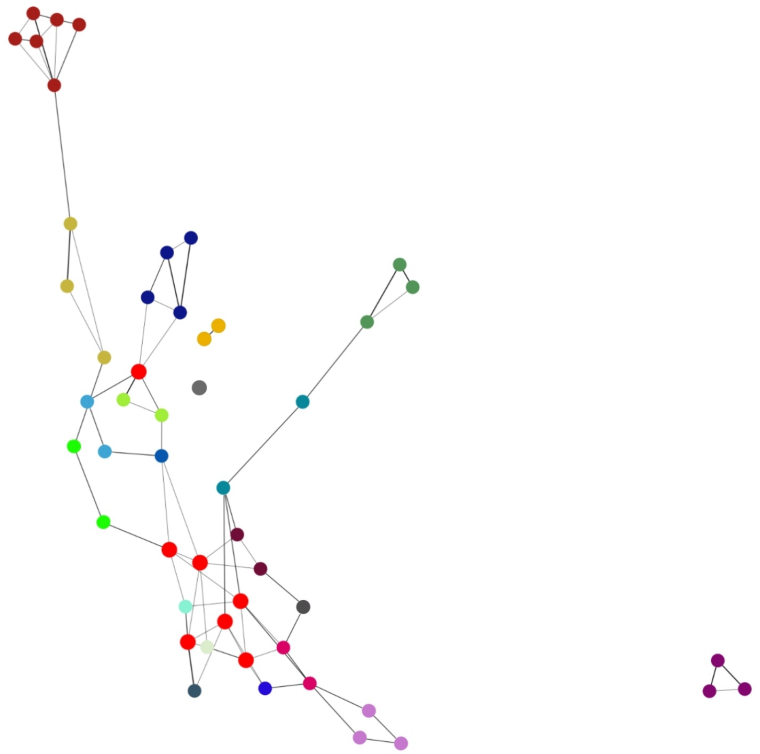


Figure 11: **Community structure found by the three-step community detection algorithm.** After my implementation of the community detection (the steps of which are explained in Algorithm 1) more nodes have been assigned to disjoint communities, instead of having all the fallout nodes make up communities by themselves or get assigned to one vast group of fallouts. Nodes classified to more than one group (i.e., nodes where communities overlap) are denoted by red colouring and increased node size. The planted community structure generated by the h-ABCD model is shown in Fig. 8.

In order to validate my method for grouping the nodes that are not classified by the k-clique method, I compared the partition shown in Fig. 10 – first by assigning the same community label to all the fallouts, and secondly assigning all the fallouts into disjoint groups (one node per group) – with the community structure shown in Fig. 11, where I used my technique to partition the outliers. The comparison was done by measuring their element-centric similarity scores to the communities created by the h-ABCD model (seen in Fig. 8).

This way I can represent the enhancements made – as I am searching for communities on the weighted representation after a threshold has been implemented – with element-centric similarity scores. The final element-centric similarity scores S of clusterings $H, K1, K2, E$ (**h**-ABCD communities; **k**-clique percolation after weight thresholding with the fallouts as one big group and with the fallouts as disjoint groups containing only themselves; and the **E**nhanced three-step community detection, respectively) are the following:

- $S(H, K1) = 0.4382$
- $S(H, K2) = 0.4247$
- $S(H, E) = 0.4867$

So my solution for the clustering, by reviewing the fallout nodes in search for connected components and assigning groups to them as well, yields a visibly better similarity when compared to the community structure generated by the h-ABCD model, than the simple k-clique percolation model with leaving all un-grouped nodes as a vast group of fallouts or in distinct groups only containing themselves. Thus, throughout the following section that presents my measurements regarding a phase transition in the detected community structure of h-ABCD graphs, I will use my three-step community detection algorithm described in Algorithm 1.

3 Percolation phase transition

In physics, a phase transition [22] (or phase change) is the physical process of transition between one state of a medium and another. Phase transitions have several universal features. During a phase transition of a given medium, certain properties of the medium change as a result of the change of external conditions. The identification of the external conditions at which a transformation occurs defines the phase transition point.

I will study the percolation transition [23, 24] within our model. Traditional percolation means 'walking' through the edges of the network and checking if we can get through the whole network – from any starting node to any node – like so. It is equivalent to the k-clique percolation method (explained in Sect. 2.1), because with traditional percolation we are 'walking' through a series of adjacent k-cliques with $k = 2$. In simple terms, if we can reach any node from any starting node, the network is one connected component, else it is formed from disjoint components.

Here, with the edge weight levels implemented and the weight thresholds applied, my expectation is that an inverse percolation transition will emerge. With inverse percolation-transition in the initial phase, the network contains a giant component, but with a well-defined removal of our nodes (weight thresholds), it is going to be completely split up. We can check whether our network consists of one vast component or numerous smaller ones, by percolating through it. This way the system is pushed towards the other phase of being split into isolated small components instead of a giant component. During the transition between the two phases, we can check the change in certain measures to show, how this transition affects the network as a whole.

As the system is getting close to the transition point, with the removal of edges labeled 'weak' regarding their weight and making the system break up into more isolated and smaller clusters, the system itself can show extremely large sensitivity with respect to outside perturbations that drive the system towards the other phase. The removal of just one link can induce the breakup of the giant component and turn our network into disjoint components. The quantity to measure the sensitivity of the system is called susceptibility and is denoted by χ. For macroscopic (or infinitely large) systems χ is divergent at the transition point. When studying percolation phase shifts, the usual definition for the susceptibility value [25] is seen in Eq. 6.

$$\chi = \sum_{s \neq s_{max}} \frac{s^2}{N_s}, \tag{6}$$

where s is the relative size of each community and $\sum_{s \neq s_{max}} N_s$ is the number of all communities (minus one, the biggest) within the graph. One important thing to note is the fact that the largest community should be left out of the formula for χ. If it would be included in the formula, we would get extremely large χ values everywhere in the phase, because of this large community. But we want a peak in χ at the transition point.

Another measure that goes through a change within the phase shift is the order parameter. The order parameter – denoted by O – is given by the relative size of the largest component, or in our case, the largest community, $O = \frac{s_1}{N}$, where s_1 is the size of the community at hand and N is the size of the network (or simply, the number of nodes). O can range from 0 in the disordered phase to 1 in the complete phase (when the network is fully connected so in one word, complete). When the order parameter is non-zero we call it the "ordered" phase. Within the inverse-percolation transition, the network will go from a complete phase to a total or at least almost disordered phase.

In this section, I am going to study the percolation phase shift within the networks generated by the h-ABCD model, first by introducing the parameters for the generating of networks in Sect. 3.1, and then the results for my measurements in Sect. 3.2.

3.1 Generated networks

Let me start off with the parameters for the h-ABCD generator. The details and ranges for the parameters can be seen in Table 1.

I have worked with the average of 10 distinct networks, each of those were generated using $\mathbf{n = 1000}$ nodes. The other network generation parameters were the following:

- the degree values, namely δ and D, which are the least and most degree sizes. I have used 3 and 30 respectively;
- the least and most community sizes, s and S, 10 and 100 respectively;
- the level of noise, ζ, for which I have used 0.0 and 0.3;
- the distribution of hyperedge sizes, vector q contains 1+8 values discussed in Eq. 7;
- W matrix, the distribution of hyperedge composition, is *linear* in every instance;
- and the power-law distribution exponents, γ and β, of degree distribution and of distribution of community sizes in this order were kept in the middle, 2.5 and 1.5 respectively.

When modifying the q parameter, I fixed the first element of the vector (so the probability of hyperedges containing 1 node) to 0.0, so no self-edges will be generated. The additional 8 values were generated in a slowly decreasing format:

$$q = 0.0, 0.1786, 0.1633, 0.1479, 0.1327, 0.1173, 0.1021, 0.0867, 0.0714 \tag{7}$$

3.2 Measurements

I measured the order parameter, the susceptibility, and the element-centric similarity as a function of the weight threshold on 10 networks in the case of both of the examined parameter settings. Figure 12 shows the average and the standard deviation of the quantities measured on the 10 networks of noise level $\zeta = 0.0$, while Fig. 13 refers to the 10 networks generated at a higher noise level of $\zeta = 0.3$.

As explained in Sect. 1.4, by raising the noise level – guided by the parameter ζ – the number of community hyperedges will decrease, meaning that the community structure will be more scattered, hence making the composure of the community structure harder. This is visible within the values of the element-centric similarity scores, shown in the third rows of Figs. 12 and 13, as the values for this measure peaked at

- $max(S) = (0.179, 0.194, 0.185)$ for $\zeta = 0.0$ and
- $max(S) = (0.119, 0.137, 0.131)$ for $\zeta = 0.3$

at $k = (3, 4, 5)$, respectively. I.e., as expected, the element-centric similarity values are higher for the lower noise level, showing the differences in the homogeneity of the community structure.

The values for noise levels show us that although with a higher ζ value, the community structure is quite different, but the drop in the order parameter O (shown it the first row of Figs. 12 and 13) and the peak within the susceptibility χ (shown it the second row of Figs. 12 and 13) is visible still, and the phase-transition is happening closely at the same weight level threshold in both instances. Also, interesting information is that the optimal clique size k for community detection – yielding the highest element-centric similarity score peak – seems to be $k = 4$ at both settings of the noise level, showing that even with a less homogeneous community structure, the same k value is found as the most favorable.

So with the weight thresholds implemented, and certain edges – ones considered to form connections not strong enough in the network – removed, the network falls to smaller pieces, and instead of a huge community (like seen in the top left panels in Figs. 12 and 13, at low edge weight thresholds, the network was one huge community for $k = 3$) with an addition of smaller overlapping ones, it will consist of smaller mostly disjoint communities, shown both by the susceptibility measure and order parameter. This way, the implementation of weight thresholds does help the community detection, as at a certain weight level, the community structure reached the peak element-wise similarity. An important result is that this peak in the element-wise similarity scores is in parallel with the peak in the susceptibility indicating the phase shift within the network.

It is quite visible that within my measurements I was able to achieve a reverse percolation phase transition. It is shown by the peak in the susceptibility for my networks as well as the drop in the order parameter. The change in these two measures fit together perfectly, as when the susceptibility value χ has its peak – visualized as a golden vertical line in Figs. 12 and 13 –, the order parameter O has a huge drop. The most visible peak is seen when I am using the k-clique percolation (as part of the enhanced three-step community detection procedure explained in Algorithm 1) with $k = 4$.

An intriguing result is that a peak emerges in the element-centric similarity scores as well. Although the emergence of this peak is regardless of the k value within each community detection, but the value at this peak does rely on the clique sizes. I got the highest peak with $k = 4$ for the k-clique percolation as part of the enhanced three-step community detection procedure, so the same clique size seems to be the best from the viewpoint of the element-centric similarity that yields the most apparent phase transition based on the curves of the order parameter O and the susceptibility χ. Also, the peaks in the element-centric similarities fall quite close to the peak measured in the susceptibility value, suggesting that setting the weight threshold based on the peak in the susceptibility curve is a good strategy, because – in contrast to the element-centric similarity – the χ can be measured even in cases when the ground truth community structure is not known.

This result also solidifies an answer to a question previously asked by Gergely Palla, Tamás Vicsek, Imre Derényi, and Illés Farkas in Ref. [26]. The recommendation was, to use the weight threshold level at which the peak susceptibility is measured, so at the critical point of the phase shift, to properly uncover the intricate community structure of networks. As if we would like to uncover the community structure of a network with the help of the k-clique percolation and we are unsure at what weight threshold should we do the process, the critical point of the phase shift gives us a standpoint for choosing a proper threshold level, perfect for community detection and evaluation. Although this theory has not been proved within the original paper, it has been answered in Ref. [25], as the calculations have shown that at the critical point of the phase shift, a peak is visible within the modularity measures – just like the element-centric similarity measure. My calculations

– working with weighted simple graph representations of hypergraphs – show that for different networks generated and different k values, at the point of peak susceptibility, the element-centric similarity scores have a peak as well, making the community detection yield the best result at that point, not only solidifying the previous calculations seen in Ref. [25] but also extending them to hypergraphs.

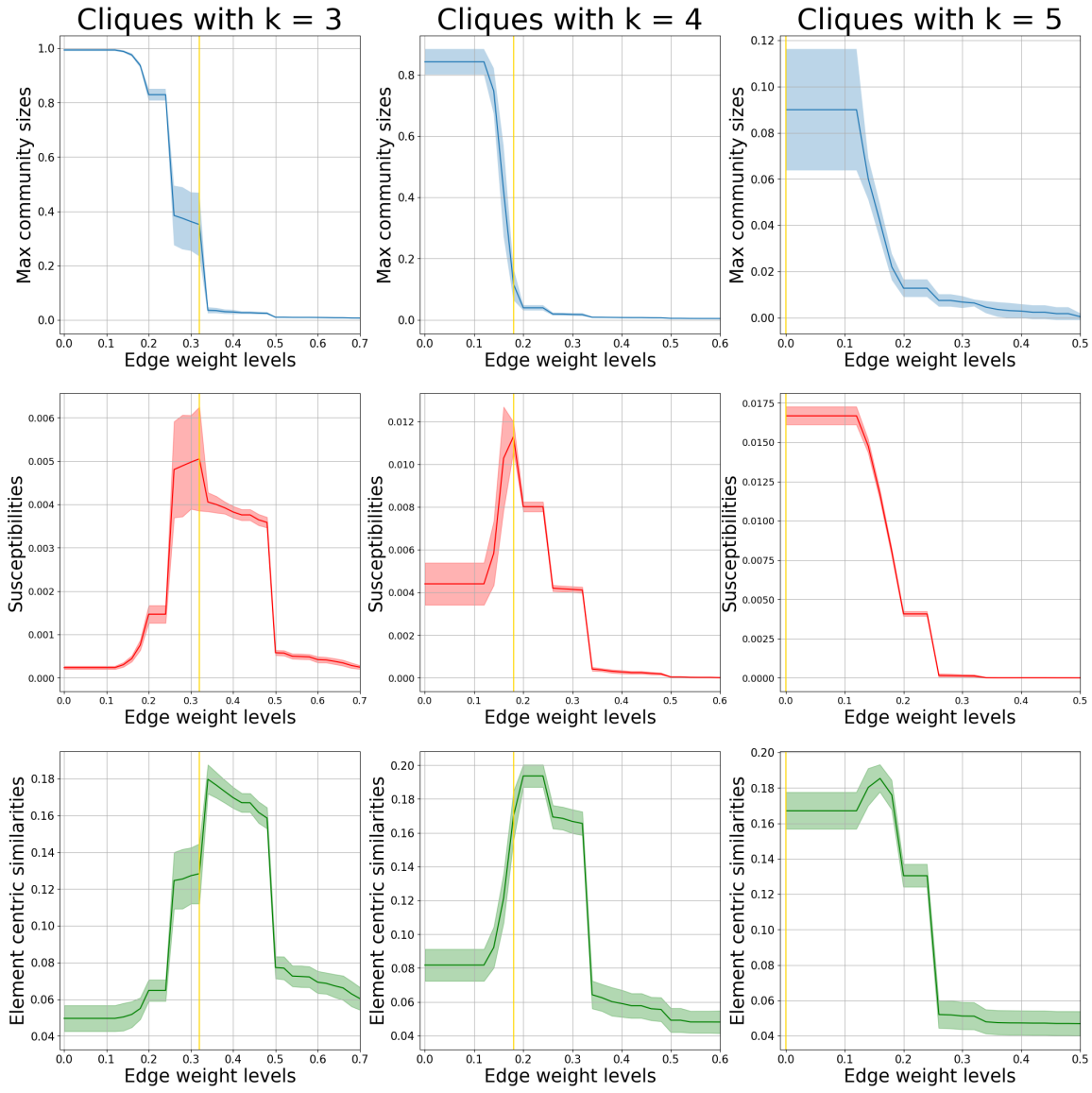


Figure 12: **The characteristics of the community structures detected in h-ABCD graphs of noise level $\zeta = 0.0$ using the three-step community detection method that includes a weight thresholding.** The network generation parameters are described in the list at the end of Sect. 3.1. The different columns refer to different k values used in the k-clique percolation algorithm within the first step of the implemented three-step community detection procedure explained in Alg. 1. The different rows correspond to the different measures that were examined: the first row shows the maximum relative community sizes – i.e., the order parameter O –, the second row presents the susceptibilities χ – calculated according to Eq. 6 –, while the third row depicts the achieved element-centric similarity scores $S(H, E)$, as a function of the applied weight threshold. The solid lines depict the average over the generated 10 networks, and the shaded areas show the standard deviation among the 10 networks.

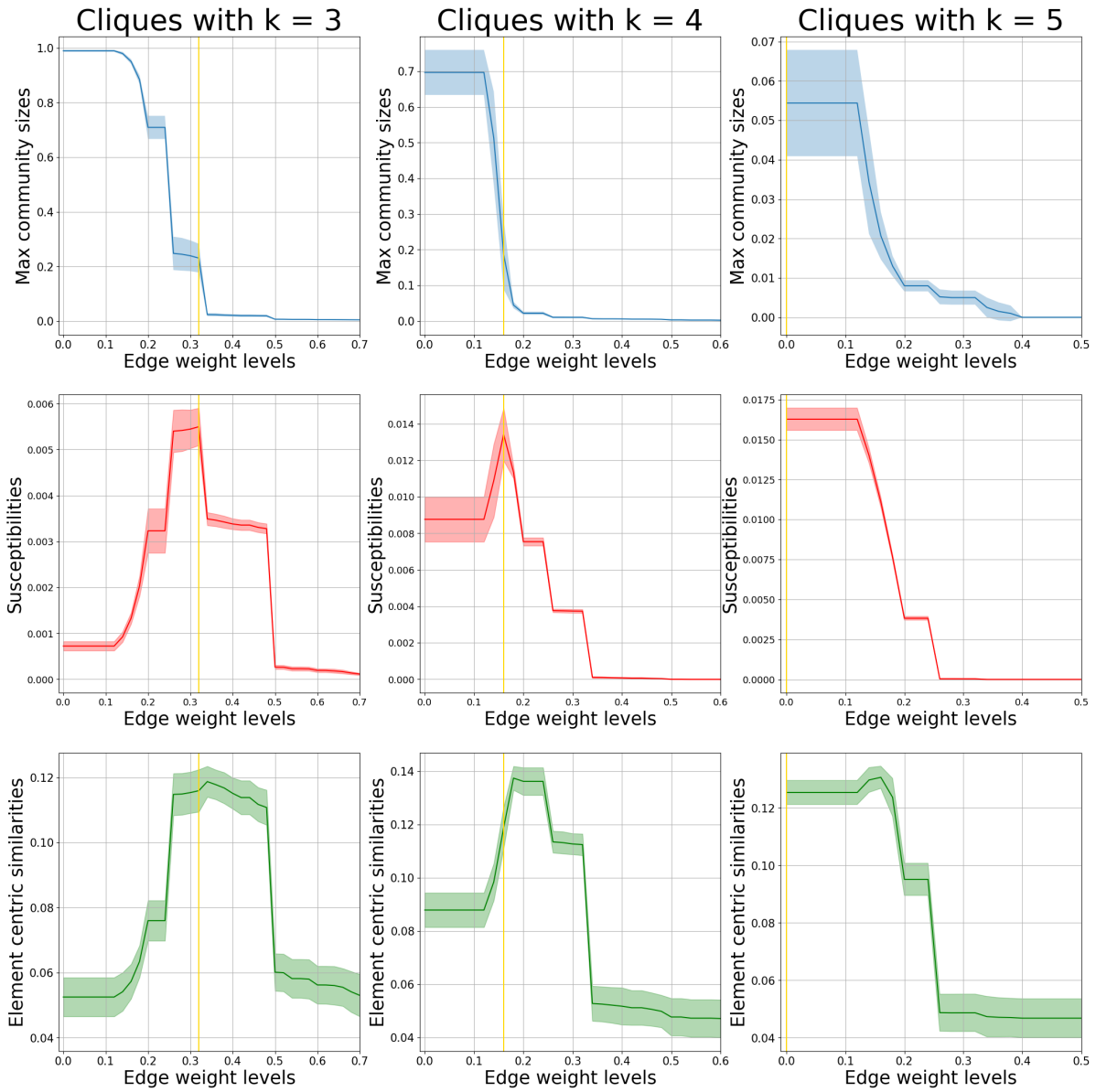


Figure 13: **The characteristics of the community structures detected in h-ABCD graphs of noise level $\zeta = 0.3$ using the three-step community detection method that includes a weight thresholding.** The network generation parameters are described in the list at the end of Sect. 3.1. The different columns refer to the different k values used in the k-clique percolation algorithm within the first step of the implemented three-step community detection procedure explained in Alg. 1. The rows correspond to the measures examined: in the first row, the order parameters O; in the second row, the susceptibilities χ – calculated according to Eq. 6 –; and in the third row, the achieved element-centric similarity scores $S(H, E)$ are plotted, as a function of the applied weight threshold. The solid lines depict the average over the generated 10 networks, and the shaded areas show the standard deviation among the 10 networks.

Conclusions and outlook

As real-life problems and complex systems can be easily understood and studied by being represented as networks, Network Science is rapidly gaining attention. Most of the problems at hand were represented as simple graphs, utilizing pairwise relations to show the connection between the objects in question, but more complex problems require complex solutions, while the compactness of the model is also a helpful addition. The hypergraphs representing higher-order systems help us with most of these issues while maintaining a compact and information-packed structure. In this work, I have generated hypergraphs with the h-ABCD model and studied the community structure of their weighted simple graph representations. By altering external conditions – namely, by applying different edge weight thresholds –, I aimed at the analysis of the crucial behaviour of the community structure in response to a gradual sparsification of the network through weight thresholding.

My implementation for the community detection within the weighted simple graph representation of the generated hypergraphs utilized the edge weight thresholding and the k-clique community detection in the first and second parts, but also relied on the nodes not being allocated into communities during these steps. I classified these fallout nodes into groups too, treating each connected component in the sub-graph formed by the fallout nodes as a community. And if after these steps, some nodes still remained without a community, they have been labeled as a community of their own.

As my measurements show in Sect. 3, the community structure of the h-ABCD hypergraphs undergoes a phase transition-like change when larger and larger fraction of the edges get removed. The order parameter of the network – given by the relative size of the biggest community – goes through a visible drop; while the susceptibility – calculated as the sum of the squared relative community sizes divided by the number of communities, leaving the biggest group out of the summation – has a well-defined peak in its values at certain weight thresholds.

According to my measurements regarding the element-centric similarity between the planted and the detected community structures of the examined hypergraphs, the phase transition point helps us to find the appropriate clique size k for the enhanced k-clique community detection and the proper value of the weight threshold, below which the connections can be considered to be weak enough to simply remove them from the network before the community detection. Namely, at a good setting of k, the susceptibility shows a sharp and high peak as a function of the weight threshold, and the maximum of this peak corresponds to a relatively good setting of the weight threshold.

As a part of later research, further enhancements, or in better words, refinements could be added to the community detection procedure. Although my algorithm is still working with fixed clique sizes, it still yields a better element-centric similarity score than the simple form of k-clique percolation when compared with the one embedded into the generative algorithm. This could be upgraded in favour of a more dynamic solution, always finding the biggest clique of the biggest possible size, and then, decreasing the clique sizes, finding more and more possible ways to percolate through the network and build a community structure. Besides, treating the hyperedges as natural cliques of

different sizes, the community detection could be performed not on the weighted simple graph representation but on the hypergraph itself, thereby taking into consideration the hyperedge cardinalities and edge overlappings as well in a simple manner, also by-passing the issue of certain nodes not being included in k-clique groups obtained in the weighted simple graph representation, because in hypergraphs, every node would be part of at least one hyperedge. Lifting the restriction of a single clique size and performing the community detection on the hypergraph would help in revealing more realistic community structures in hypergraphs, providing a better standpoint for analyzing the phase shift within representations of higher-order interactions.

Acknowledgements

At this point, I would like to thank the people who made the writing of this thesis possible. I would like to thank Dr. Gergely Palla for introducing me to the field of Network Theory and helping me out with all the information and knowledge needed for the thesis. I am also really thankful to Bianka Kovács for all of her help, the discussions and explanations, as well as the technical advice.

References

1. Stephen P. Borgatti, D. S. H. On Network Theory. *Phys. Rev. E.* https://doi.org/10.1287/orsc.1100.0641 (2011).
2. Mata, A. S. d. Complex Networks: a Mini-review. *Brazilian Journal of Physics* **50,** 658–672. ISSN: 1678-4448. https://doi.org/10.1007/s13538-020-00772-9 (2020).
3. Krioukov, D., Papadopoulos, F., Kitsak, M., Vahdat, A. & Boguñá, M. Hyperbolic geometry of complex networks. *Phys. Rev. E* **82,** 036106. https://link.aps.org/doi/10.1103/PhysRevE.82.036106 (3 2010).
4. ESSAM, J. W. & FISHER, M. E. Some Basic Definitions in Graph Theory. *Rev. Mod. Phys.* **42,** 271–288. https://link.aps.org/doi/10.1103/RevModPhys.42.271 (2 1970).
5. *A well written and comprehensive report can be read about simplicial complexes at https://www.math.uci.edu/~mathcircle/materials/MCsimplex.pdf.*
6. Devriendt, K. & Van Mieghem, P. The simplex geometry of graphs. *Journal of Complex Networks* **7,** 469–490. ISSN: 2051-1329. eprint: https://academic.oup.com/comnet/article-pdf/7/4/469/29161067/cny036.pdf. https://doi.org/10.1093/comnet/cny036 (Jan. 2019).
7. Salnikov, V., Cassese, D. & Lambiotte, R. Simplicial complexes and complex systems. *European Journal of Physics* **40,** 014001. https://dx.doi.org/10.1088/1361-6404/aae790 (2018).
8. Ciftcioglu, E., Ramanathan, R. & Basu, P. Generative Models for Global Collaboration Relationships. *Scientific Reports* **7** (June 2016).
9. Bretto, A. in *Hypergraph Theory: An Introduction* 111–116 (Springer International Publishing, Heidelberg, 2013). ISBN: 978-3-319-00080-0. https://doi.org/10.1007/978-3-319-00080-0_7.
10. Mulas, R., Horak, D. & Jost, J. in *Higher-Order Systems* (eds Battiston, F. & Petri, G.) 1–58 (Springer International Publishing, Cham, 2022). ISBN: 978-3-030-91374-8. https://doi.org/10.1007/978-3-030-91374-8_1.
11. Bretto, A., Cherifi, H. & Aboutajdine, D. Hypergraph imaging: an overview. *Pattern Recognition* **35.** Image/Video Communication, 651–658. ISSN: 0031-3203. https://www.sciencedirect.com/science/article/pii/S003132030100067X (2002).
12. Kamiński, B., Prałat, P. & Théberge, F. Hypergraph Artificial Benchmark for Community Detection (h-ABCD). arXiv: 2210.15009 [cs.SI] (2023).
13. *For generating the hypergraphs, I have used a generator called the Hypergraph Artificial Benchmark for Community Detection, the source code of which can be found at https://github.com/bkamins/ABCDHypergraphGenerator.jl.*
14. Zhou, T., Ren, J., Medo, M. c. v. & Zhang, Y.-C. Bipartite network projection and personal recommendation. *Phys. Rev. E* **76,** 046115. https://link.aps.org/doi/10.1103/PhysRevE.76.046115 (4 2007).

15. *For plotting the simple graph representations in similar manners, I used the Python function 'stochastic_block_model' available in the 'NetworkX' package at* ***https://networkx.org/documentation/stable/reference/generated/networkx.drawing.layout.spring_layout.html*** *.*

16. Lancichinetti, A., Fortunato, S. & Radicchi, F. Benchmark graphs for testing community detection algorithms. *Physical Review E* **78.** https://doi.org/10.1103%2Fphysreve.78.046110 (2008).

17. Bollobás, B. A Probabilistic Proof of an Asymptotic Formula for the Number of Labelled Regular Graphs. *European Journal of Combinatorics* **1,** 311–316. ISSN: 0195-6698. https://www.sciencedirect.com/science/article/pii/S0195669880800308 (1980).

18. Newman, M. E. J. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences* **103,** 8577–8582. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.0601602103. https://www.pnas.org/doi/abs/10.1073/pnas.0601602103 (2006).

19. Derényi, I., Palla, G. & Vicsek, T. Clique Percolation in Random Networks. *Phys. Rev. Lett.* **94,** 160202. https://link.aps.org/doi/10.1103/PhysRevLett.94.160202 (16 2005).

20. Moon, J. W. & Moser, L. On cliques in graphs. *Israel Journal of Mathematics* **3,** 23–28. ISSN: 1565-8511. https://doi.org/10.1007/BF02760024 (1965).

21. Gates, A. J., Wood, I. B., Hetrick, W. P. & Ahn, Y.-Y. Element-centric clustering comparison unifies overlaps and hierarchy. *Scientific Reports* **9,** 8574. ISSN: 2045-2322. https://doi.org/10.1038/s41598-019-44892-y (2019).

22. Bruce, A. D. & Cowley, R. A. in *Structural phase transitions* (1981).

23. Sahimi, M. in *Encyclopedia of Complexity and Systems Science* (ed Meyers, R. A.) 6538–6545 (Springer New York, New York, NY, 2009). ISBN: 978-0-387-30440-3. https://doi.org/10.1007/978-0-387-30440-3_387.

24. Saberi, A. A. Recent advances in percolation theory and its applications. *Physics Reports* **578.** Recent advances in percolation theory and its applications, 1–32. ISSN: 0370-1573. https://www.sciencedirect.com/science/article/pii/S0370157315002008 (2015).

25. Tóth, B., Vicsek, T. & Palla, G. Overlapping Modularity at the Critical Point of k-Clique Percolation. *Journal of Statistical Physics* **151,** 689–706. ISSN: 1572-9613. https://doi.org/10.1007/s10955-012-0640-5 (2013).

26. Palla, G., Derényi, I., Farkas, I. & Vicsek, T. Uncovering the overlapping community structure of complex networks in nature and society. *Nature* **435,** 814–818 (June 2005).