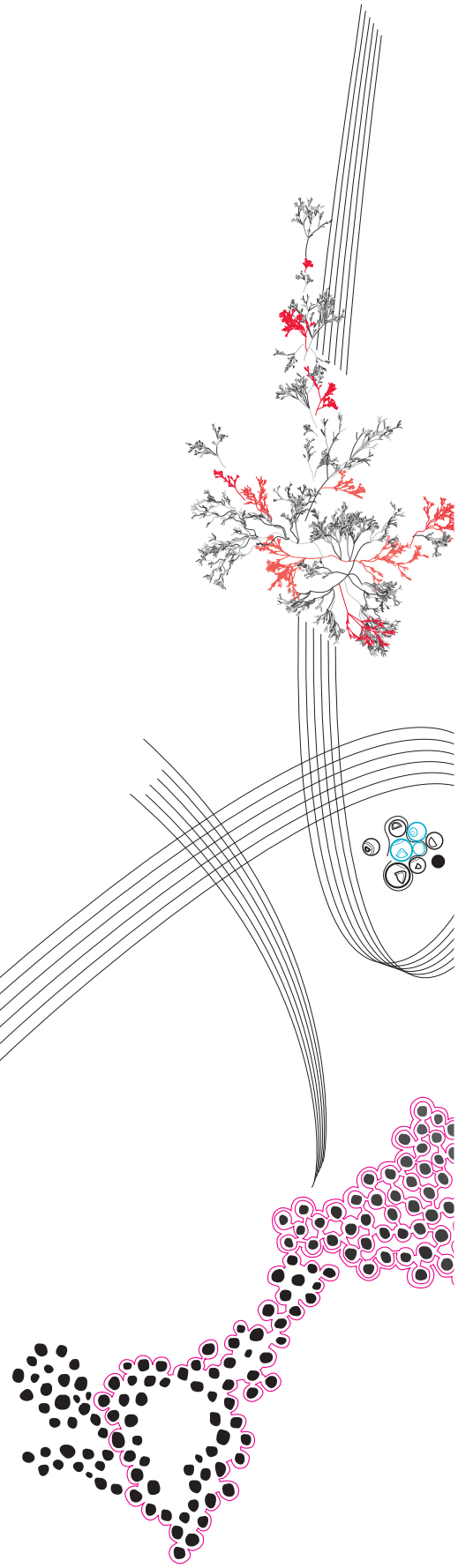


MSc Thesis Applied Mathematics

Heterogeneous Hypergraphs Models for Collaboration Networks: A Bridge between Mathematics and Business

G.A.W. Moltman

Supervisor: Dr. J.M. Meylahn

August, 2023

Department of Stochastic Operations Research
Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Acknowledgements

I would like to thank a few people that have contributed to this thesis. First, I would like to thank dr. Ariane von Raesfeld for initiating the idea of analysing collaboration networks and for providing the data on R&D collaboration. I would like to thank Samuel Mok for his advise on literature sources and searching techniques. I would also like to thank prof. dr. Richard Boucherie and dr. Clara Stegehuis who have offered to be the second and third supervisor for this master thesis. Prof. dr. Richard Boucherie I also would like to thank for his supportive role as chair of the Stochastic Operations Research department at the University of Twente. Next, I would like to thank studyadvisor Lilian Spijker, family and friends, who helped me dealing with personal issues and always kept supportive during the master thesis process. Finally, I would like to thank dr. Janusz Meylahn for his role as first supervisor of this thesis. Especially, I would like to thank him for his compassion and his supportive feedback during the process of creating this master thesis.

Summary

In this thesis, data on university-industry collaboration networks of nanotechnology R&D is studied with three heterogeneous hypergraph models. In business, there are not many suitable methods to study collaboration networks. With the concept of hypergraphs we provide a good representation method for collaboration networks to business people. In this thesis, the following research question is central: How could mathematical hypergraph models be helpful in the field of collaboration networks in business? In order to answer this question, three hypergraph models are used to configure networks. These networks are constrained by a degree sequence and a dimension sequence. The models we investigate are two models by Chodrow (2020) and the Chung-Lu hypergraph model by Kamiński, Poulin, Prałat, Szufel, and Théberge (2019). Since the data consists of heterogeneous node types, the three models are changed to heterogeneous hypergraph models based on the idea of stochastic block models. Measures are used to compare the resulting networks of the different models. These measures are vertex degree, dimension, hyperedge degree, eigenvalue centrality, modularity and homophily. Three main tests are done in order to be able to answer the research question. First, the three models are compared among each other to test which model mimics the original collaboration network best. Second, cases are compared that might give conclusions about the advantages and disadvantages of using the hypergraph models instead of normal graph models. Finally, we tested whether adding the heterogeneity to the hypergraph models adds value.

The main results of this thesis are the following. First, we found that the NHHCM model is the best model when comparing the SHHCM, NHHCM and CLHHM, in the case that there is not a dominant node type available. Otherwise the SHHCM model performs best. In addition, we found that a normal graph is not a good representation of a collaboration network. It replicates some measures of the collaboration network, but a hypergraph does this better. Moreover, it could not be stated whether hypergraph models are better models than normal graph models. We also found that the heterogeneous hypergraph models significantly improve the original SHHCM, NHHCM and CLHHM. This is the case, even though the heterogeneous hypergraph models are based on equations that are only based on an intuition of heterogeneous graphs and are not mathematically proven.
.

Heterogeneous Hypergraphs Models for Collaboration Networks.

G.A.W. Moltman

August, 2023

1 Introduction

The theory of hypergraph models is an emerging field within complex networks. Hypergraphs models are able to capture additional features in comparison to general graph models (Battiston et al., 2021; Benson, Gleich, & Leskovec, 2016; Ghoshal, Zlatić, Caldarelli, & Newman, 2009), and have the potential to fruitfully be applied in many different fields. Particularly, when these hypergraph models are changed to heterogeneous hypergraph models with different node states. Nevertheless, many fields are not familiar with the concept of hypergraphs. This results mainly from the fact that hypergraph models are hard to understand due to the difficult mathematics involved and the use of inconsistent notation. This is especially true in business fields, since these fields are less familiar with abstract mathematics, even though hypergraphs might be particularly useful in these fields where networks of collaboration exists. In this thesis we will fill this gap by building a bridge between mathematics and business. The following research question is answered: How could mathematical heterogeneous hypergraph models be helpful in the field of collaboration networks in business?

In order to address this question, we first show which mathematical hypergraph models represent collaboration networks best. For this purpose different existing hypergraph models are compared and changed into heterogeneous hypergraph models with heterogeneous node types. The models will be evaluated using data on R&D collaboration. Furthermore, we show how these models could easily be applied in business by providing clear notation, accessible hypergraph models and results for business. In order to make this thesis accessible for people with little mathematical knowledge, special sections are added that should clarify mathematical expressions. It is recommended to continue reading at Section 2.6, when it is preferred to avoid mathematical issues.

2 Theory

2.1 Complex Networks

To begin, we make a distinction between three main concepts: graphs, networks and complex networks. A graph G is a combination of a vertex set V and an edge set E, $G = (V, E)$, where vertices are objects and an edge is an unordered or ordered pair of these objects, dependent on if the graph is undirected or directed, respectively (Van der Hofstad, 2016). The graph could also be represented as an adjacency matrix A, a matrix with zeros and ones, where $A_ij = 1$ and $A_ij = 0$ indicates that vertex v_i and v_i are connected or not. The definition of a network is less clear. In short, a network is "A collection of points joined together in pairs by lines" (Newman, 2010). A network is often referred to as a graph with

attributes (names) for the vertices or edges. In networks vertices are often called nodes and edges are called links. Finally, a complex network is a large-scale network which in its simplest form could be realized by a random graph (Albert & Barabási, 2002).

For mathematicians the term "graph" is most appropriate, while the term "network" is most used in applied fields. In essence, definitions of graphs are equivalent: a graph consists of a vertex set and an edge set. However, definitions differ in length and focus. As an example, the definitions for graphs of Van der Hofstad (2016) and Chodrow (2020) are rather extended, where Van der Hofstad (2016) focuses in his definition on whether edges are directed or undirected and Chodrow (2020) focuses on explaining selfloops, parallel edges and the underlying set structure of the vertex set and the edge set. On the other hand, the definition of (Avin, Lotker, Nahum, & Peleg, 2019) is rather short with a focus on the underlying sets and a small remark on self-loops. After defining a graph, the notion of vertex degree is often introduced. In short, the vertex degree is the number of connections to other vertices, where a connection to itself (selfloop) counts twice (Avin et al., 2019; Chodrow, 2020).

Furthermore, an edge is sometimes valued based on the strength of the connection. This value is called an edge weight (Kim & Lee, 2015; Kuznetsov, Panov, & Yakovlev, 2020; Pan et al., 2012). Kim and Lee (2015) define a weighted graph as a combination of a vertex set V and a matrix w of edge weights. Matrices are also used by Battiston et al. (2020), Kuznetsov et al. (2020) and Pan et al. (2012) for representing edge weights. Edge weights could also be used to induce graphs (Bhagat, Cormode, & Muthukrishnan, 2011). In Figure 1, a general weighted graph is illustrated with vertices connected in a non-specific order. In practice, the vertices could, for example, represent authors and the edges whether authors published a paper together, with the weight as the number of citations of that particular paper in which the authors collaborated.

In this thesis, the convention is followed that a graph $G = (V, E)$ consists of a set V of vertices and a set E of edges, where the edges might be weighted. An edge e consists of a pair of unordered vertices, $e = (u, v)$, $u, v \in V$. An edge (u, u) is called a self-loop. The degree of edge v is defined as

$$d_v = \sum_{e \in E} \mathbb{1}(v \in e).$$

The step-function $\mathbb{1}(v \in e)$ equals one when a vertex v is in a edge e, and zero otherwise. A vector with the degree for each vertex is also referred to as the degree distribution of a network. In order to keep track of all the variables and abbreviations that appear in this thesis, in Table 20 in Appendix A the variables and abbreviations are listed in alphabetic order.

2.2 Heterogeneous, Hierarchical and Multi-Layer Graphs

When considering graphs where vertices differ, multiple graph models are appropriate. First, the heterogeneous graph or network could be used. Ghosh and Lerman (2019) define the task of a heterogeneous network as linking different types of entities. In addition, Zhang, Song, Huang, Swami, and Chawla (2019) describe heterogeneous graphs as graphs that "contain abundant information with structural relations (edges) among multi-typed nodes as well as unstructured content associated with each node" (p. 793). The term "heterogeneous networks" also appears in applied fields, for example in the field of mobile broadband, where a heterogeneous network is "a network built with a variety of different base station types" (Von Wrycza et al., 2015). So within a heterogeneous graph, the vertices are of different types.

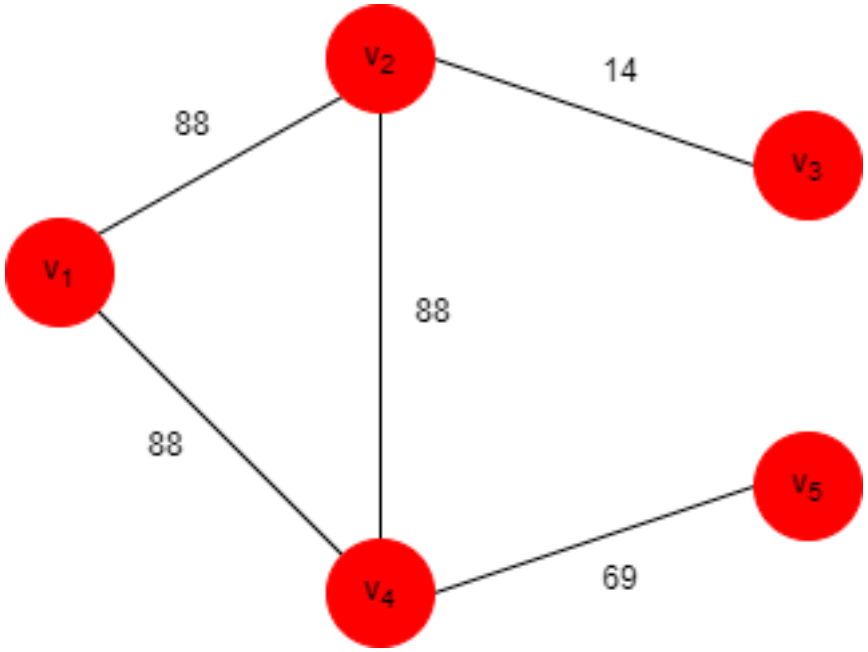
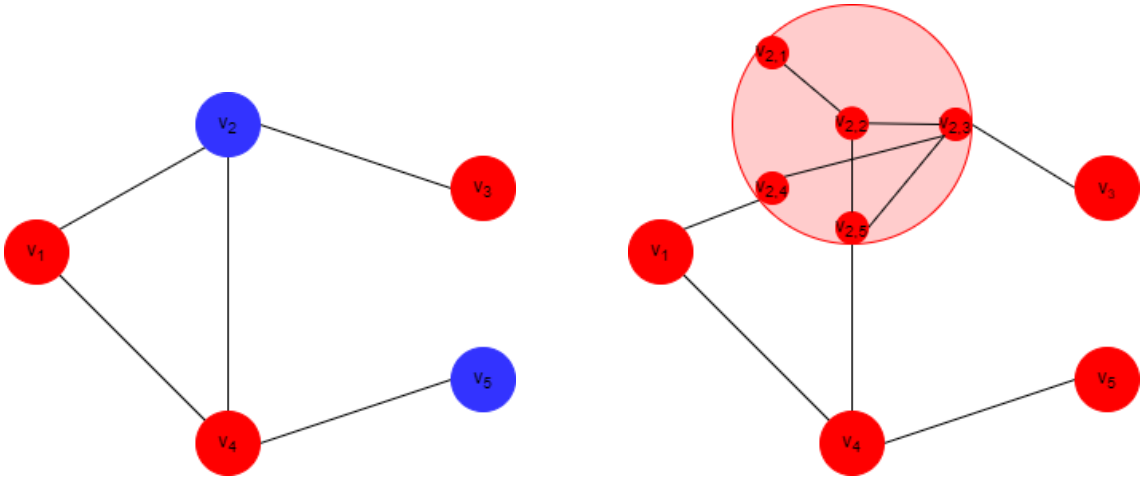


FIGURE 1: Weighted graph

Heterogeneous graphs are defined in several ways. On the one hand, Ghosh and Lerman (2019) describe a graph as a layered graph, where there are intra- and inter-layers between different types of entities. An intra-layer connects the same type of vertices and inter-layers connect vertices of different types. These layers are represented by matrices. On the other hand, Zhang et al. (2019) use the concept of a content-associated heterogeneous graph, where a graph $G = (V, E, O_V, R_E)$ in which besides the vertex set and edge set, there exists a set of object types, O_V, and relation types, R_E. The O_V defines per vertex its type, and R_E defines per edge which type of edge it is, meaning which types of nodes are connected. In Figure 2a an example of a heterogeneous graph is illustrated. When considering the vertices as authors who collaborate (through the links), one could distinguish between different types of authors. As an example, the blue nodes (v_2 and v_5) could represent Dutch authors and the red nodes (v_1, v_3 and v_4) could represent German authors.

In addition to heterogeneous graphs, the concept of hierarchical graphs is used in literature, also appearing as hierarchical structure or hierarchical graph structure. Kuznetsov et al. (2020) argue that a hierarchical graph structure contains vertices that consist of a set of one or more elements. In general, this could be extended to an n-level hierarchical graph (Mäkinen, 1990). Nevertheless, this is not always taken into account by proper definitions of a hierarchical graph (Busatto & Hoffmann, 2001; Drewes, Hoffmann, & Plump, 2002). The definitions as described by Busatto and Hoffmann (2001) and Drewes et al. (2002) consider a hierarchical graph as a triple, where there is an original graph, an underlying graph and a connection between these. On the other hand, hierarchical graphs are also described in terms of intra- and inter-community layers (Van der Hofstad, Van Leeuwaarden, & Stegehuis, 2017), as was done for heterogeneous graphs by Ghosh and Lerman (2019). When one wants to detect whether there are communities in a graph, often a hierarchical structure or hierarchical graph is used (Van der Hofstad et al., 2017; Blondel, Guillaume, Lambiotte, & Lefebvre, 2008). A hierarchical structure or hierarchical graph is also known as a hierarchical decomposition (Newman, 2010). As an example, in Figure 2b a hierarchical graph is drawn. In the example of collaborating authors, v_2 could represent an



(A) Heterogeneous graph (B) Hierarchical graph

FIGURE 2: The differences between a heterogeneous and hierarchical graph.

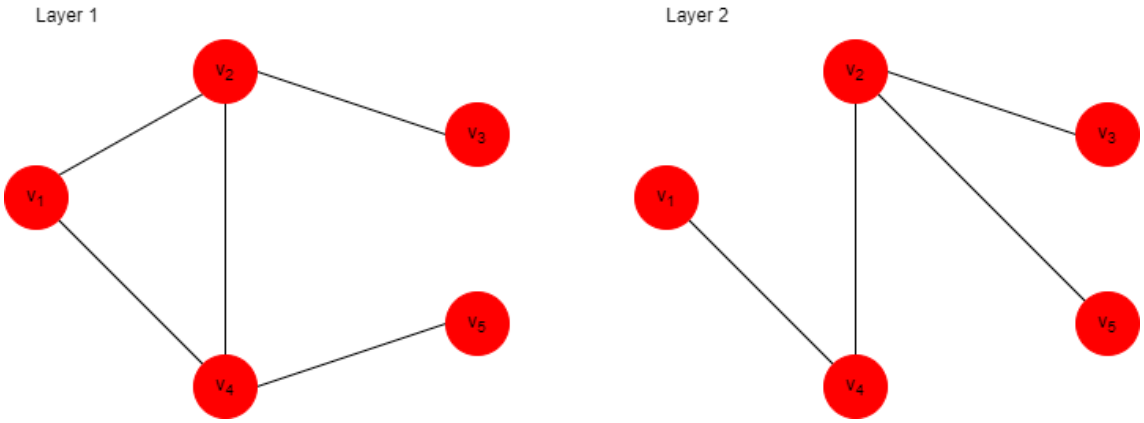


FIGURE 3: (Pillar) Multi-layer graph

author that consists of a group of people (research group or company), which have their own interrelationships.

Finally, the term multi-layered graph is closely related to heterogeneous and hierarchical graphs. In some cases, a multi-layer graph is defined as a set of vertices, where the edges in different layers differ (Dong, Frossard, Member, Vandergheynst, & Nefedov, 2012, 2014; Battiston et al., 2020). Kim and Lee (2015) refer to this as a pillar multi-layer graph, where a multi-layer graph in general is defined as a tuple of graph layers and an identity mapping. In this definition, it is also possible to split vertices into multiple vertices, which comes closer to the definition of a hierarchical graph. Furthermore, Kim and Lee (2015) state that these graphs could also be used to represent heterogeneous graphs. In Figure 3 a pillar multi-layer is illustrated as defined by Kim and Lee (2015). As an example, layer 1 could exist of authors that write articles together and layer 2 could represent friendships among authors.

In this thesis, a distinction between models for graphs with different types of vertices is made. Even though in this section a clear distinction is made between heterogeneous, hierarchical and multi-layer graphs, literature on these types of graphs is ambiguous. In addition to these three terms, other terms are used interchangeable.

2.3 Random Graph

Previously, we stated that a complex network is a large-scale network which in its simplest form could be realized by random graph (Albert & Barabási, 2002). Random graphs take many forms. In this subsection, three models are introduced (1) the generalized random graph (GRG), (2) the configuration model and (3) the preferential attachment model (PA-model). First, the idea of a GRG model is based on edges that exist based on a probability. For this purpose, within a GRG model the concept of vertex weights is introduced. In addition to edge weights vertices could also have weights, w_i, based on the importance of a vertex. Within GRG models, the probability that an edge exists between vertices i and j is denoted by (Van der Hofstad, 2016):

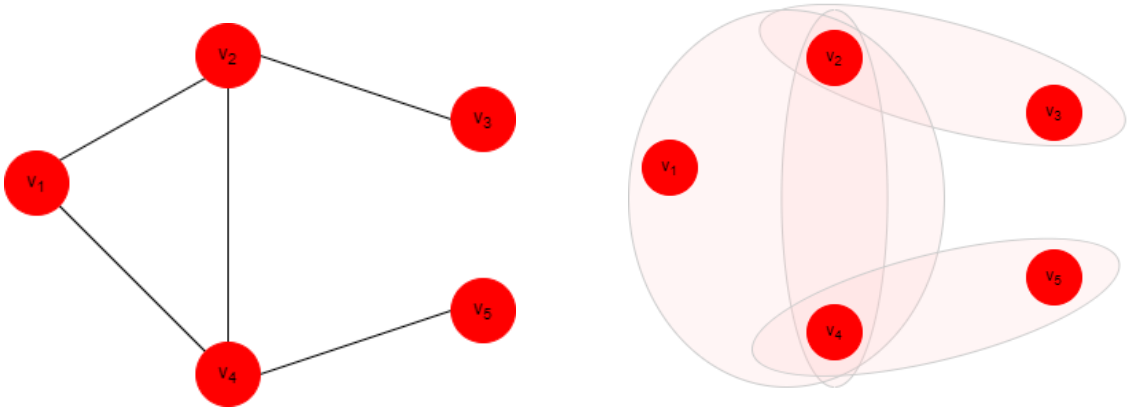
$$p_{ij} = \frac{w_i w_j}{l_n + w_i w_j}, \text{ where } l_n = \sum_{i \in V} w_i.$$

Second, Van der Hofstad (2016) describes the configuration model as a model for configuring graphs with a fixed degree distribution, where half-edges are uniformly matched. A half-edge is defined as the half of a normal edge, where each half-edge could be connected to another half-edge. Each vertex has a number of half-edges, equalling the degree of that specific vertex. The configuration model matches the half-edges of the vertices with each other one by one until all half-edges are transformed into complete edge. A half-edge is also referred to as stub (Chodrow, 2020). The configuration model is very suitable in replicating existing graphs, when you want to constrain the degree distribution. Third, the PA-model is described as a graph that starts with two vertices that are connected and at every time step a new vertex is added and connected or not connected to an existing vertex, based on the degree of the vertices. A vertex is connected to vertices with a high degree with higher probability. In this way the graph grows linearly in time.

Besides the standard random graph models, many other models have been developed. Chung and Lu (2002) present a GRG model in which the probability that an edge exists depends on an expected degree sequence. The Chung-Lu model is also often extended, for example by Kamiński et al. (2019). In addition, Chodrow (2020) introduces the configuration model by adding the definition of a stub-labeled graph. Furthermore, Chodrow (2020) gives an algorithm to remove selfloops from graph. The PA-model is also often elaborated on. Avin et al. (2019) defines a model which considers in addition to the vertex-step an edge step. This allows them to add only edges between existing vertices. Giroire, Nisse, Trolliet, and Sulkowska (2021) and Giroire, Nisse, Ohulchanskyi, Sulkowska, and Trolliet (2022) extend the model of Avin et al. (2019), further Giroire et al. (2021) add the step of allowing adding multiple edges at a time, and Giroire et al. (2022) add the possibility that vertices become inactive.

2.4 Hypergraphs

Some networks contain additional features which cannot be captured by a general graph models, but could be captured by hypergraphs (Battiston et al., 2021; Benson et al., 2016; Ghoshal et al., 2009). The notation used for hypergraphs in the literature differs significantly, but the definition is generally the same. A hypergraph $H = (V, E)$ consists of a vertex set V and an edge set E consisting of edges with more than two vertices. The definition of the edge set E depends on if hyperedges can appear multiple times (parallel edges) and if a vertex can appear multiple times in a hyperedge (selfloops). As an example, Chodrow (2020) defines the edgeset as a multiset of multisets of V including parallel edges and selfloops. If a set contains two of the same vertices, it is called a degenerate hyperedge.



(A) Graph (B) Hypergraph

FIGURE 4: Graph and Hypergraph

On the other hand, Storm (2006) explicitly mentions that a vertex may not appear twice in a hyperedge, although "it is easy to generalize to this case". In business, two vertices also do not appear twice in a hyperedge, since this would mean that in collaboration networks a company could work with itself. Therefore, in a hypergraph parallel edges are allowed, but selfloops are not.

Visualizing hypergraphs is difficult in comparison to normal graphs. Graphs are visualised by dots (vertices) and lines (edges). Hyperedges in hypergraphs could not be visualized by lines. However, there are other options to visualize hypergraphs. First a hypergraph with at most three vertices in a hyperedge, could be visualised as a simplicial complex (Battiston et al., 2021). A special type of such a graph, a tripartite heterogeneous graph (hypergraphs with hyperedges of dimension 3), could also be visualized by projecting the nodes on one of its vertex types (Ghoshal et al., 2009). Simplicial complexes are presented by Battiston et al. (2020) as a way to represent higher-order interactions. Moreover, they show that a bipartite graph could be used to represent a hypergraph, by introducing hyperedges as nodes connected to vertices. The connections could easily be drawn in a node-edge matrix, as done by Saracco, Petri, Lambiotte, and Squartini (2022), such a matrix is also called an incidence matrix. Furthermore, Saracco et al. (2022) present a cartoon where all vertices belonging to one hyperedge are encircled by a specific color. This way of representation relates most to a "Venn diagram representation", a subset standard (Mäkinen, 1990). Mäkinen (1990) also presents the "edge standard", a drawing form where hyperedges are represented by curves, which is somewhat related to bipartite graph drawing, as explained above. All of these representations have their advantages and disadvantages, therefore we will use a variety of representations, choosing the most suitable in each instance. In Figure 4b a possible representation of a hypergraph is given (Venn Diagram). From the Figure, the extra features of hypergraphs can be seen when compared to Figure 1. From Figure 4a, the relation between vertices v_1, v_2 and v_4 and an extra relation between vertices v_2 and v_4 cannot be seen. In the example, where the nodes represent authors, this is also a more involved graph, since a collaboration could also exist of more than two authors. Furthermore, it is possible that a subgroup of authors published their own article.

Just as in graphs, hypergraphs can also have weighted edges. Battiston et al. (2020) state that the adjacency matrix of hypergraphs can be represented by $A = BWB^T - D$, where B is the incidence matrix, W is a diagonal matrix with the weights, and D is the

diagonal matrix with the vertex degree. Furthermore, they add that it might be tricky to properly define and compute degrees. Weighted hypergraphs are not considered as a necessity in this thesis, therefore a specific approach to deal with this will not be chosen.

In Section 2.2, heterogeneous, hierarchical and multi-layer graphs were discussed, which also played a role in representing tripartite graphs in the example above (Ghoshal et al., 2009). Another example would be representing heterogeneous contagion models, where part of the nodes are indicated as infected and the other nodes are indicated as healthy (Landry & Restrepo, 2020). Often, hypergraphs are introduced to connect different types of nodes with one hyperedge (Battiston et al., 2020; Bhagat et al., 2011). However, it is striking that the number of types, in most cases, is often not extended to more than 3-4 types.

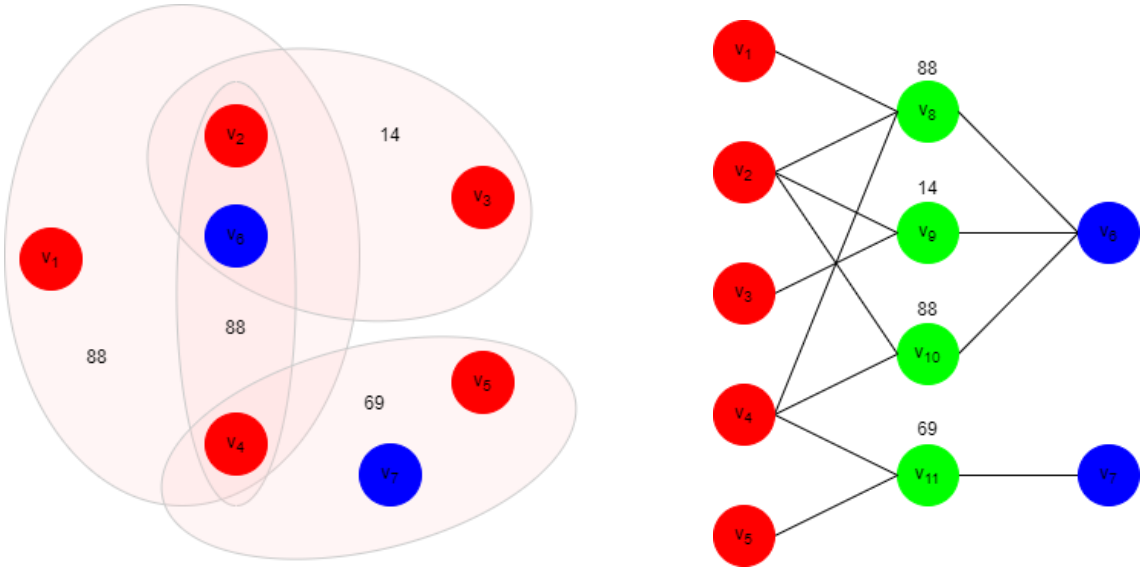
Finally, the random graph models as presented in Section 2.3 could be extended to hypergraphs. First, Chodrow (2020) presents the configuration model for hypergraphs. The preferential attachment model for graphs is extended to hypergraphs by Avin et al. (2019) and by Wang, Rong, Deng, and Zhang (2010). Giroire et al. (2021) and Giroire et al. (2022) extend the PA-model for hypergraphs of Avin et al. (2019) by the option of adding extra hyperedges and include the option of vertex deactivation. In this thesis, the models as presented in Chodrow (2020) are generalized and the model presented by Chung and Lu (2002) is extended to hypergraphs.

So one could combine the different concepts introduced here, in order to create a graph that captures all the features one is interested in. As an example, in Figure 5a a weighted heterogeneous hypergraph is illustrated. Applying this graph to the collaborating authors example, the red nodes represent the authors and the blue nodes represent the publishers of the papers. It can be seen that there are four papers written in total. Furthermore, publisher v_6 publishes three papers and publisher v_7 publishes one paper. The weights represent the number of citations. We see that two papers have the same number of citations (88). In order to show that the Venn Diagram is not the only visualization of a hypergraph. In Figure 5b, another visualization is shown. In this visualization the hyperedge is projected to a vertex. Each edge weight is transformed into a node weight.

2.5 Graph Analysis

In order to compare different graphs, different measures are used. The measures capture features of the vertices of the hypergraph, the hyperedges or the complete hypergraph. The measures we are going to use are centrality, modularity and homophily, which is also referred to as assortativity mixing. Centrality indicates how central nodes are in a network, modularity indicates whether there are modules, or clusters in the networks, and homophily indicates the preferences between the same node types to connect to each other. Centrality measures appear in many different forms. The most well-known centrality measure is degree centrality (Newman, 2005; Zhang & Luo, 2017). When considering random graph (GRG) models and PA-models, the degree distribution is often considered and whether the model is scale-free, i.e., whether the degree distribution follows a power-law (Albert & Barabási, 2002). In general, centrality stands for the importance of a node in a network (Ghosh & Lerman, 2019). Besides centrality, modularity is used for analysing networks, which is also known as clustering, community detection or connectivity (Newman, 2006b; Kamiński et al., 2019; Giroire et al., 2021; Blondel et al., 2008; Albert & Barabási, 2002). Finally, the tendency to connect with vertices of the same type (in heterogeneous graphs) is known as homophily or assortative mixing (Newman, 2010; Bhagat et al., 2011). Centrality, modularity and assortativity will be discussed in the remaining part of this section.

In Section 2.1, the vertex degree was already defined for standard graphs. This vertex



(A) Venn Diagram

(B) Projection hyperedge to vertex.

FIGURE 5: Two representations of Weighted Heterogeneous Hypergraph

degree coincides with the degree centrality discussed in this section. For hypergraphs, it is defined as the number of hyperedges a vertex is in (Chodrow, 2020). Giroire et al. (2021, 2022) and Avin et al. (2019) also define this as the vertex degree, but Wang et al. (2010) define this as the node hyperdegree. In addition to the vertex degree, the number of vertices in a hyperedge is sometimes defined as a degree, which is called hyperedge degree by (Saracco et al., 2022). Moreover, (Chodrow, 2020) defines this term as the dimension.

Finally, a hyperedge degree is defined by Wang et al. (2010), as the number of hyperedges incident with the hyperedge. So in general there are three "degrees" that should be distinguished in hypergraphs. First, (vertex) degree is used for the number of hyperedges a vertex takes a part in,

$$d_v = \sum_{e \in E} \mathbb{1}(v \in e).$$

Second, the dimension is the number of vertices in one hyperedge

$$d_e = \sum_{v \in V} \mathbb{1}(v \in e),$$

and finally the hyperedge degree is the number of hyperedges incident with the hyperedge

$$D_e = \sum_{v \in e} (d_v - 1) = \sum_{v \in e} \left(\sum_{e \in E} \mathbb{1}(v \in e) - 1 \right).$$

In addition to the degree centrality, other centrality measures that are used frequently are betweenness centrality, closeness centrality, eigenvector centrality and Bonacich centrality (Newman, 2005). First, betweenness centrality is defined as the amount of flow passing through a particular vertex considering shortest paths or geodesics (Newman, 2005; Zhang & Luo, 2017; Ruhnau, 2000). Zhang and Luo (2017) compare it with a mediation role, Ruhnau (2000) expresses it as a transmitter, and Newman (2005) defines it as measure for control. Mathematically, for a node v_i we define it as the sum over all combination of

nodes j and k, dividing the geodesics from node j to node k containing node v_i by the total number of geodesics from node j to node k. Second, closeness centrality is defined as the length of the path from a node to all other nodes (Brandes, 2001; Ruhnau, 2000; Zhang & Luo, 2017). Mathematically, the closeness centrality measure of a node v_i is the reciprocal of the sum over the length of shortest paths from a specific node v_i to all other nodes. Third, eigenvector centrality builds on the idea that the centrality of a node depends on the centrality of adjacent nodes (Ruhnau, 2000; Bonacich, 2007). Mathematically, this relates to the eigenvector of the highest eigenvalue of the adjacency matrix. The eigenvalue centrality of a vertex is defined as a linear equation of eigenvalue centralities of connected vertices. Whether vertices are connected by an edge is defined in the adjacency matrix. When writing the eigenvalue centrality as a system of equations, one could rewrite the system in such a way that the eigenvector is representing the eigenvalue centralities of the nodes. Lastly, the Bonacich centrality, also called beta-centrality, is defined as the weighted sum of paths connecting vertices to each other (Bonacich, 2007; Ghosh & Lerman, 2019). Bonacich (2007) defines beta-centrality as

$$c(\beta) = \sum_{k=1}^{\infty} \beta^{k-1} A^k,$$

where k represents the length of a path and A is the adjacency matrix.

The centrality measures as described above are applicable to graphs with dyadic collaborations. Centrality measures for hypergraphs are a bit more involved. Comparing the four types of centrality as described above, the measures could be categorized by centrality measures based on shortest path (betweenness and closeness centrality) and centrality measures based on the adjacency matrix (eigenvector and Bonacich centrality). First, centrality based on shortest path is applied to hypergraphs in the same way as to graphs with dyadic links. Lee, Lee, Oh, and Kahng (2021) propose to use bipartite graph representation for this purpose, which could be used to set up trees that easily measures the shortest paths. Gao et al. (2015) also use trees called relationship trees, a set of trees that order all the possible paths to other vertices. The relationship trees of Gao et al. (2015) do not contain the edges, which is the case for the trees presented by Lee et al. (2021). Second, in order to deal with centrality measures, Benson (2019) presents eigenvector centrality for uniform hypergraphs with help of hypermatrices or tensors. For hypergraphs where all hyperedges have size $|k_i|$, an $n^{|k_i|}$ symmetric hypergraph adjacency tensor T can be defined as (Benson, 2019):

$$T_{v_1,\ldots,v_{|k_i|}} = \begin{cases} 1 & \text{if } (v_1, \ldots, v_{|k_i|}) \in E \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

On the other hand, Bonacich, Cody Holdren, and Johnston (2004) suggest a method to translate the adjacency matrix to an incidence matrix, which can be used to compute eigenvector centrality for two-mode data. Knowing x and y as the centrality scores for the row and column elements of an incidence matrix B, the following pair of equations is defined

$$B^t x = \lambda y, \qquad By = \lambda x. \tag{2}$$

In this case, x and y are eigenvectors of different matrices, but they are associated with the same eigenvalue λ (Bonacich et al., 2004):

$$B^t B y = \lambda^2 y, \qquad BB^t x = \lambda^2 x. \tag{3}$$

Tudisco and Higham (2021) also use the notion of an incidence matrix, but their method is much more involved and will not be discussed in detail. The beta-centrality presented by Bonacich (2007) for hypergraphs is not developed yet.

Besides centrality measures, modularity measures could be used to characterize networks, especially networks with heterogeneous vertices. In general, modularity is "the number of edges falling within groups minus the expected number in an equivalent network with edges placed at random" (Newman, 2006a). Thus, the extent to which networks could be divided into modules or clusters. Mathematically, "edges in groups" could be expressed differently, especially in the case of hyperedges. Kamiński et al. (2019) maintain a strict definition where all vertices in a hyperedge should be of the same type. Giroire et al. (2021) follows this definition. Newman (2006a) uses the adjacency matrix for the clique reduction to represent this. In the same trend, Kumar, Vaidyanathan, Ananthapadmanabhan, Parthasarathy, and Ravindran (2020) presents the modularity for hypergraphs as:

$$Q = \frac{1}{2m} \sum_{v_i \in V} \sum_{v_j \in V} [A_{ij} - P_{ij}] \delta(T_{v_i}, T_{v_j}), \tag{4}$$

where $A = H^T(D_e - I)^{-1}H$ is the clique reduced adjacency matrix, H is the incidence matrix of the hypergraph, D_e is the diagonal matrix of the edge dimensions, I is an identity matrix, T_v is the type of node v, and P_{ij} is expressed as

$$P_{ij} = \frac{d_{v_i} d_{v_j}}{\sum_{v \in V} d_v}. \tag{5}$$

This last definition for hypergraph modularity is most suitable for this thesis, since the hyperedges in R&D collaboration networks can contain several types. The strict definition of Kamiński et al. (2019) would be less insightful for this thesis.

Finally, the concept of homophily or assortative mixing is considered. Homophily, "a tendency of various types of individuals to associate with others who are similar to themselves" (Currarini, Jackson, & Pin, 2009), and assortative mixing, "the tendency for vertices in networks to be connected to other vertices that are like (or unlike) them in some way" (Newman, 2003), are similar concepts, in essence, but looking at the definitions homophily is applied in the real world, so more often used in practice. Assortative mixing is more often used for dealing with graphs. The principle of homophily or that similarity breeds connection is extensively explained by Mcpherson, Smith-lovin, and Cook (2001). Currarini et al. (2009) define the homphily index as $H_i = s_i/(s_i + d_i)$, where s_i is the average number of friendships a type i node has with its own type and d_i is the average number of friendships a type i node has with nodes of different types than i. Inbreeding homophily appears when $H_i > w_i$, where w_i is the relative fraction of type i in the population. Assortativity for hypergraphs is not well-developed and is often focused on a specific hypergraph model. Papanikolaou, Lambiotte, and Vaccario (2023) study uniform hypergraph models as initial configuration. They define a homophily measure based on a merge and split technique with the fraction of nodes of a certain type. This fraction is compared to a threshold parameter γ. In addition, Alaluusua, Avrachenkov, Kumar, and Leskelä (2023) considers the level of assortativity for uniform hypergraphs, in the case of multi-layer hypergraph stochastic block models (HSBMs). Another model is the Location-Based Social Network (LBSNs), where an attempt is made to preserve homophily by maximizing the co-occurence probability of all homogeneous edges (Wang, Yuan, Zhang, Peng, & Liu, 2023). Furthermore, Failla, Citraro, and Rossetti (2023) present an evolving hypergraph model for heterogeneous hypergraphs: Attributed Stream Hypergraphs (ASHs). Presented measures for these

ASHs are consistency, purity, entropy and homogeneity, where homogeneity is most similar to homophily.

2.6 Concrete Example: WhatsApp Communication

In order to clarify the idea of hypergraphs and related terms, in this section a concrete example is given. This example consists of illustrating WhatsApp communication as a hypergraph models. This section can be skipped if the previous sections were clear.

Nowadays almost everyone is familiar with the social media application WhatsApp. This application provides the possibility to send texts to other people via the internet. This way of communicating could be represented by a graph model. A graph consists of actors and relations between these actors. The actors in WhatsApp are the people that text and a relationship exist between actors when people exchange texts with each other. Actors are often illustrated by dots and relationships are illustrations by lines between dots. In graph theory an actor is also referred to as a vertex and a relationship is referred to as an edge. When a graph represents a real situation the graph is often called a network in which vertices are called nodes and edges are called links. Perhaps, a person texts to more than one person. The number of persons a person exchanges texts with is called their degree. Furthermore, a relationship between two texting people could be assigned a value based on the frequency or importance of the communication, then this relationship is weighted. In this last case, one speaks about a weighted graph. Besides relationships having weights, an actor could also have a weight or a specific characteristic. A texting person could be male or female, have a certain age, or live at a specific place. When the characteristics of texting people are taken into account, the graph is called heterogeneous.

A texting person could also have relationships on other social media platforms, for example on LinkedIn, Facebook or Instagram. The relationships within these other social media platforms form a graph itself, but these graphs together form a multi-layer graph. In this type of graph the actors are the same, texting people, but the relationship differ per layer, via WhatsApp (layer 1), LinkedIn (layer 2), Facebook (layer 3) or Instagram (layer 4). Furthermore, an actor could be a texting family instead of a texting person. The texting family consists of texting people that also text each other. A graph with texting families could be considered, but the extra layer underneath these texting families, with texting people, could also be considered at the same time. When doing this the graph is called hierarchical.

It might be interesting to know how the WhatsApp graph or network arose or will evolve. The company WhatsApp might use this information for example to predict future revenues or to customize its app and improve profits. For this purpose random graph models are used. Some random graph models are based on how graphs are configured and others are based on the evolution of graphs.

One specific feature of WhatsApp is not discussed yet: WhatsApp groups. Within a WhatsApp group a texting person does not send a text to one person, but to a group of persons. Therefore, there is a relationship between a group of texting people and not between two texting people. This relationship could not easily be illustrated by a line. Therefore, such features cannot adequately be captured by a normal graph. Such a relationship is called a hyperedge and a graph with hyperedges is called a hypergraph. Thus, the WhatsApp network is a concrete example of a hypergraph. In Figure 6, a possible WhatsApp network is presented.

To make predictions or compare interventions, you need to quantify properties of the network. We call such properties measures. Where a company might use key performance indicators (KPIs), a graph considers centrality measures, measures that consider the im-

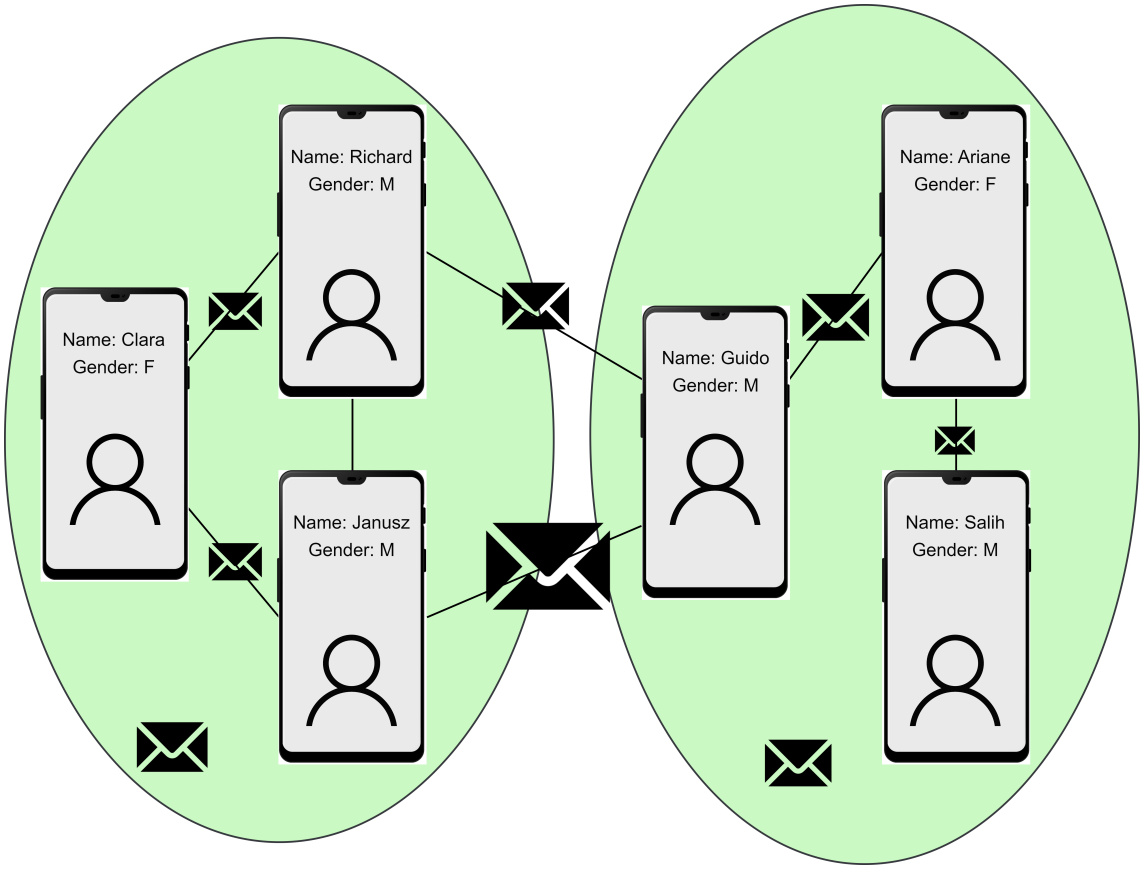


FIGURE 6: Small example of communication between pairs (lines) and groups (light green circles) of texting people.

portance of an actor in a network. The number of people a person texts with, called degree, is already discussed and is one of the centrality measures. A group counts only once when counting up the number of relationships, since there is a relation with a group and not with individual persons in the group. It might also be that persons text to themselves in a memo, this is called a selfloop and counts twice within counting for the degree. Another centrality measure is the number of people that join each other in a WhatsApp group, called the dimension. Furthermore, one may look at the overlap between a WhatsApp group and other WhatsApp group, called the hyperedge degree. Next to degree, other measures that will be considered are dimension, hyperedge degree, closeness, betweenness and eigenvector centrality. Considering centrality in the WhatsApp network, the texting persons with the highest centralities are those people who get to know the most information. In addition, modularity and homophily measures are often used in analysing a network. These measures are based on the idea that actors differ in features, so texting people have different characteristics (heterogeneous graph). Modularity indicates the extent to which similar actors are put together or clustered. Homophily indicates whether there is a preference for texting people to text with people with the same characteristics.

In the next section, the data used in this thesis is described, the network is analysed, and the models are introduced and extended. Furthermore, the parameters are tuned. For readers wishing to avoid mathematical technicalities, it is recommended to only read Section 3.1 and Section 4.1.

3 Method

3.1 Data

For this project data is used considering R&D collaboration projects in nanotechnology. The data is provided by the Dutch Technology Foundation for Engineering Sciences (Stichting voor Technische Wetenschappen (STW)), now NWO domain Applied and Engineering Sciences (AES). STW will be used to refer to the data. This data is captured in two data files. The first file includes features of the projects (project file) and the second file includes features of the project partners (project partner file). A project refers to a collaboration within the university-industry R&D collaboration network. A partner is an actor that takes part in such a collaboration or project. The projects that are involved start between 01-06-2000 and 01-09-2004 and are evaluated after 5 and 10 years.

In total the data consists of 419 projects with 797 project partners. To make the data useful, we do data cleaning, where certain projects and partners are removed from the data. Ten projects only have 1 project partner. This is not seen as a collaboration, and therefore these projects are removed. Furthermore, a main property of a network is that all nodes are connected. Three projects were not connected to the network, and therefore removed from the data set. Finally, two project partners were unknown and these are also removed from the data. We are left with 406 projects and 783 project partners.

The features of the projects captured by the data are project size, technology field, application field, received budget and external financing, level of commitment of users, degree of development, and amount of revenues generated. The main features of the projects are summarized in Table 1. The project size stands for the number of partners in a project. The technology field is referred to as the field in which the R&D collaboration project takes place (see Table 1 for the categories). The application field is focused on nanotechnology fields of application (the exact application fields are stated in Table 1). The number of projects within a certain technology field or application field are also shown

in Table 1. STW provides financing for R&D nanotechnology projects, which is referred to as the received budget. Sometimes an R&D project is also financed by an external party. The projects are evaluated based on three measures: commitment within the project of project partners, product development of the product in a project and revenues generated by the project. More on the four-point scale (0, A, B, C) related to these performance measures is explained later in this section. The code that is used to do the descriptive analysis is attached in Appendix C.1.

Feature	**Categories**	**(Range of) Values**
Number of Projects		406
Project sizes		2-22
Technology field	Electrical Engineering	123
	Life Sciences	49
	Medical Technology	68
	Chemistry	95
	Mechanical Engineering	44
	Civil Engineering	27
Application Field	Bionanotechnology	57
	Nanoelectronics	60
	Nanomaterials	45
	Nanomanufacturing and Tools	35
	Undefined	209
Financing	STW	€ 36,075-2,274,557
	External	€ 0-900,000
Commitment	A	83
	B	231
	C	92
Product development	A	92
	B	191
	C	123
Revenues generated	A	289
	B	82
	C	30
	n.a.	5

TABLE 1: Overview of the project variables

The main features of the project partners are: total number of projects participated in, number of partners within a project, project role, value chain position, number of patents in technology classes, operating income, and number of employees. The features of the project partners are summarized in Table 2. Based on Table 2, it should be noted that participated projects refers to the number of projects a unique partner participated in. Furthermore, the project role refers to the role a partner takes within a project. These roles are the producer, user and R&D role indicating whether it produces a product, uses a product or researches a product. The numbers behind the project roles in Table 2 sum up to a number larger than the number of unique partners, since in some cases a partner has different roles in different projects. The value chain position is relating to the overarching term for a group offering and representing specific services and interests in society. We notice that the value chain position is not distributed well over the different categories.

Especially the value chain position "companies" is over presented. The operating income refers to the gross income minus the operating expenses. The operating income is divided in 7 categories ranging from very large to very small. A company falls within the category of very small, when it has an operating income of less than 2 million euro. A company falls within the category of very large, when it has an operating income of more than 250 million euro. The medium category contains companies with operating of around 25 million euro. The rest of the categories are in between these values. It it remarkable that the operating income data contains a medium category, while it consists of only one partner.

Feature	**Categories**	**(Range of) Values**
Number of Partners		783
Participated Projects		1-103
Project Role	Producer	483
	User	919
	R&D	833
	Undefined	19
Value Chain Position	Companies	606
	Governmental parties	42
	Research Institutes	52
	(Academic) Hospitals/Medical Institutions	17
	Universities/Schools	47
	Special interest groups	19
Patents in Technology Classes	Human Necessities	0 - 74,698, n.a.
	Performing Operations / Transporting	0 - 9,830, n.a.
	Chemistry; Metallurgy	0 - 32,859, n.a.
	Textiles/ Paper	0 - 3,411, n.a.
	Fixed Constructions	0 - 331, n.a.
	Mechanical engineering / Lighting/ Heating/ Weapons/ Blasting	0 - 16,477, n.a.
	Physics	0 - 13,513, n.a.
	Electricity	0 - 41,534, n.a.
Operating Income	Very Small	181
	Small	69
	Medium Small	47
	Medium	1
	Medium Large	20
	Large	78
	Very Large	212
	Undefined	175
Employees	Average over 10 years	0 - 353,120

TABLE 2: Overview of the partner variables

To understand the data better, the project sizes and the performance of the projects are plotted and explained. In Figure 7 a histogram of the project sizes is shown. The project

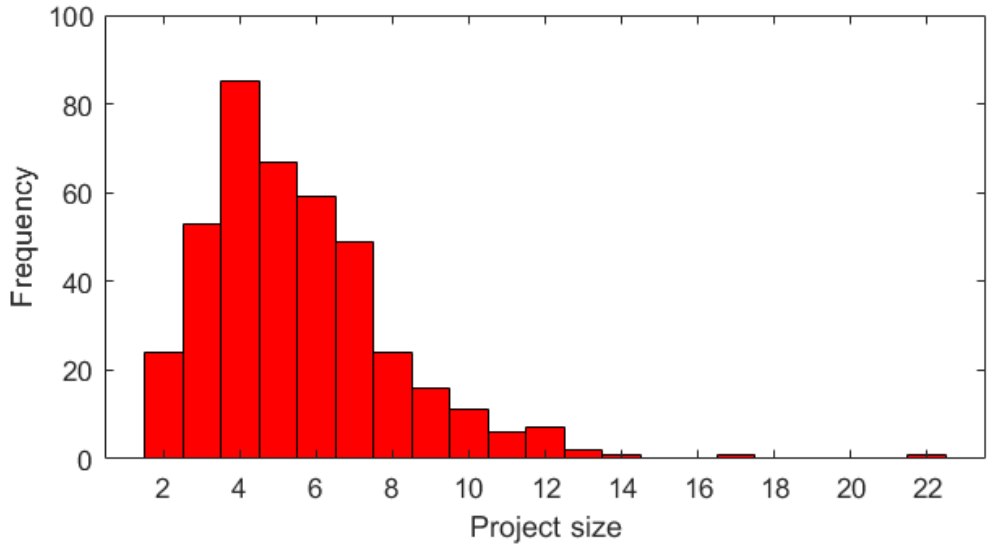


FIGURE 7: Frequency of Project Sizes

size varies between 2 and 22. A project size of 4 occurs most frequently and projects of size 14, 17 and 22 appear only once.

In Figure 8 the performance of the projects is shown. Performance is based on three concepts: involvement, product development and revenues. Involvement is related to the extent to which one (or more) users are involved in the project, product development relates to what extent the objectives of the project have been achieved, and the availability of a demonstrable "product", and revenues relates to the amount of revenues the project has lead to. These concepts are also used by Raesfeld, Geurts, Jansen, Boshuizen, and Luttge (2012), who elaborated on the same data. Each concept is subdivided into a four-point scale (0,A,B,C) as described in STW (2014), where 0 means failure and C indicates best performance. The abbreviation n.a. indicates a score of 0, since in some cases the valuation of the performance is not available.

3.2 Network Analysis

In the previous section we did explanatory analysis on the projects, its partners and their features. The R&D collaboration network, also referred to as the network, could be analysed based on graph theory and the concept of hypergraphs. In Section 2.5 different measures are explained. The analysis starts by plotting the network in Figure 9. The red dots represent project partners and the blue dots represent projects. In addition, the vertex degree distribution, dimension distribution and hyperedge degree distribution are given in Figure 10, Figure 7 and Figure 11, respectively. Furthermore, the network is analysed based on three centrality measures: betweenness centrality, closeness centrality and eigenvector centrality (see Section 2.5). Moreover, the modularity by Kumar et al. (2020) will be computed and the homophily index by Currarini et al. (2009) will be extended to hypergraphs to compute homophily in collaboration network, which will be presented as a hypergraph.

Three degree measures are considered, namely, the vertex degree distribution, dimension and hyperedge degree distribution. In Figures 10 and 11 the vertex degree distribution and the hyperedge degree distribution are shown. The dimension was already given in Figure 7 as project size equals the dimension. Figure 10 shows the vertex degree distribution

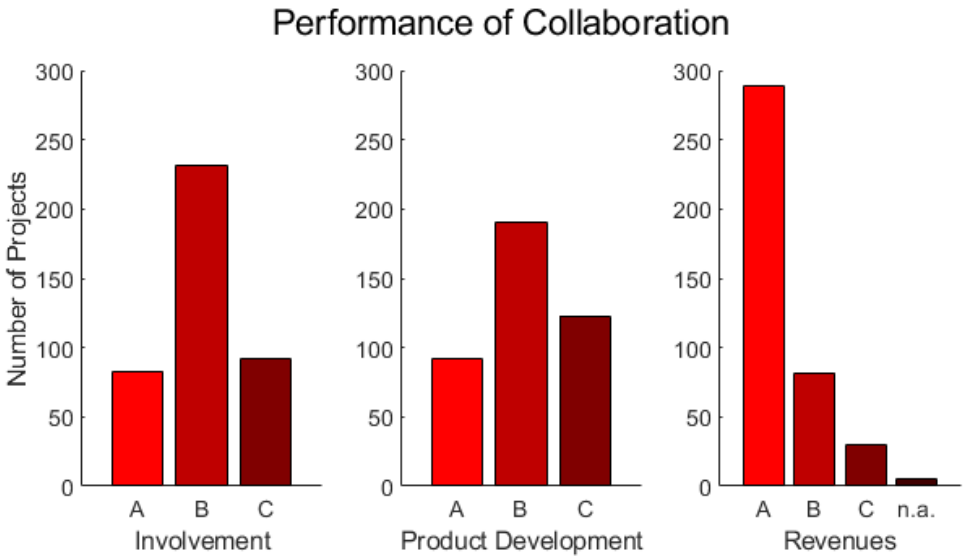


FIGURE 8: Performance of projects based on: (1) involvement, (2) product development and (3) revenues.

in linear and logarithmic scale. We see that the degree distribution follows a power law, meaning that a degree distribution follows the exponential equation $f(x) = ax^{-\tau}$, where a is a constant and τ is the exponent of the power law. The exponent τ lies around two, with $a \approx 500$. In Figure 7 the project sizes are shown and therefore the dimensions of the collaborative network are shown. It seems that the dimension follows a Poisson distribution, but it could also be a negative binomial distribution, for example. In Figure 11 the hyperedge degree distribution of the collaboration network is shown. It seems that the hyperedge degree distribution does not follow a specific distribution.

Besides the degree measures, we used three centrality measures to analyse the collaboration network: closeness centrality (CC), betweenness centrality (BC) and eigenvector centrality (EC). The CC and BC are based on the formulation of Brandes (2001). The EC is based on the product of the incidence matrices as shown by Bonacich et al. (2004). For computing these centralities, we use the data as explained in Section 3.1. The centralities are computed for each vertex. Since the field of hypergraph analysis is an emerging and developing field, the implementation of measures is often left to own interpretation. Although the main ideas are based on existing ideas, own creativity is often required to apply them to the hypergraph setting. The code used to compute these centralities is shown in Appendix C.2. In Table 3, the ranks are indicated for the degree, CC, BC and EC, where rank 1 means it has the highest degree, CC, BC or EC and where for example rank 4 means that it has the fourth highest degree, CC, BC or EC. The ranks for degree and centrality are almost the same, except for the CC, where Philips (node 684) has the fourth highest CC instead of the highest CC. So, the centrality measures are not always directly related to the degree. Shell (node 617) and Technical University Eindhoven (node 671) have the same degree, but still it seems that the CC is higher for Shell and the BC and EC are higher for the Technical University Eindhoven.

In Figure 12, we have plotted the centralities in decreasing order. For closeness centrality, it seems that it is normally distributed. For betweenness centrality and eigenvector centrality, a power law was expected, but the plots with logarithmic axes contradict these expectations.

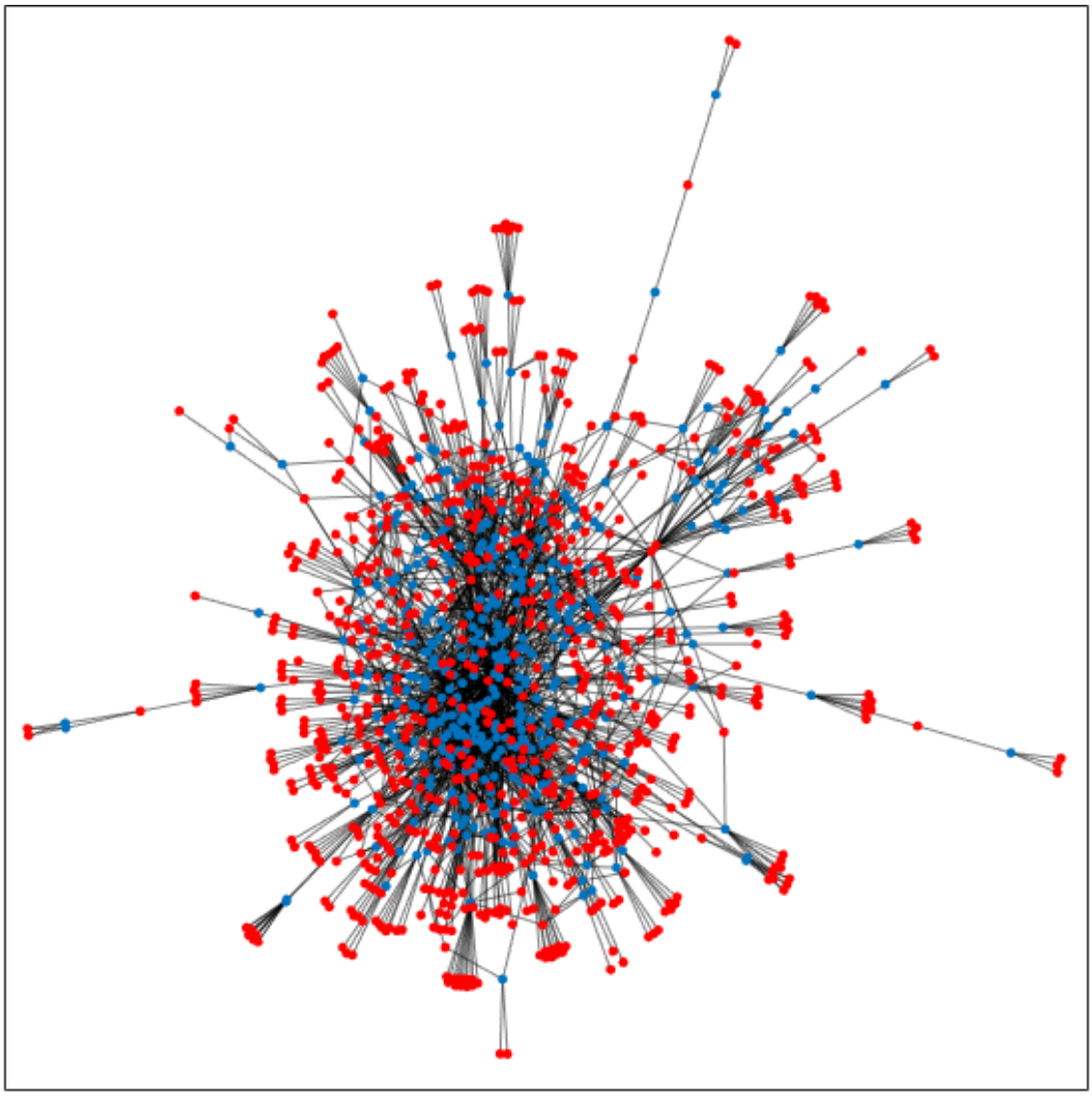


FIGURE 9: Collaboration Network

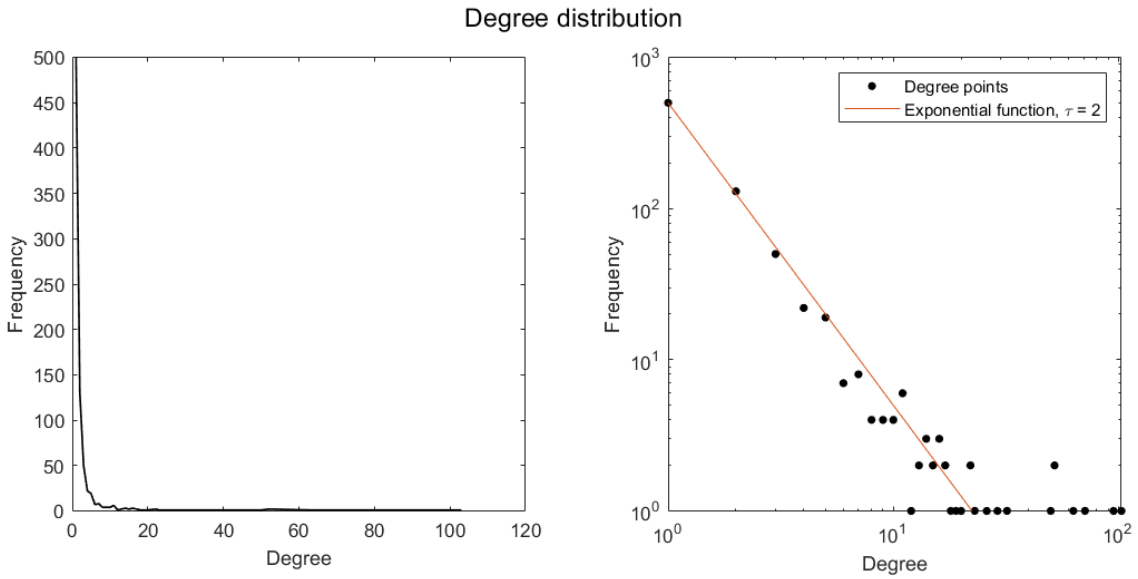


FIGURE 10: Vertex degree distribution of Collaboration Network in linear and logarithmic scale

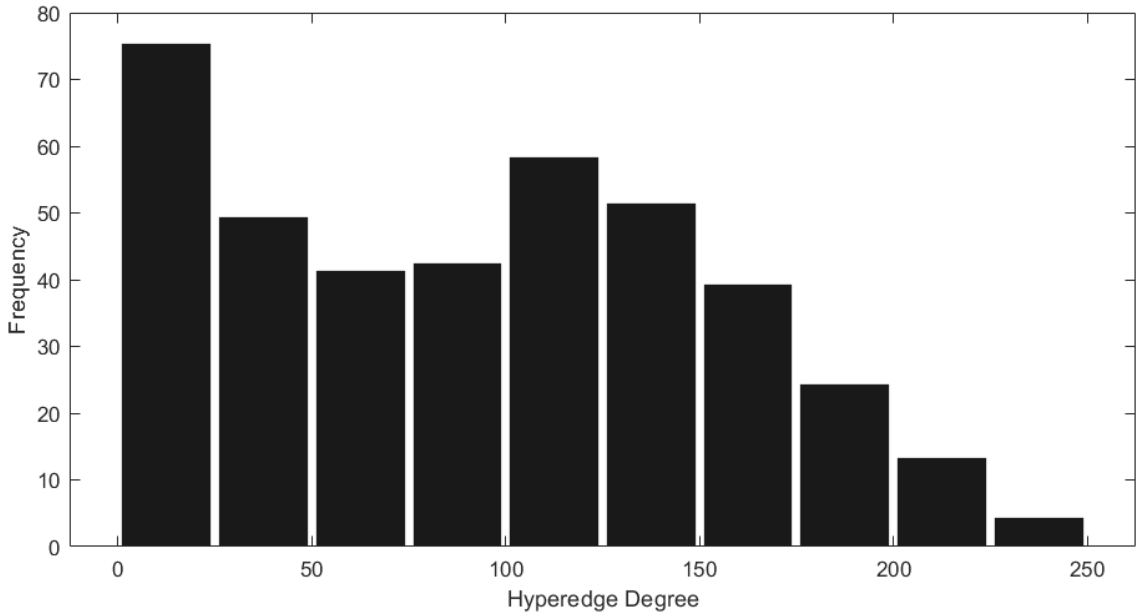


FIGURE 11: Hyperedge degree distribution of Collaboration Network

Besides the centrality measures, modularity is also computed. The modularity of the collaboration network is based on the modularity explained by Kumar et al. (2020). The modularity equation is shown in Equation 4. We take $A = H^T(D_e - I)^{-1}H$ to be the clique reduced adjacency matrix. The matrix H is the incidence matrix of the collaboration network, and D_e is a diagonal matrix with the dimensions of the edges on its diagonal. The modularity is based on predefined types. The types defined in this thesis are based on the Project Role and Value Chain Position, see Table 2. These node types are considered separately.

A project partner could have different project roles in different projects, therefore we

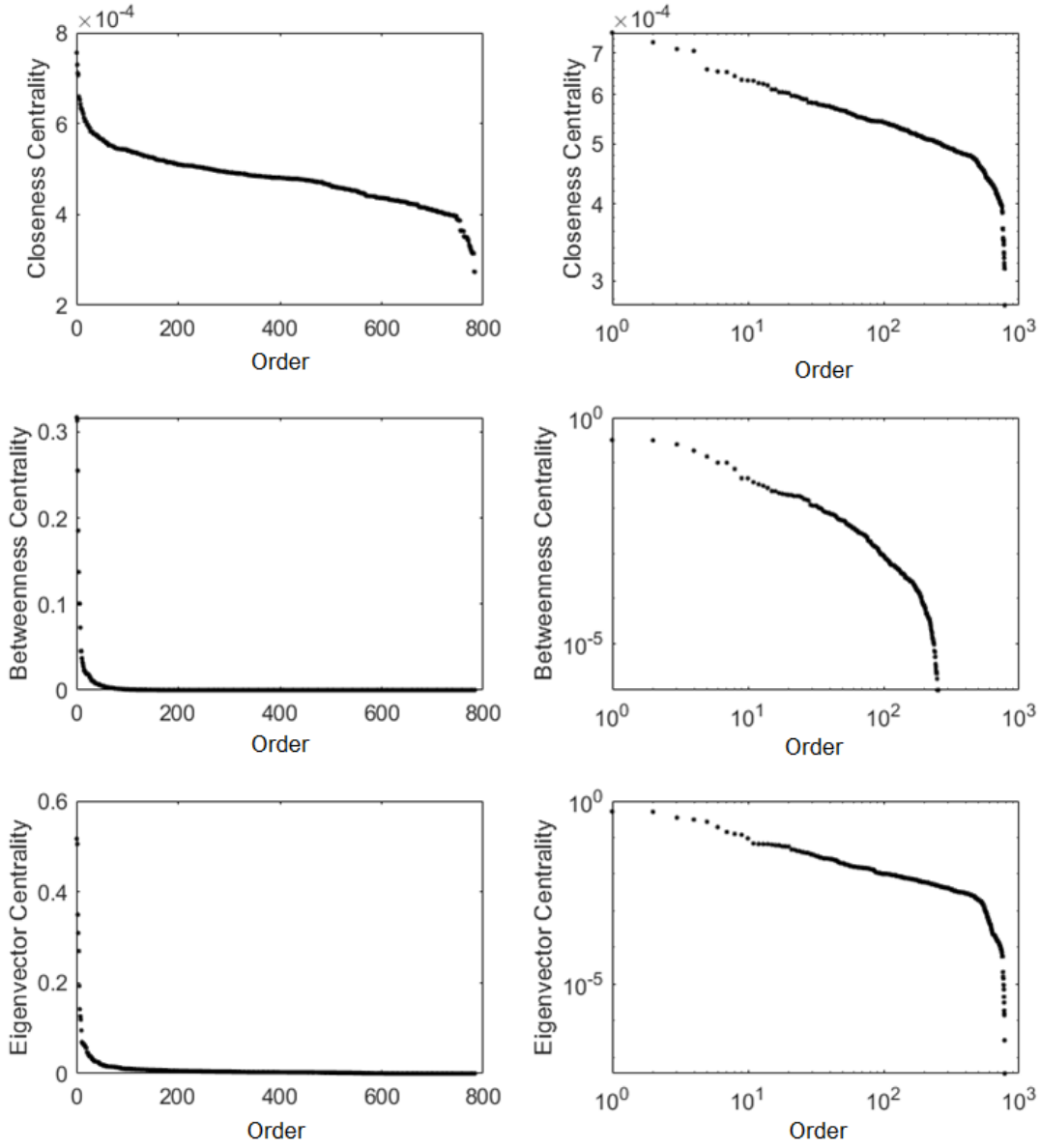


FIGURE 12: Sorted centralities linear and logarithmic scale

Node	Company Name	Highest Degree	Highest CC	Highest BC	Highest EC
684	Philips	1	4	1	1
670	TNO	2	1	2	2
721	Technical University Delft	3	2	3	3
528	University of Maastricht	4	3	4	4
617	Shell	5	5	6	6
671	Technical University Eindhoven	5	6	5	5

TABLE 3: Ranks of degree, CC, BC, and EC of the 5 companies with the highest degree.

choose to define the project role for each partner by the most occurring project role. When a project role appears an equal amount of times, project role 1 (Producer) has the preference over project role 2 (User), which has a preference over project role 3 (R&D). This is chosen for computational convenience. On the other hand, one could also state that the least appearing project role has the highest relative fraction in the whole network. In this case the preference order will be role 1, role 3, role 2. Still project role 1 has the most preference. In order to solve this problem, each node with more than one project role could have been divided into 2 or more nodes, where each node represents a department of the company and each node has a different project role (Producer, User, R&D). However, for computational reasons this is not preferred. In total 68 out of 783 partners have double project roles of which 25 partners have project roles that appear equally often. This happened 19 times between role 1 and role 2, 2 times between role 1 and role 3, and 4 times between role 2 and role 3. Since the number of equal project roles between role 2 and role 3 is negligible, in combination with the relative fraction of the project roles, the current preference order is retained. In addition to the three defined project roles there are also 19 project roles that are undefined. We deal with this group by stating that this is an extra project role. The project role is not defined, since the information was not available on these partners while coding the data.

The results show that modularity based on the project role distribution has a value of 0.1013. The modularity based on value chain position distribution has a value of 0.0905. The modularity function explained by Kumar et al. (2020) ranges between -1 and 1, where 1 indicates complete modularity and -1 indicates complete integration. Since the values are both around 0.1, there is a small tendency for modularity. In other words, there is a small preference for nodes of the same types to connect and therefore to cluster.

Since the modularity measure indicates that there is a small preference to cluster, probably there is also a preference to collaborate with own type partners. Therefore, it is expected that the homophily indices indicate this for some node types. In order to compute the homophily indices, as a guideline the measure by Currarini et al. (2009) is used and extended in order to be suitable for hypergraphs. The relative fraction of a type i in the population is represented by w_i,

$$w_i = \frac{|V_i|}{|V|}, \tag{6}$$

where V is the set of nodes and V_i is the set of nodes of type i. The homophily index H_i

for type i, is computed by

$$H_i = \frac{a_i}{a_i + b_i}, \tag{7}$$

where a_i is the average number of partners of a node of type i with a same type i node and b_i is the average number of partners of a node of type i with another type node $j, j \neq i$. This measure is extended to the hypergraph setting by stating that $a_i = a_{ii}$, and $b_i = \sum_{j \neq i} a_{ij}$. The value a_{ij} is defined as

$$\hat{a}_{ij} = \frac{1}{|V_i|} \sum_{v \in V_i} \sum_{u \in V_j} \sum_{e \in E} \mathbb{1}(v, u \in e). \tag{8}$$

However, Equation 8 assumes that the relationship between two project partners is the same for projects with different dimensions. One may argue that the relationship between project partners in a project with only three partners is higher than the relationship between project partners in a project with 10 project partners. Therefore, it is plausible to assume that the relationship between two partners depends on the size of the hyperedge, i.e., the dimension $|e|$. We propose to extend Equation 8 in the following way:

$$\check{a}_{ij} = \frac{1}{|V_i|} \sum_{v \in V_i} \sum_{u \in V_j} \sum_{e \in E} \frac{2}{|e|(|e|-1)} \mathbb{1}(v, u \in e). \tag{9}$$

As could be seen above, we use $2/(|e|(|e|-1))$ instead of $1/|e|$. This form ensures that connections in a dyadic relationship weigh the same as connections in a multiadic relationship. Suppose two persons work together, then person 1 has a relationship with person 2 and person 2 has a relationship with person 1, so in total there are 2 relationships. When three persons work together in a group then there is a relation between person 1 and person 2, between person 2 and person 3, and between person 1 and person 3, in total 3 dyadic relationships, so 6 relationships in total. Nevertheless, they work together in a group, so there is actually one multiadic relationship, so the sum of relationships should equal 2, the same number of relationships as in a dyadic relationship. Since the total of relationships should be equal to two, and there are three project members, each member has a total of 2/3 relationships (2/|e|) and these relationship should be divided between the remaining two members, so the relationship with each member is 1/3 (2/(|e|(|e|-1))).

For computational reasons we are going to simplify Equation 9 with the help of the frequencies each type appears in a hyperedge. For this purpose we introduce the function:

$$freq_i(e) = |\{v | v \in e, T(v) = i\}|. \tag{10}$$

The function $freq_i(e)$ is also referred to as $freq_i$, when it is clear which edge is considered. As stated, this newly defined function is used to simplify Equation 9. Instead of summing over the different types of vertices, this newly defined function could be used as:

$$\check{a}_{ij} = \begin{cases} \frac{1}{|V_i|} \sum_{e \in E} \frac{2}{|e|(|e|-1)} freq_i freq_j & i \neq j \\ \frac{1}{|V_i|} \sum_{e \in E} \frac{2}{|e|(|e|-1)} (freq_i - 1) freq_j & i = j \end{cases} \tag{11}$$

Equation 11 is much quicker in computing the values for $\check{a}$ than Equation 9, since Equation 9 has a triple sum that needs to be computed for all possible combinations of node types and Equation 11 only uses one sum and a function that most programming codes have a build-in function for. Finally, this $\check{a}$ is used to compute the homophily index H_i.

Finally, H_i is compared to w_i in order to establish homophily. In total, there are three cases:

$H_i > w_i$ preference for own type

$H_i = w_i$ no preference

$H_i < w_i$ preference for other types

Computations for both the project role type distribution and the value chain type distribution result in two $\check{a}$-matrices. The matrices are given in Table 4 and Table 5. PR1, PR2 and PR3 in Table 4 stand for project role 1, project role 2 and project role 3, respectively, which relate to the project roles in order of appearance in Table 2. The VC1, VC2, etc. stand for value chain type 1, value chain type 2, etc., which relate to the value chain position in order of appearance in Table 2.

	PR1	**PR2**	**PR3**	**Undef.**
PR1	0.1645	0.2331	0.2382	0.0046
PR2	0.1847	0.4524	0.3525	0.0061
PR3	0.4192	0.7827	0.6932	0.0090
Undef.	0.0782	0.1298	0.0867	0

TABLE 4: Matrix $\check{a}$ with preferences between project roles (PR) types

	VC1	**VC2**	**VC3**	**VC4**	**VC5**	**VC6**
VC1	0.5418	0.0272	0.0927	0.0413	0.1348	0.0076
VC2	0.3926	0.0935	0.1233	0.0189	0.0965	0.0058
VC3	1.0797	0.0996	0.2836	0.0227	0.3291	0.0170
VC4	1.4723	0.0467	0.0694	0.4322	0.3115	0.0118
VC5	1.7384	0.0862	0.3641	0.1127	0.3700	0.0210
VC6	0.2419	0.0129	0.0465	0.0105	0.0520	0.0244

TABLE 5: Matrix $\check{a}$ with preferences between value chain (VC) types

Based on the $\check{a}$-matrices, the homophily indices can be computed. These are compared with the relative fraction. The homophily indices and the relative fractions are shown in Table 6. The results show that there is a preference of project role 1, the producer, and the undefined group to collaborate with other project roles. For project role 2, the user, the values are almost the same for the homophily index and the relative fraction, there is a slight preference to work with other project roles. Project role 3, R&D, has a high preference to collaborate with other project role types. This seems reasonable. The data we are analysing consists of a University-Industry collaboration network (UICN) for nanotechnology R&D projects. So projects are put in the data that have at least one university or research institute and one company. Since the percentage of universities/research institutes is relatively low (99 out of 783 partners, see VC position), in comparison to the percentage of companies, there is not a high chance that universities or research institutes appear in the same project when there are in total 406 projects. Therefore, there is not a preference for universities or research institutes to work with each other.

3.3 Basic Models

The underlying dynamics of how networks are formed might be relevant for understanding these networks. Furthermore, it might be relevant to know which input variables are cru-

T = PR	**PR1**	**PR2**	**PR3**	**PR4**		
H_i	0.2568	0.4544	0.3640	0		
w_i	0.3461	0.4368	0.1967	0.0204		
T = VC	**VC1**	**VC2**	**VC3**	**VC4**	**VC5**	**VC6**
H_i	0.6409	0.1280	0.1548	0.1844	0.1374	0.0629
w_i	0.7739	0.0536	0.0664	0.0217	0.0600	0.0243

TABLE 6: Homophily indices versus the relative fractions for project role (PR) types and value chain (VC) types

cial in replicating collaboration networks. Measures that are crucial to define a network are the vertex degree distribution and the dimension distribution. Therefore, we wish to have models that constrain these measures. However, whether models with hard constrains or models with soft constrains are more useful needs to be tested in the hypergraph setting. Therefore, we are going to compare two models. The first model is based on hard constrains, where the exact vertex degree distribution and the exact dimension distribution are replicated. The second model replicates the exact dimension distribution, but a expected vertex degree distribution. The first model is the stub-labeled hypergraph configuration model (SHCM) presented by Chodrow (2020). Chodrow (2020) also introduces a model that avoids the presence of selfloops in the resulting hypergraph, the Markov Chain Monte Carlo for hypergraph configuration model. We will refer to this model as the no-selfloop hypergraph configuration model (NHCM). As explained in Section 2.4, a collaboration network does not have selfloops. Therefore, the SHCM and the NHCM are both considered. The second model is the Chung-Lu hypergraph model (CLHM) by Kamiński et al. (2019). In Algorithm 1 and Algorithm 2, respectively, the SHCM and NHCM by Chodrow (2020) are given. The CLHM is given in Algorithm 3.

In short, the SHCM given by (Chodrow, Eikmeier, & Haddock, 2023) in Algorithm 1 is based on randomly assigning stubs to a hyperedge with predefined dimension. In order to randomly assign the stubs, a degree sequence $\mathbf{d} \in \mathbb{Z}_+^n$ and a dimension sequence $\mathbf{k} \in \mathbb{Z}_+^m$ need to be defined beforehand.The degree of a vertex v, d_v, defines the number of stubs or half-edges per vertex v. The degree sequence is a vector with the previously defined degree for all n vertices $v_i \in V$, and the dimension is the previously defined dimension for all m hyperedges $k_i \in E$. The algorithm builds a hypergraph S. At initialization, $S = \emptyset$ and the set Σ is the multiset containing the stubs of all vertices. A specific vertex v_i is indicated by the subscript i and a specific stub v_{i_j} is indicated by the subscript j of the subscript i. For simplicity, both subscripts are not always used. When context is clear, v_j represents the stub of vertex v. During the algorithm for each hyperedge j, k_j hyperedges are uniformly picked from the multiset Σ. The randomly picked stubs form a hyperedge R. The stubs in R are then removed from the multiset of stubs Σ and added as a hyperedge to the hypergraph S. Eventually S contains m hyperedges with dimensions corresponding to the dimension sequence k, and each vertex has a degree corresponding to the degree sequence d.

Algorithm 1 is a simple CM for configuring a hypergraph given the dimension and the degree sequence. Nevertheless, this algorithm often results in a hypergraph that contains selfloops, i.e. a hyperedge that contains a specific vertex v_i multiple times. This would mean that within a collaboration, a partner contributes twice in the same group, which does not make sense in the context of R&D networks. Therefore, the algorithm should be performed several times before creating a hypergraph without selfloops. In order to handle this problem, Chodrow (2020) present the Markov Chain Monte Carlo Algorithm

Algorithm 1 Model Chodrow (2020) I: Hypergraph stub-matching (SHCM)

Input: Configurable $\mathbf{d} \in \mathbb{Z}_+^n$ and $\mathbf{k} \in \mathbb{Z}_+^m$
Output: S
Initialization: $S = \emptyset, \Sigma = \biguplus_{v \in V}\{v_1, ..., v_{d_v}\}$

1: **for** j = 1, ..., m **do**
2: $R \leftarrow$ Uniform $\binom{\Sigma}{k_j}$,
3: $\Sigma \leftarrow \Sigma \backslash R$
4: $S \leftarrow S \cup \{R\}$
5: **end for**

for hypergraph configuration models, see Algorithm 2.

In general, Algorithm 2 reshuffles the hyperedges, which form a selfloop based on an acceptance rate. In order to get started, the Algorithm 2 needs several inputs. Again, the dimension and degree sequence are considered as inputs. Furthermore, we need an initial hypergraph H_0, complying with the dimension and degree sequence, a sample size s and sample interval h. At initialization $t = 0$ and the hypergraph H_0 needs to be set at some hypergraph H.

During the implementation of the algorithm, it loops over sh time steps. At every time step the edge set E_t is changed slightly. At every step two hyperedges, Δ and Γ, are picked randomly from the current edge set, E_t. The randomly picked hyperedges are reshuffled by a function b and saved in a new hypergraph H'. After creating this new hypergraph H', we check if this new hypergraph is accepted using the acceptance rate α in a probabilistic way by comparing α with a uniformly picked number between 0 and 1 (see line 4 in Algorithm 2). The acceptance rate α is defined as:

$$\alpha(H'|H) = \begin{cases} \frac{2^{|\Delta \cap \Gamma|}}{m_\Delta m_\Gamma}, & H \sim_{\Delta,\Gamma} H' \\ 0, & \text{else} \end{cases} \tag{12}$$

in which m_Δ and m_Γ represent the number of parallel hyperedges to hyperedges Δ and Γ, respectively. A parallel hyperedge is defined as a hyperedge that has the exact and no less or more vertices than another hyperedge. The notation $|\Delta \cap \Gamma|$ refers to the number of vertices in the intersection of the hyperedges Δ and Γ. If the reshuffled hyperedges are accepted, the hypergraph for time $t+1$ is set as the new graph H', otherwise H_{t+1} remains the previous hypergraph H_t.

A different model for generating hypergraphs based on a degree distribution and a dimension distribution is the Chung-Lu hypergraph model (CLHM) presented by Kamiński et al. (2019). This model is not focused on exactly reproducing the dimension and degree sequence, but it is based on reproducing the exact dimension sequence and the expected degree sequence. The CLHM is closer to a generalized random graph model than a configuration model. Therefore, the CLHM is based on a probability distribution. First, the family of multisets, F_{k_i}, of size k_i (uniform hypergraphs with edges with dimension k_i) is defined as:

$$F_{k_i} := \left\{ \{(v_i, M_i)\, 1 \leq i \leq n\} : \sum_{j=1}^{n} M_j = k_i \right\} \tag{13}$$

In this equation M_i refers to the multiplicity of a node v_i in a hyperedge. Based on the multiset in Equation 13, the original Chung-Lu model (Chung & Lu, 2002) is used by

Algorithm 2 Model Chodrow (2020) II: Markov Chain Monte Carlo for hypergraph configuration models (NHCM)

Input: Configurable $\mathbf{d} \in \mathbb{Z}_+^n$, $\mathbf{k} \in \mathbb{Z}_+^m$, $H_0 \in \mathcal{H}_{d,k}$, $h \in \mathbb{Z}_+$, $s \in \mathbb{Z}_+$.
Output: $\{H_t$ such that $t/h \in \mathbb{Z}_+\}$
Initialization: $t = 0$, $H_0 = H$

for t = 1, 2, ..., sh **do**
  sample Δ and Γ) uniformly at random from E_t
  $H' = b(\Delta, \Gamma | H_t)$
  if Uniform($[0, 1]$) $\leq \alpha(H'|H_t)$ **then**
    $H_{t+1} \leftarrow H'$
  else
    $H_{t+1} \leftarrow H_t$
  end if
end for

Kamiński et al. (2019) to define a multinomial distribution. The probability of generating a hyperedge $e \in F_{k_i}$ is given by:

$$P_{\mathcal{H}}(e) = \binom{k_i}{M_1, ..., M_n} \prod_{j=1}^{n} \left(\frac{d_{v_j}}{\sum_{v \in V} d_v} \right)^{M_j} \tag{14}$$

After defining the probability of generating a hyperedge, we choose how many edges with dimension k are added to the graph. After the number of edges of a certain dimension is defined for each dimension, the edges are randomly picked based on the probability as defined in Equation 14. So the number of edges should be preserved, but it is not guaranteed that the degree of each vertex is preserved, only the expected degree distribution is preserved. The CLHM is summarized in Algorithm 3.

Algorithm 3 Chung-Lu hypergraph model (CLHM)

Input: Configurable $\mathbf{d} \in \mathbb{Z}_+^n$, $\mathbf{k} \in \mathbb{Z}_+^m$,
Output: S
Initialization: $\mathbf{p} = \mathbf{d} / \sum_{i=1}^{n} \mathbf{d}_i$, $S = \emptyset$

for $i = 1 : |k|$ **do**
  Define F_{k_i}
  Randomly pick an edge e from $F_{k_i} \sim$ Multinomial($\mathbf{p}$)
  $S = S \cup e$
end for

3.4 Heterogeneous Hypergraph Models

The models as introduced in Section 3.3 should be changed in order to make them suitable for modelling collaboration networks with heterogeneous vertices. The heterogeneous hypergraph models are based on the idea of stochastic block-models (SBMs), where a block-matrix indicates the preferences of nodes to connect with nodes in the same community (or block) or with nodes in other communities (Casiraghi, 2019; Battiston et al., 2020; Chodrow et al., 2023). The block-matrix given by Ω, where Ω_{ij} denotes the probability that vertex i connects with vertex j. Casiraghi (2019) specifically defines the block-constrained configuration model, but this is not applied to hypergraphs. Chodrow et al. (2023) introduces a sparse hypergraph SBM (HSBM). Within this HSBM a probability distribution

η over hypergraphs is introduced, where events are all possible edges. Considering that a collaboration could exist of 22 partners and supposing that there are three communities they could belong to, there are in total

$$\binom{22+3-1}{22} = 276$$

different options. This would mean that 276 parameters need to be found to get the probability to form a group of 22 partners. Furthermore, this needs to be done for the other dimensions (1-21). In addition, when one more type is added, 2300 parameters need to be found for groups with dimension 22. Therefore, this approach is not suitable in our case. Battiston et al. (2020) give a short overview of SBMs for hypergraphs. Besides the method Chodrow et al. (2023) suggest, Battiston et al. (2020) points to the adapted Kronecker Graph Model by Eikmeier, Ramani, and Gleich (2018). Unfortunately, this approach does not give the certainty to retain exact dimension and degree sequences, so this would not be usable for the configuration model of Chodrow (2020).

Since the idea of SBM with block-matrix Ω, also known as the propensity matrix, is easy to understand, this idea is used to deal with heterogeneity in hypergraphs. Since the propensity matrix Ω consists of dyadic preferences, Ω_{ij}, it could not immediately be applied to a hypergraph where hyperedges should have multiadic preferences. Therefore, we develop an approach that builds up a hyperedge step by step. For this approach a linear function of dyadic preferences is introduced. The linear function gives the probability that a certain vertex type joins an existing collaboration or hyperedge R. The type of a vertex is stored in vector T, which means that T_{v_i}, indicates the type of vertex v_i. The probability that a vertex joins an existing hyperedge R is expressed as

$$P(R \to R \cup v_i) = \frac{\sum_{v \in R} \Omega_{T_v, T_{v_i}}}{|R|}. \tag{15}$$

This linear function is easy to use in practice. Based on this function, the SHCM, NHCM and the CLHM are changed into heterogeneous hypergraph models. The heterogeneous hypergraph models are given in Algorithm 4, Algorithm 5, and Algorithm 6.

In Algorithm 4 the stub-labeled heterogeneous hypergraph configuration model (SHHCM) is given. As input, it requires a degree sequence, a dimension sequence, a vector of node types, and the propensity matrix Ω with the probabilities with which each of the vertex types are connected. At initialization, the vertices are split into sets V_i, which contain the vertices of type i. The stubs are also ordered by type i in the multiset Σ_i, which are combined in the multiset Σ. The algorithm runs over the number of hyperedges, $j = 1, ..., m$. At the beginning of forming a hyperedge, a random stub is picked from the remaining stubs Σ. After this vertex is chosen, the other vertices $(k_j - 1)$ are added one by one based on Equation 15. If a stub of a certain type is not available anymore, the probability to pick this type within the hyperedge is zero. Therefore, Ω is adapted during the implementation of the algorithm. In Algorithm 4 this is illustrated in lines 4-6, and lines 14-16. The full length code is attached in Appendix D.1.

In Algorithm 5, the no-selfloop heterogeneous hypergraph configuration model (NHHCM) is shown. We also choose to show the complete process of reshuffling Δ and Γ as indicated by $b(\Delta, \Gamma)$ in Algorithm 2. The input for Algorithm 5 is expanded by including a vector T with vertex types and the propensity matrix Ω with preferences to connect between vertex types. In addition, we have an initial hypergraph H_0, a sample interval h and a sample size s. The initial hypergraph is obtained by implementing the SHHCM. By performing Algorithm 5, we intended to obtain a hypergraph in which no selfloops are present after reshuffling hyperedges.

Algorithm 4 Stub-labeled heterogeneous hypergraph configuration model (SHHCM)

Input: Configurable $\mathbf{d} \in \mathbb{Z}_+^n$, $\mathbf{k} \in \mathbb{Z}_+^m$, $\mathrm{T} \in \mathbb{Z}_+^n$, $\Omega \in \mathbb{R}_+^{\max(T)\times\max(T)}$
Output: S
Initialization: $S = \emptyset, \Sigma = \biguplus_{i=1}^{\max(T)} \Sigma_i, \Sigma_i = \biguplus_{v\in V_i}\{v_1, ..., v_{d_v}\}, V = \bigcup_{i=1}^{\max(T)} V_i$

for j = 1, ..., m **do**
  $R \leftarrow$ Uniform $\binom{\Sigma}{1}$,
  $\Sigma = \Sigma \backslash R$
  if $\Sigma_{T_R} = \emptyset$ **then**
    $\Omega_{:,T_R} = 0$
    normalize rows Ω
  end if
  for $i = 1 : k_j - 1$ **do**
    Compute $P(R \to R \cup \{t\}) = \frac{\sum_{j=1}^{|R|} \Omega_{T_{R_j},t}}{|R|} \qquad \forall t = 1, 2, ..., \max(T)$
    $P_s = \frac{1}{|\Sigma_{T_s}|} P(R \to R \cup \{T_s\}) \qquad \forall s = 1, ..., |\Sigma|$
    $K = \mathrm{random}(\Sigma, P_s)$
    $R = R \cup \{K\}$
    $\Sigma \leftarrow \Sigma \backslash \{K\}$
    if $\Sigma_{T_K} = \emptyset$ **then**
      $\Omega_{:,T_K} = 0$
      normalize rows Ω
    end if
  end for
  $S \leftarrow S \cup \{R\}$
end for

When implementing Algorithm 5, a loop over time is performed. In total s times h time steps are performed. In general, at each time step the algorithm picks two hyperedges, it applies reshuffling, and it checks whether the reshuffle is accepted. In order to show the reshuffling process some parameters are introduced. Again, the parameters Δ and Γ represent the randomly selected hyperedges. The parameters Δ' and Γ' are the reshuffled hyperedges, which are set empty at every time step. Furthermore, Δ_{new} and Γ_{new} are auxiliary parameters, that keep track of the stubs that should still be distributed over the reshuffled hyperedges, which at initialization are set as the randomly selected hyperedges Δ and Γ. In lines 8-15, all stubs that exist in both randomly selected hyperedges are added to both reshuffled edges. After assembling the remaining stubs, the reshuffled edge Δ' is completed by the required number of remaining stubs. Furthermore, the completion is based on types. Equation 15 is used for this purpose (see line 19). The complete process of completion is illustrated in lines 17-26. The stubs that still remain after the completion process of hyperedge Δ', are added to Γ'. The reshuffled hyperedges are added to S' and this hypergraph is checked for acceptance (see equation 12). This part of the algorithm is the same as the original algorithm. The full length code is attached in Appendix D.2.

The CLHM by Kamiński et al. (2019), was summarized in Algorithm 3. The Chung-Lu heterogeneous hypergraph model (CLHHM) is captured in Algorithm 6. The input consists of a dimension and degree sequence, d and k, a vector with vertex types T and a propensity matrix Ω with probabilities of connections between heterogeneous vertices. The algorithm starts with m empty hyperedges, meaning no stubs are attached to the hyperedges. Furthermore, the parameter p is defined as the vector with the parameters p_i for the multinomial distribution related to vertex v_i. It is computed by dividing the vertex degree by the sum of all vertex degrees. Based on the multinomial distribution with parameter p, which is also shown in equation 14, one stub is added to every hyperedge as described in lines 1-3 of the algorithm. The number of stubs that still need to be added to a hyperedge is referred to as undefined stubs. In total $\sum_{i=1}^{m} \mathbf{k}_i - m$ stubs are undefined. A stub is randomly sampled based on the multinomial distribution with parameter p. Based on the propensity matrix Ω, the probabilities, p_{het}, that the stub is attached to a certain hyperedge are computed. Again, the linear function that deals with Ω is used as given in Equation 15. The subscript $T_{S_{j_l}}$ in line 8 means that we intend to find the type T of the l-th stub in the j-th hyperedge of hypergraph S. Eventually, the probability vector p_{het} is used to randomly assign the stub to a hyperedge. The full length code is attached in Appendix D.3.

3.5 Parameter Tuning

For the previously mentioned algorithm certain input parameters are needed that need to be estimated from the data we want to model. These parameters include a degree vector, with the degrees for each node, a dimension vector, with the dimensions for each project, a type vector with the type for each node, a preference matrix, in which it is stated what the preference between nodes is, the sample size and the sample interval. The degree vector and the dimension could immediately be contained from the data, which is also shown in Figure 10 and Figure 7. During the network analysis in Section 3.2, two types are defined the project role and the value chain position. This definition is adopted as the type vector. The preference matrix Ω will be based on the $\check{a}$-matrix, since this matrix states a preference between types. The preference matrix is the $\check{a}$-matrix, in which the rows are normalised.

The sample size and the sample interval parameters still need to be tuned. These parameters are only used in Algorithm 5 and should be sufficiently large such that it is ensured that there are no selfloops in the generated network. In total there are 2254 stubs

Algorithm 5 No-selfloop heterogeneous hypergraph configuration model (NHHCM)

Input: Configurable $\mathbf{d} \in \mathbb{Z}_+^n$, $\mathbf{k} \in \mathbb{Z}_+^m$, $\mathrm{T} \in \mathbb{Z}_+^n$, $\Omega \in \mathbb{R}_+^{\max(T)\times\max(T)}$
$H_0 \in \mathcal{H}_{d,k}$, $h \in \mathbb{Z}_+$, $s \in \mathbb{Z}_+$.
Output: $\{H_t$ such that $t/h \in \mathbb{Z}_+\}$
Initialization: $H_0 = S$

for t = 1, 2, ..., sh **do**
  sample (Δ, Γ) uniformly at random from E_t
  $S' = S \backslash (\Delta, \Gamma)$
  $\Delta' = \emptyset$
  $\Gamma' = \emptyset$
  $\Delta_{new} = \Delta$
  $\Gamma_{new} = \Gamma$
  for $i = 1 : |\Delta|$ **do**
    if $\Delta_i \in \Gamma_{new}$ **then**
      $\Delta' = \Delta' \cup \Delta_i$
      $\Delta_{new,i} = \emptyset$
      $\Gamma' = \Gamma' \cup \Delta_i$
      $\Gamma_{new} = \Gamma_{new} \backslash \Delta_i$
    end if
  end for
  $Rem - Stubs = \Delta_{new} \cup \Gamma_{new}$
  for $i = 1 : |\Delta_{new}|$ **do**
    for $j = 1 : \max(T)$ **do**
      $Pt_j = P(\Delta' \to \Delta' \cup \{j\}) = \frac{\sum_{l=1}^{|\Delta'|} \Omega_{T_{\Delta'_l}, j}}{|\Delta'|}$
    end for
    Normalize Pt
    $P = Pt_{T_{Rem-Stubs}}$
    $K = \mathrm{random}(Rem - Stubs,\ P)$
    $\Delta' = \Delta' \cup \{K\}$
    $Rem - Stubs = Rem - Stubs \backslash \{K\}$
  end for
  $\Gamma' = \Gamma' \cup \{Rem - Stubs\}$
  $S' = S' \cup \{\Delta', \Gamma'\}$
  if $\mathrm{Uniform}([0,1]) \leq \alpha(S'|S)$ **then**
    $H_t \leftarrow S'$
  else
    $H_t \leftarrow H_{t-1}$
  end if
end for

Algorithm 6 Chung-Lu heterogeneous hypergraph model (CLHHM)

Input: Configurable $\mathbf{d} \in \mathbb{Z}_+^n$, $\mathbf{k} \in \mathbb{Z}_+^m$, $\mathrm{T} \in \mathbb{Z}_+^n$, $\Omega \in \mathbb{R}_+^{\max(T)\times\max(T)}$
Output: S
Initialization: $\mathbf{p} = \mathbf{d}/\sum_{i=1}^{n} \mathbf{d}_i$, $S = \emptyset$

for $i = 1 : m$ **do**
    Assign vertex to edge $i \sim$ Multinomial($\mathbf{p}$)
end for
$Undef\_Stubs = \mathbf{k} - 1$
for $i = 1 : \sum_{i=1}^{m} \mathbf{k}_i - m$ **do**
    Pick *stub* from the $Undef\_Stubs \sim$ Multinomial($\mathbf{p}$).
    for j = indices in $Undef\_Stubs > 0$ **do**
        $p_{het,j} = \sum_{l \in S_j} \Omega_{T_{S_{j_l}}, T_{stub}} / |S_j|$
    end for
    Normalize p_{het}
    R = random($Undef\_Stubs > 0$, p_{het})
    $S_R = S_R \cup stub$
    $Undef\_Stubs_R = Undef\_Stubs_R - 1$
end for

at most there are 1127 selfloops. For 100 runs of the NHHCM, it is tested how many time steps ($t = 1 : s * h$) are needed to avoid selfloops. In Figure 13 we plot a histogram of how many time steps were needed to remove the selfloops from the random graph. This number does not seem to be normally distributed, but it resembles a Poisson distribution. However, this is purely speculative. From the Figure we could state that a time span of 5000 probably is enough, and therefore we chose $s = 125$ and $h = 40$.

3.6 Comparison of the Models

In the previous sections, the SHCM, the NHCM, and the CLHM models are explained and changed into the heterogeneous hypergraph models SHHCM, NHHCM, and CLHHM, respectively. In short, the SHHCM uses a given degree for each node and the dimension of each collaboration to randomly create a network. However, selfloops are often created with this model. The NHHCM tries to correct this by reshuffling the partners in collaborations. The CLHHM is a model based on an expected degree for each node instead of an exact degree distribution. Therefore, the CLHHM could be used to test the importance of constraining the exact degree distribution.

In the next section, these models are run based on the parameters as estimated from the data. In total the models are run 50 times in order to be able to obtain a good ensemble average of the newly found models. After the models are run the models are compared to each other using a variety of measures. These measures include the vertex degree distribution, dimension distribution and hyperedge degree distribution, centrality, modularity and homophily. The closeness centrality and the betweenness centrality will not be used, since we cannot easily compute these centrality measures for nodes that are not part of the giant component of a network. Furthermore, the computation time of these centralities is large. Therefore, only the eigenvector centrality is computed. This should be sufficient as from Table 3 it could be concluded that the highest peaks for the eigenvector centrality were corresponding with the peaks of the closeness and betweenness centrality measures.

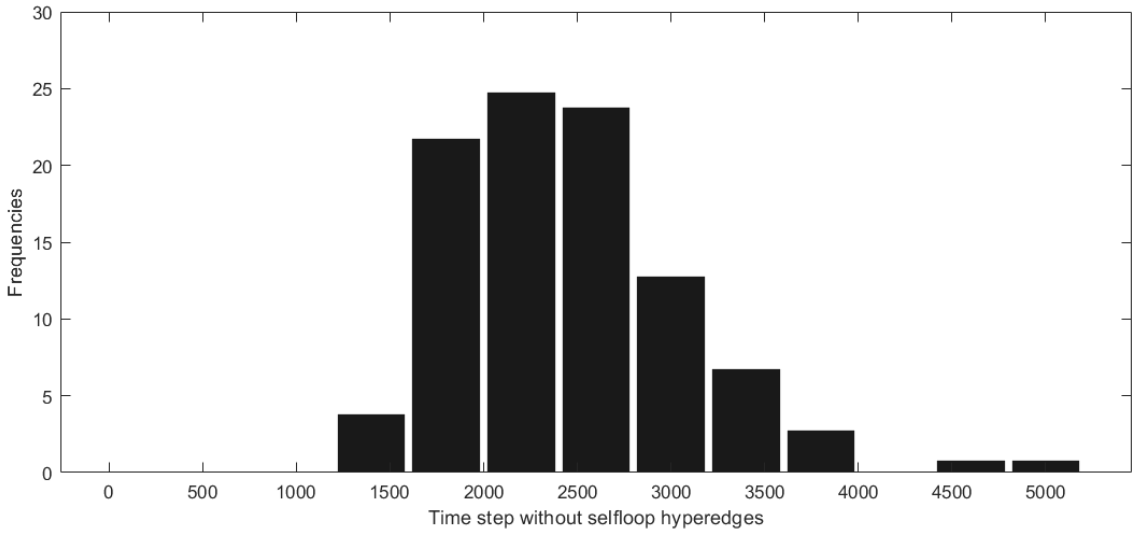


FIGURE 13: Number of time steps to avoid selfloops.

When comparing the hyperedge degree distributions the Kullbach-Leibler divergence (KL-div) is used and when comparing eigenvalue centrality (EC) the mean squared error (MSE) is used. The KL-div is comparing the hyperedge degree distribution of the original network D_N with the hyperedge degree distribution of the sampled networks of the hyperedge models D_{HM}. The KL-div is computed by summing over a sample space $\mathcal{X}$. The sample space relates to the bins as shown in Figure 11. The size of the bins of D_N and of D_{HM} of each sample $x \in \mathcal{X}$, is referred to as $D_N^{(x)}$ and $D_{HM}^{(x)}$, respectively. The Kullbach-Leibner divergence is computed as (Kullback & Leibler, 1951):

$$\text{KL-div}(D_N|D_{HM}) = \sum_{x=1}^{\#bins} D_{HM}^{(x)} \cdot \log\left(\frac{D_{HM}^{(x)}}{D_N^{(x)}}\right). \tag{16}$$

It is important to choose bins such that $D_N^{(x)}$ and $D_{HM}^{(x)}$ are nonzero. The mean squared error computes the average difference between the EC of all vertices of the original network (EC_N) and the EC of all vertices of the sampled networks (EC_{HM}). In total there are n vertices. The MSE is computed as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(EC_N^{(i)} - EC_{HM}^{(i)})^2. \tag{17}$$

The values $EC_N^{(i)}$ and $EC_{HM}^{(i)}$ refer to the EC of node v_i of the original network and the sampled networks, respectively.

In addition, the models will also be implemented with parameters that assume that the collaboration data could be represented by a normal graph. In this case each collaboration or hyperedge is replaced by a clique where each participant of a collaboration is connected to all other participants by a pairwise connection. Selfloops are excluded.

4 Results

4.1 Comparison of heterogeneous hypergraph models

In the previous section, we explained the heterogeneous hypergraph models, SHHCM, NHHCM and CLHHM. In this section each of these models is implemented 50 times with

the parameters found in Section 3.5, resulting in 50 sampled networks per model. After sampling from these models, the vertex degree distribution, dimension, hyperedge degree distribution, eigenvector centrality, modularity and the homophily indices are computed to establish 95% confidence intervals (CIs) for models. These measures are used to compare the models with each other and with the original data. We implemented the models for two different node types: project role and value chain position. The measures for both node types are compared with the measures for the original graph. The full length code for implementing these models given in Appendix E.1.

Project role. First, the implemented models with project role node types are compared. We expect that the NHHCM model is better than SHHCM, since NHHCM removes self-loops from the model. Furthermore, we expect that NHHCM is better than the CLHHM model for hypergraphs, since for NHHCM the degree sequence is more constrained. The degree sequences for SHHCM and NHHCM are equal to the degree sequence of the original network, as expected, since this was constrained.

The sampled network of the CLHHM model is not exactly reproducing the exact degree sequence, but it reproduces the expected degree sequence. Therefore, the degree sequence for every sampled network of the CLHHM model differs. A 95%-CI is computed for all 783 nodes. We expect that in about 95% of the cases the real degree of a vertex is within the interval. We see that in 736 out of the 783 cases the degree lies in the 95% confidence interval (CIs), so in approximately 94% of the cases. The CLHHM model thus preserves the expected degree sequence. The dimension sequences of the sampled networks is equal to the dimension sequence of the original network, as expected since this was a hard constraint of the CLHHM model.

Hyperedge degree. The hyperedge (HE) degree distribution is characterized by five measures: maximum HE degree (max), minimum HE degree (min), mean HE degree, standard deviation HE degree (std) and the Kullbach-Leibler divergence (KL-div). The Kullback-Leibner divergence is a measure that indicates how much a first distribution differs from a second distribution (Kullback & Leibler, 1951). The heights of the HE degree bins as shown in Figure 11 are compared to the heights of the HE degree bins of the sampled hypergraphs of the SHHCM, NHHCM, and CLHHM (see equation 16). The bins have a width of 25 and the last bin also contains the hyperedge degrees that are larger than 250.

The 95%-CIs of the HE degree distribution measures are shown in Table 7. Considering the max, min and mean of the hyperedge degree distribution, it seems that the CLHHM Model is the best model for reproducing the hyperedge degree distribution. It is remarkable that. SHHCM has better values for the hyperedge degree than the NHHCM, which might indicate that reshuffling does not lead to a better representation of the network, according to this measure. However, when looking at the standard deviation this is contradicted, since the the original value almost lies in the 95%-CI of the sampled network of NHHCM for the standard deviation, while the standard deviation of the sampled networks of the SHHCM and CLHHM are significantly lower than the original standard deviation. The same is concluded when comparing the KL-div. of the three models. The KL-div. for the NHHCM is considerably lower than the KL-div. of the SHHCM and CLHHM.

Centrality. By comparing the eigenvalue centralities between the original network and the sampled networks, we consider five measures that characterizes the eigenvector centrality. First, the (mean) largest eigenvalue centrality and the corresponding node are shown. Second, the mean eigenvalue centrality gap, the difference between the mean largest and the second mean largest eigenvalue centrality, is shown. Then, for the original collaboration

Model	Original	SHHCM	NHHCM	CLHHM
Max	242	(248.05, 257.75)	(273.89, 283.19)	(254.81, 265.19)
Min	1	(0.040, 0.240)	(-0.016, 0.096)	(0.762, 1.199)*
Mean	93.931	(86.987, 88.096)	(101.08, 101.51)	(90.592, 92.959)
Std	61.852	(54.908, 55.779)	(61.867, 62.423)	(54.420, 55.771)
KL-div	0	12.802	7.577	13.050

TABLE 7: The 95%-CI intervals for the characteristics of the HE degree distribution of the SHHCM, NHHCM and the CLHHM.
N = 50, $\alpha = 0.05$; two-tailed. *Original value is in CI.

network, we check the number of eigenvector centralities that lie within the 95%-CIs of the sampled networks of SHHCM, NHHCM and CLHHM and we express this by a percentage. Finally, we compute the mean squared error (MSE) of the means of vertex eigenvector centralities. The results are shown in Table 8.

The largest eigenvector centrality is almost the same for all models, the CLHHM is closest to the original graph. The node corresponding to the largest eigenvector is always the same. The eigenvalue gap differs much between the original network and the sampled networks. The CLHHM EC gap is closest to the original EC gap, even though the difference is significant. It appears that for the SHHCM, NHHCM and the CLHHM, in total 147, 140 and 206 of the 783 eigenvalue centralities lie within the 95%-CI intervals, respectively. When comparing the MSEs, the MSE of the sampled network of the SHHCM is closest to the original graph. Overall, we conclude that the CLHHM model is the best model to mimic the eigenvector centrality.

Model	Original	SHHCM	NHHCM	CLHHM
Largest EC	0.5166	0.5338	0.5474	0.5237
Node largest EC	528	528	528	528
EC Gap	0.0115	0.0591	0.0630	0.0561
Percentage in CI	100%	18.8%	17.9%	26.3%
MSE	0	$6.6457 * 10^{-5}$	$6.7739 * 10^{-5}$	$6.7545 * 10^{-5}$

TABLE 8: Comparison of characteristics of the eigenvalue centrality of the SHHCM, NHHCM and the CLHHM.

Modularity. The original network has a modularity of 0.10125 and the 95%-CIs for the sampled networks of the SHHCM, NHHCM and the CLHHM are, respectively, (-0.00966, 0.00224), (0.0442, 0.0548) and (-0.0236, -0.0122), these are plotted in Figure 14. It is remarkable that the modularity for SHHCM lies around 0, which means that there are no clusters. Furthermore, the modularity of the sampled network of the CLHHM model is negative, which indicate the opposite behaviour of the original network regarding modularity. So, the modularity of the NHHCM imitates the modularity of the original network best, the other two models fail to capture the slight modularity in the original graph.

Homophily The homophily indices of the network are also computed for the sampled networks. The 95%-CI are shown in Table 9. From this table it could be seen that for H_1, NHHCM is the best estimator followed by SHHCM and CLHHM, respectively. For H_2, SHHCM is best followed by CLHHM and NHHCM, respectively. For H_3, both NHHCM and CLHHM have an interval that contains index H_3. This is not the case

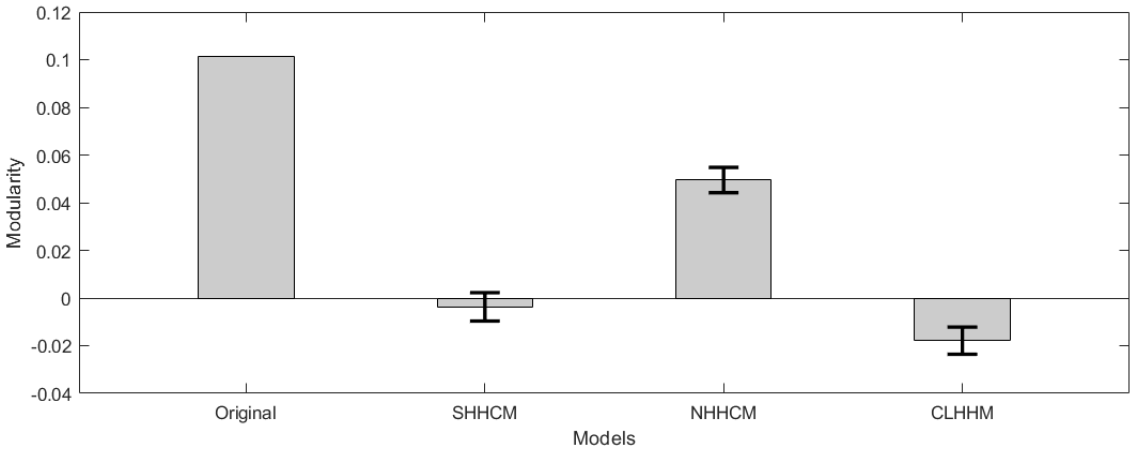


FIGURE 14: Modularity CIs of the SHHCM, NHHCM and the CLHHM.

for SHHCM. Finally, CLHHM is the best estimator for H_4, followed by SHHCM and NHHCM, respectively. The relative fraction of node type 1, 2, 3 and 4 are, 0.3461, 0.4368, 0.1967 and 0.02043, respectively. It should be noted that the conclusions about homophily change relative to the conclusions of the original graph when comparing H_2 of the sampled networks with the relative fraction. This is also the case for H_4, except for the CLHHM model. Based on the homophily indices, we cannot state which model is best. However, there is perhaps a slight preference for the CLHHM model. Even though most of the homophily indices do not fall in the CIs, the values are remarkably close.

Model	**Original**	**SHHCM**	**NHHCM**	**CLHHM**
H_1	0.2568	(0.2353, 0.2445)	(0.2433, 0.2545)	(0.2292, 0.2398)
H_2	0.4544	(0.4235,0.4319)	(0.4078, 0.4158)	(0.4200, 0.4285)
H_3	0.3640	(0.3645,0.3737)	(0.3581, 0.3657)*	(0.3612, 0.3704)*
H_4	0	(0.03285,0.08576)	(0.1622, 0.2341)	(0.001339, 0.006832)

TABLE 9: The 95%-CI intervals for the homophily indices of the SHHCM, NHHCM and the CLHHM.
N = 50, $\alpha = 0.05$; two-tailed. *Original value is in CI.

Value Chain Position. The three models are also implemented and evaluated with value chain position node types. There are 6 different value chain positions, and there were 4 project roles, so the number of different types is larger when considering value chain position node types. The degree sequence of the original network and the degree sequences of the networks produced by the SHHCM model and the NHHCM model are again the same, due to being a constraint. The degree sequences of the networks produced by the sampled networks of the CLHHM model are again variable, just as in the case with project role nodes. Therefore, for the sampled networks for each node a 95%-CI of the degree have been computed. We compute the number of times the original degree is in the 95%-CIs of the sampled networks of CLHHM. We see that in 45 of the 783 95%-CIs, the original degree is not contained in the CI, which is in $45/783 = 5.7\%$ of the cases. This percentage is slightly less than the percentage for the cases with project role node types, but still around the expected 5% (due to the 95%-CIs). The dimension sequences of the sampled networks are the same as the dimension sequence of the original network, which was also constrained, so expected.

Hyperedge degree. The characteristics of the HE degree distribution for the networks with value chain node types is shown in Table 10. Again the CLHHM model seems best at approximating the hyperedge degree, even though the SHHCM model comes close to the CLHHM model. When considering the standard deviation and the Kullback-Leibler divergence, the NHHCM model seems to be the best model. This result is the same as the result for the models with project role node types.

Model	**Original**	**SHHCM**	**NHHCM**	**CLHHM**
Max	242	(250.28, 259.60)	(264.54, 272.38)	(254.75, 264.77)
Min	1	(0.0139, 0.1861)	(0.0930, 0.3870)	(0.866 1.294)*
Mean	93.931	(87.352, 88.552)	(100.94, 101.41)	(90.884, 93.157)
Std	61.852	(54.604, 55.452)	(60.761, 61.441)	(54.919, 56.254)
KL-div	0	14.236	7.704	12.824

TABLE 10: The 95%-CI intervals for the characteristics of the HE degree distribution of the SHHCM, NHHCM and the CLHHM with VC node types.
N = 50, $\alpha = 0.05$; two-tailed. *Original value is in CI.

Centrality. The original eigenvector centrality is compared with the eigenvector centrality of the sampled networks of the SHHCM, NHHCM, and CLHHM, based on the five characteristics of the eigenvalue centrality as introduced for networks with the project role node types. The characteristics of the eigenvalue centrality are shown in Table 11. The largest EC are approximately the same as in the original graph. The node with the largest EC is the same for all the models and matches that of the original graph. The EC gap of the NHHCM and the CLHHM are significantly decreased in comparison to the SHHCM. The CIs of the eigenvector centralities of all nodes are most correct for the CLHHM. The MSE of the NHHCM is now the lowest, which is remarkable, since the MSE of the NHHCM is the highest for the network with project role node types. Based on the results one might conclude that the NHHCM model is the best model for modelling the eigenvector centrality of a network with a dominant node type.

Model	**Original**	**SHHCM**	**NHHCM**	**CLHHM**
Largest EC	0.5166	0.5350	0.5207	0.5031
Node largest EC	528	528	528	528
EC Gap	0.0115	0.0620	0.0245	0.0330
Percentage in CI	100%	17.8%	18.5%	24.8%
MSE	0	$6.7546 * 10^{-5}$	$5.9754 * 10^{-5}$	$6.7204 * 10^{-5}$

TABLE 11: Comparison of characteristics of the eigenvalue centrality of the SHHCM, NHHCM and the CLHHM with value chain position node types.

Modularity. The modularity of the original network with VC node types is 0.0905. The 95%-CIs of the modularity for the sampled networks of the SHHCM, NHHCM and CLHHM model are (0.0007, 0.0105), (0.2450, 0.2554), and (-0.0200, -0.0116). The CIs are shown in Figure 15. We see that the CI of SHHCM is around zero. The modularity for NHHCM is much higher, and the modularity for CLHHM is negative. This is deviating from the modularity measures found in the network with project role types. For the network with project role types, the modularity measures found for the NHHCM model were around 0.05. The modularity measures for value chain position node types, for the NHHCM model is around 0.25. A possible explanation could be that the NHHCM model

is inadequate when one node type is dominant.

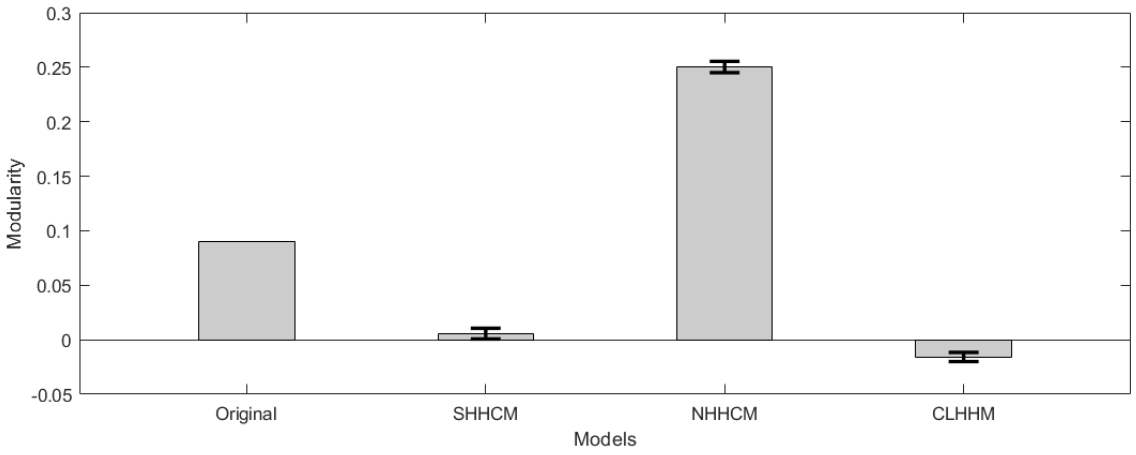


FIGURE 15: Modularity CIs of the SHHCM, NHHCM and the CLHHM with value chain position node types

Homophily. The homophily indices are illustrated in Table 12. We see that the 95%-CIs of the NHHCM model contain the original homophily H_3 and H_6. The original homophily H_6 is also in the 95%-CI of the SHHCM model. The other CIs of SHHCM, NHHCM and CLHHM, and the CIs do not contain the original homophily indices. So in this respect, it seems that NHHCM is the best model. However, when comparing homophily indices $H1$, $H2$, $H3$, and $H5$ with the CIs then we see that the CIs of the NHHCM model are deviating form the original homophily index. Therefore, the results are ambiguous. This result can also be observed, but to a lesser degree, in Table 9.

Model	**Original**	**SHHCM**	**NHHCM**	**CLHHM**
H_1	0.6409	(0.6286, 0.6348)	(0.7220, 0.7270)	(0.6198, 0.6267)
H_2	0.1280	(0.0983, 0.1273)	(0.0910, 0.1101)	(0.0770, 0.0930)
H_3	0.1548	(0.1339, 0.1456)	(0.1478, 0.1591)*	(0.1290, 0.1443)
H_4	0.1844	(0.1116, 0.1329)	(0.0808, 0.0942)	(0.1002, 0.1226)
H_5	0.1374	(0.1574, 0.1671)	(0.1900, 0.2010)	(0.1489, 0.1583)
H_6	0.0629	(0.0396, 0.0795)*	(0.0383, 0.0632)*	(0.0307, 0.0577)

TABLE 12: The 95%-CI intervals for the homophily indices of the SHHCM, NHHCM and the CLHHM Model for VC node types.
N = 50, $\alpha = 0.05$; two-tailed. *Original value is in CI.

Summary. When comparing the sampled networks of the three models with the original network, we cannot conclude which hypergraph model is modelling the collaboration network the best. In fact, all three heterogeneous hypergraph models are good models to configure a heterogeneous hypergraph. When considering the three main valuable measures, hyperedge degree distribution, modularity and homophily measures, the NHHCM models the collaboration network for the project role node types best, since it has the best standard deviation for hyperedge degree, it has the best modularity and half of the homophily are best modelled by the NHHCM model. However, when looking at the collaboration network with value chain position node types, the SHHCM model seems the best model. The SHHCM model models the max and the mean of the hyperedge degree most closely. Furthermore, the original modularity is falling within the CI and 4 out of

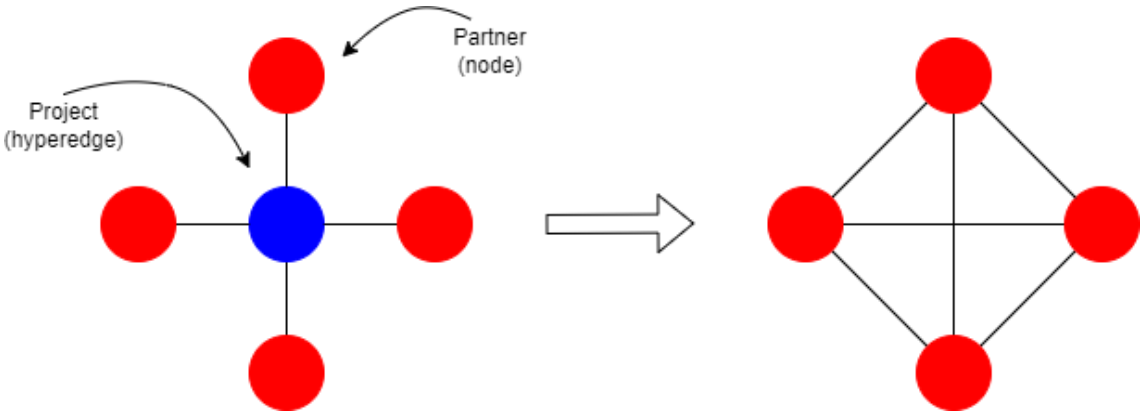


FIGURE 16: Transformation of hyperedge to a clique of nodes.

6 homophily indices are closest to the original homophily indices, of which 1 original homophily index (H_6) lies in the CI. This last result was not expected, but it seems that the combination of a dominant node type and reshuffling hyperedges lead to a deterioration of the original hypergraph characteristics. Perhaps, this leads to problems since reshuffling is based on picking randomly selected hyperedges to reshuffle. A solution could be to reshuffle the hyperedges that are close (in distance) to each other. Since all models appear to be good models to model heterogeneous hypergraphs, researches might want to choose the models based on the preference for certain constraints. When it is preferred to configure a network with an exact degree sequence, the SHHCM and the NHHCM are the best models. When an expected degree sequence is preferred the Chung-Lu model can be used.

4.2 Comparison of hypergraphs with normal graphs

In addition to the comparison between hypergraph models, as done in the previous section, we are also going to test the added value of hypergraphs in comparison to normal graphs. For clarity, a normal graph is defined as a graph with only dyadic relationships, no selfloops and no parallel edges. Each hypergraph created by a heterogeneous hypergraph model is transformed into a normal graph. To create a normal graph, each hyperedge is transformed into a clique of nodes that were connected by the hyperedge (see Figure 16). When all hyperedges are transformed, the selfloops and parallel edges are removed from the graph. The details of this process are given in Appendix E.2. The transformed original hypergraph consists of 4824 edges and 783 nodes. The network is illustrated in Figure 17. Figure 17 also includes a zoomed-in picture in which some cliques are visible.

Measures. The measures for the transformed original hypergraph are computed and both project role and value chain position types are considered. The degree sequence continues following a power law, except for low degree values (degree 1-5). Since every hyperedge is replaced by a clique, every vertex is automatically connected to all the other vertices in the clique, which lead to an increase of the degree, when the dimension of the hyperedge is greater than 2. Since many hyperedges have a dimension larger than 2, the vertex degree often increases, and therefore low degrees do not appear often. Besides the degree sequence, the dimension sequence also changes. Since the transformed graph only has dyadic relationships, the dimension of each hyperedge is 2. When considering the hyperedge degree distribution, we see that the HE degree distribution of the normal graph ranges between 2 and 517, where in the original network the hyperedge degree distribution ranges between

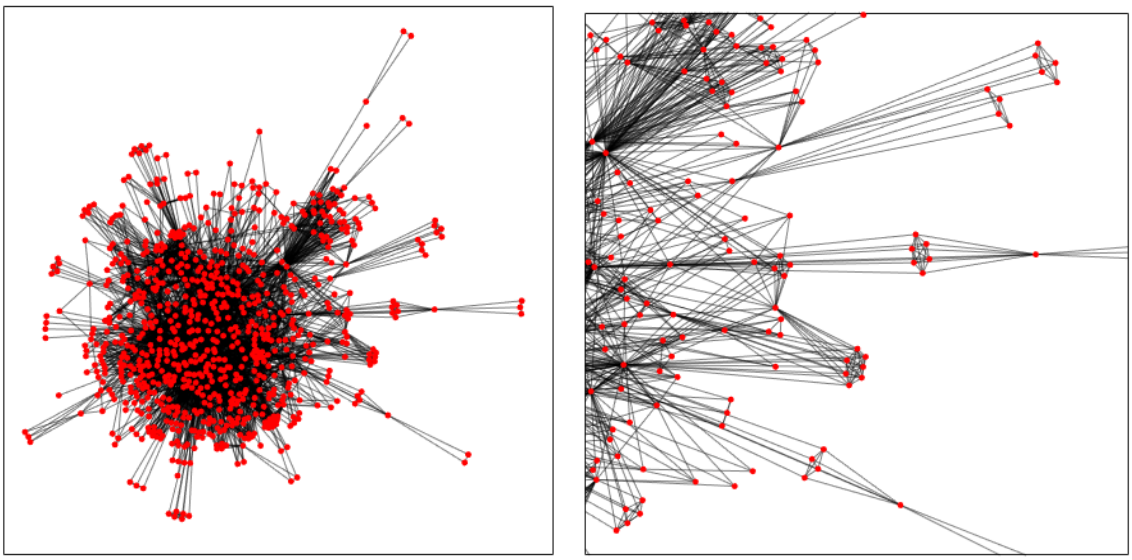


FIGURE 17: Normal and Zoomed-in Transformed Collaboration Network

1 and 242. Just as in the hypergraph, the hyperedge degree distribution does not follow a smooth distribution. The eigenvalue centrality for the normal graph is particularly high for one vertex, 0.9932, and below 0.1 for the rest of the eigenvalue centralities. For the hypergraph this was more insightful, since the 9 largest eigenvalue centralities were above 0.1. The modularity of the the original hypergraph was 0.1013 for project role types and 0.0905 for value chain position types, and for the transformed hypergraph the modularities are 0.0263 and 0.0268, respectively. So the modularity is strongly reduced by transforming the hypergraph into a normal graph. Furthermore, using normal graphs lead to the conclusion that there is no modularity. The homophily indices do not change much for project role types. For project role types, H_1, H_2 and H_3 are changing with percentages of 4.7%, -2.0% and -8.8%, respectively. Homophily index 4 does not change and remains 0. For value chain position types, the indices 1 to 6 are changing with percentages of -0.05%, 32.2%, -19.7%, -19.5%, -3.8%, -13.2%, which are quite large percentages. However, in this particular case the conclusions about homophily do not change. So when comparing the measures of the original graph with the measures of the transformed graph, it appears that some measures are changing significantly. Furthermore, the hypergraph measures make much more sense when simulating a collaboration network. In a hypergraph the projects are seen as a separate entity, instead of a set of dyadic relationships between project members. Furthermore, project sizes, dimension, is more easily dealt with and one may see the degree as the number of projects a company is part of. Therefore, it could be concluded that hypergraphs are better representations for collaboration networks than normal graphs.

Models. We have now concluded that a measure over a hypergraph differs from a measure over a normal graph. However, we did not conclude whether using hypergraph models for hypergraphs are better than using normal graph models. If a normal model is really good, then perhaps there is a need for finding methods to transform a normal graph back to a hypergraph, instead of finding a good hypergraph model. To test this, three cases are compared for the three models (SHHCM, NHHCM and CLHHM). Furthermore, we are going to compare between the two node types (project role and value chain position). The cases we are going to compare are:

C1: The transformed original graph.

C2: A transformed graph that results from the heterogeneous hypergraph model with as input the original graph.

C3: A transformed graph that results from the heterogeneous hypergraph model with as input the transformed original graph.

The heterogeneous hypergraph models refer to the SHHCM, NHHCM and CLHHM model. A hypergraph is a result of these hypergraph models and this is transformed into a normal graph. Case 2 and case 3 are both implemented 50 times. The full length code of this process is given in Appendix E.3.The measures that are used to compare these cases are degree, hyperedge degree, eigenvector centrality, modularity and homophily.

Dimension per vertex. Before comparing the results for the three cases, we would like to introduce an aspect of the normal graph that is available in the degree sequence when transforming a hypergraph into a normal graph, but that is not directly available in the degree sequence and the dimension sequence of a hypergraph. The process of transforming the degree and dimension sequence of a hypergraph into the degree and dimension process of a normal graph is shown in Figure 18. When transforming a hypergraph into a normal

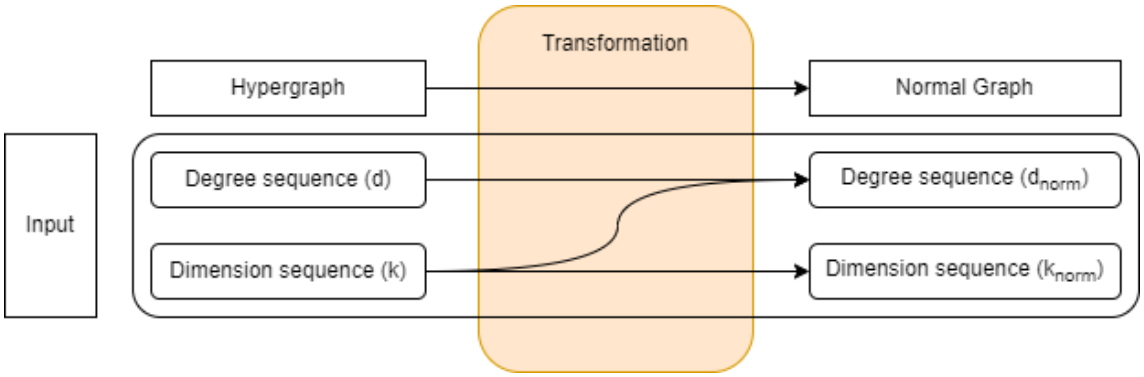


FIGURE 18: Transformation of model input from a hypergraph to a normal graph

graph, every hyperedge becomes a clique. Trivially, the degree distribution of a vertex (d) will affect the degree distribution of a vertex in a transformed hypergraph (d_{norm}). However, the dimension sequence also affects d_{norm}. When a hyperedge is transformed into a clique, the vertex degree of the normal graph becomes the dimension of the hyperedge minus 1 ($d_{norm} = k - 1$), since in the clique the vertex connects to all other vertices in the hyperedge and it does not connect to itself. Therefore, the degree sequence of a transformed hypergraph (d_{norm}) is affected by both the degree sequence (d) and the dimension sequence (k) of the hypergraph. When e_j defines the j-th hyperedge ($j = 1, ..., m$), then, for all vertices $i = 1, 2, ...n$, the degree of a vertex v_i in the transformed graph is computed as

$$d_{norm,i} = \sum_{j=1}^{m} \mathbb{1}(v_i \in e_j) \cdot (k_i - 1) \tag{18}$$

Summarized, this means that the degree sequence of the normal graph contains a part of the information about which dimension belongs to which vertices. The nodes with high degrees in the transformed graphs, where part of the hyperedges with high dimensions, since there is a direct relation between $d_{norm,i}$ and k_i, as shown by Equation 18. The dimension per

vertex was not available as input for the hypergraph model. In the hypergraph model, the degree sequence and the dimension were dealt with as separate input variables. Perhaps the presence of this extra information is also affecting the outcomes of case 2 and case 3. Case 3 will have the extra information of the dimension per vertex and will therefore have an advantage in comparison to case 2. The dimension sequence in the transformed graph (k_{norm}) loses information in the transformation process, since the dimension of every edge is 2. However the information lost by k_{norm} is lower than the information gained by d_{norm}.

Vertex degree. The overview of the mean, standard deviation and mean squared error (MSE) of the vertex degree distribution is illustrated in Table 13. We see that the CIs overlap with each other comparing project role types and value chain position types. The mean of the degree of case 2 is higher than the original degree mean, and the mean of the degree of case 3 is lower than the original degree mean. The same trend could also be seen within the standard deviation. For the mean and standard deviation we see that case 2 is closer to the original mean and standard deviation than case 3. This could be explained by the number of edges in the sampled networks in relation to the the number of edges in the original transformed network. Case 1 has 4824 edges and the 95% confidence intervals of the number of edges for case 2 and case 3 are (5008.1, 5386.9) and (4311.3, 4454.4), respectively. When there are more edges and the same amount of vertices, the mean degree automatically increases. Within transforming a hypergraph to a normal graphs, parallel edges and selfloops are removed. Since in case 2 only one transformation takes place (at the end), less parallel edges and selfloops are removed than for case 3. In case 3, two transformations take place, the hypergraph input is transformed and the resulting model is transformed. Therefore there are more edges in the sampled networks of C2 than in the sampled networks of C3.

Type	**Case**	**Mean**	**Std**	**MSE**
n.a.	C1 Original	12.322	21.105	0.000
PR	C2 SHHCM	(13.122, 13.198)	(23.603, 23.792)	(58.356, 60.896)
	C2 NHHCM	(13.721, 13.760)	(25.504, 25.655)	(74.005, 76.577)
	C2 CLHHM	(12.792, 12.909)	(23.616, 23.804)	(99.456, 103.020)
	C3 SHHCM	(11.356, 11.378)	(16.195, 16.301)	(28.959, 30.329)
	C3 NHHCM	(11.012, 11.035)	(14.623, 14.733)	(50.921, 52.820)
	C3 CLHHM	(11.315, 11.338)	(16.343, 16.451)	(39.890, 41.813)
VC	C2 SHHCM	(13.159, 13.232)	(23.755, 23.945)	(59.850, 62.075)
	C2 NHHCM	(13.735, 13.776)	(24.997, 25.141)	(83.031, 86.393)
	C2 CLHHM	(12.733, 12.841)	(23.594, 23.760)	(98.327, 102.020)
	C3 SHHCM	(11.343, 11.367)	(16.211, 16.319)	(28.698, 30.145)
	C3 NHHCM	(10.950, 10.974)	(14.346, 14.463)	(55.706, 57.802)
	C3 CLHHM	(11.289, 11.317)	(16.333, 16.469)	(39.486, 41.378)

TABLE 13: CIs for mean, standard deviation and MSE of degree sequences: C1, C2 and C3.
N = 50, $\alpha = 0.05$; two-tailed. *C1 lies in CI

When comparing the MSEs, the MSEs of case 3 are lower than the MSEs of case 2. Therefore, it seems that case 3 is better than case 2 in reproducing the degree sequences. This could be a result of the extra information (dimension per vertex) that case 3 possesses. Since case 3 has an indication of the dimension per vertex the cliques are much more accurate and this has a positive affect on reproducing the degree sequences.

Comparing between the different models the mean and the standard deviation of the models are best estimated by the CLHHM model followed by the SHHCM and NHHCM model. The MSEs of the SHHCM model are the lowest. It is remarkable that for case 3 the CLHHM model is better than the CLHHM model for case 2, it is even better than the NHHCM model in case case 3. So the CLHHM model is better when having normal graph input when preserving degree is preferred.

Hyperedge Degree. The hyperedge degree distribution is again analysed by 5 measures, maximum HE degree (max), minimum HE degree (min), mean HE degree, standard deviation (std), and the Kullback-Leibner Divergence (KL-div.). Since the Kullback-Leibner Divergence requires nonempty bins, we use bins of size 50 (for the hypergraphs in Section 4.1 a bin width of 25 is used). The 95%-CIs of these measures are shown in Table 14.

Type	**Case**	**Max**	**Min**	**Mean**
	C1	517.0	2.0	94.849
PR	C2 SHHCM	(524.692, 536.188)	(0.358, 0.882)	(108.931, 110.212)
	C2 NHHCM	(586.766, 596.074)	(0.140, 0.581)	(120.062, 121.153)
	C2 CLHHM	(492.296, 504.584)	(1.557, 2.123)*	(110.379, 111.857)
	C3 SHHCM	(357.756, 362.804)	(2.714, 3.126)	(66.841, 67.415)
	C3 NHHCM	(309.608, 315.352)	(2.648, 3.152)	(58.809, 59.368)
	C3 CLHHM	(351.322, 359.958)	(1.831, 2.329)*	(67.758, 68.392)
		Std	**KL-div**	
	C1	89.275	0	
	C2 SHHCM	(90.547, 92.138)	655.26	
	C2 NHHCM	(100.816, 102.059)	1335.09	
	C2 CLHHM	(83.758, 85.431)	653.88	
	C3 SHHCM	(59.548, 60.308)	534.17	
	C3 NHHCM	(50.942, 51.678)	426.96	
	C3 CLHHM	(58.350, 59.482)	456.99	
		Max	**Min**	**Mean**
VC	C2 SHHCM	(529.125, 540.675)	(0.268, 0.772)	(109.872, 111.148)
	C2 NHHCM	(576.107, 585.693)	(0.408, 0.952)	(116.242, 117.315)
	C2 CLHHM	(483.734, 500.106)	(1.721, 2.359)*	(110.553, 111.796)
	C3 SHHCM	(357.628, 363.612)	(2.785, 3.175)	(66.962, 67.540)
	C3 NHHCM	(296.571, 302.229)	(2.610, 2.990)	(57.441, 58.033)
	C3 CLHHM	(353.979, 360.221)	(1.478, 2.002)*	(67.752, 68.551)
		Std	**KL-div**	
	C2 SHHCM	(91.369, 93.068)	671.53	
	C2 NHHCM	(99.245, 100.556)	1192.98	
	C2 CLHHM	(83.379, 85.384)	630.11	
	C3 SHHCM	(59.759, 60.520)	517.09	
	C3 NHHCM	(49.514, 50.261)	422.17	
	C3 CLHHM	(58.935, 59.979)	456.19	

TABLE 14: CIs for hyperedge degree measures: C1, C2 and C3.
N = 50, $\alpha = 0.05$; two-tailed. *C1 lies in CI

The differences between project role types and value chain types are minimal. The

maximum of the HE degree are for case 2 much closer to case 1 than case 3. Perhaps this is partly explained by the the number of hyperedges (m), but still the differences are quite large. The minimum HE degree for SHHCM and NHHCM indicate that not all vertices are part of the giant component. This seems not to be the case for the CLHHM model, where the minimum HE degree falls within the CI. The minimum HE degree for case 3 are higher for SHHCM and NHHCM than the minimum HE degree for case 1. The minimum HE degrees fall within the CIs for the CLHHM model. A reason for this might be that the density of the network is more equally distributed due to a more general input, a normal graph. The mean of the HE degree is higher for case 2 than for case 1, since the maximum HE degree is also a little higher. In the same line of reasoning, the mean of the HE degree for case 3 are lower. The standard deviation for case 2 is quite close to the standard deviation for case 1, but it does not fall within the CI. The standard deviation of the HE degree for case 3 are quite deviating from the standard deviation of the HE degree for case 1. Perhaps this is partly a consequence of a smaller range between the maximum and the minimum of the HE degree. On the other hand, this might also indicate that hypergraph models are preferred to normal graph models. The KL-div. indicates the opposite of the max, min, mean and std. The KL-div. for case 3 are better than the KL-div. for case 2. Since the KL-div. considers the complete hyperedge distribution, it seems that the hyperedge degree distribution is better imitated by case 3 than case 2.

Centrality. We are comparing five characteristics of the eigenvalue centrality for the the three cases and two node types. These characteristics are the largest EC, the node with the largest EC, the EC gap (difference between highest and second highest EC), the percentage of vertices for which the original eigenvalue centrality is within the CIs of the sampled networks (PCI), and the mean squared error of the means of the eigenvalue centrality. The results are shown in Table 15. We see that the vertex with the highest eigenvalue centralities for case 1 is also the vertex with the highest eigenvalue centrality for case 3, but this vertex has the third highest eigenvalue centrality for case 2 (not in the Table). The highest eigenvalue centrality has a value of around 1 for case 1 and the rest of the eigenvalues are below 0.1. The EC gaps are also close to the highest eigenvalues. The highest eigenvalue centrality for case 2 for the three models are 0.6402, 0.8494, and 0.6058. So NHHCM has a centrality around 0.85 and SHHCM and CLHHM have a eigenvalue centrality around 0.6. The rest of the eigenvalue centralities are below 0.1, which could also be seen by the EC gap. The eigenvalue centrality for case 3 are 0.7803, 0.8165 and 0.8154. So all the the three models have eigenvalue centralities around 0.8. The second highest eigenvalue centrality also has a value above 0.1, but the rest of the eigenvalue centralities are below 0.1. Comparing the PCIs, the results are ambiguous. The PCIs for the CLHHM are higher for case 2 for both node types. The PCIs for the SHHCM and NHHCM are higher in case 3, except for the SHHCM with project role node types. Based on the MSE it is clear that case 3 models the eigenvalue centralities best.

Modularity. The CIs of the modularity of the models for the different node types are shown in Table 16. We see that the modularities are all very low, for both case 2 and case 3. So, the modularity is modelled well by all cases. For case 2 some sampled networks also have negative modularity bounds. In general, the modularity values for case 3 are significantly higher than the modularity values of case 2, except for the NHHCM with value chain position node types. The modularity values for case 3 are closer to the original modularity values. However, no CIs contain the original modularities, so the modularity values of the sampled networks significantly differ from the modularity values of the original normal graph.

Homophily. Finally, the homophily indices are compared for the three cases, three

T	C	Measure	Original	SHHCM	NHHCM	CLHHM
PR	C2	Largest EC	0.9932	0.6402	0.8494	0.6058
		Node largest EC	684	528	528	528
		EC Gap	0.9136	0.6264	0.7452	0.3920
		Percentage in CI	100%	23.0%	7.4%	17.4%
		MSE	0	$1.7319 * 10^{-3}$	$1.8766 * 10^{-3}$	$1.2074 * 10^{-3}$
PR	C3	Largest EC	0.9932	0.7803	0.8165	0.8154
		Node largest EC	684	684	684	684
		EC Gap	0.9136	0.6785	0.5877	0.6578
		Percentage in CI	100%	17.9%	11.6%	13.3%
		MSE	0	$6.2048 * 10^{-5}$	$7.6108 * 10^{-5}$	$5.2513 * 10^{-5}$
VC	C2	Largest EC	0.9932	0.6309	0.9968	0.4652
		Node largest EC	684	528	528	528
		EC Gap	0.9136	0.5475	0.9643	0.2439
		Percentage in CI	100%	19.5%	2.7%	23.5%
		MSE	0	$1.5282 * 10^{-3}$	$2.3835 * 10^{-3}$	$1.0094 * 10^{-3}$
VC	C3	Largest EC	0.9932	0.6219	0.5147	0.7036
		Node largest EC	684	684	684	684
		EC Gap	0.9136	0.4972	0.3714	0.4066
		Percentage in CI	100%	24.3%	26.7%	10.3%
		MSE	0	$1.8192 * 10^{-4}$	$3.0985 * 10^{-4}$	$1.7611 * 10^{-4}$

TABLE 15: Comparison of characteristics of the eigenvalue centrality of the SHHCM, NHHCM and the CLHHM, C1, C2 and C3

Case	Modularity (Q), T = PR	Modularity (Q), T = VC
C1	0.02634	0.02683
C2 SHHCM	(0.00530, 0.00910)	(-0.00615, -0.00381)
C2 NHHCM	(0.00450, 0.00780)	(0.05983, 0.06291)
C2 CLHHM	(-0.00050, 0.00320)	(-0.00027, 0.00169)
C3 SHHCM	(0.01650, 0.01930)	(0.01517, 0.01820)
C3 NHHCM	(0.00950, 0.01290)	(0.05734, 0.05984)
C3 CLHHM	(0.01450, 0.01670)	(0.01450, 0.01672)

TABLE 16: CIs for modularity: C1, C2 and C3.
N = 50, $\alpha = 0.05$; two-tailed. *C1 lies in CI

models, and two project types. The homophily indices are illustrated in Table 17. The CIs of case 2 and case 3 for the homophily indices of the networks with the project role node types are not far from the the homophily indices of case 1. Still, for case 2 none of the CIs are containing the values of case 1. For case 3 a few more CIs containing the value of case 1. Especially the CLHHM model seems to imitate the homophily indices well, three out of four CIs contain the value for case 1. For the value chain position type, also case 2 has CIs which contain the values for C1. Still it seems that case 3 is better in imitating the homophily indices of the original network than case 2, as expected by the conclusion about modularity.

Case T = PR	**C1**	**C2 SHHCM**	**C2 NHHCM**	**C2 CLHHM**
H1	0.2691	(0.2487, 0.2556)	(0.2455, 0.2516)	(0.2408, 0.2493)
H2	0.4454	(0.4219, 0.4278)	(0.4251, 0.4305)	(0.4154, 0.4233)
H3	0.3346	(0.3129, 0.3183)	(0.3122, 0.3169)	(0.3067, 0.3140)
H4	0.0000	(0.0340, 0.0869)	(0.1144, 0.1456)	(0.0026, 0.0084)
		C3 SHHCM	**C3 NHHCM**	**C3 CLHHM**
H1	0.2691	(0.2709, 0.2773)	(0.2685, 0.2734)*	(0.2653, 0.2729)*
H2	0.4454	(0.4464, 0.4510)	(0.4322, 0.4370)	(0.4448, 0.4511)*
H3	0.3346	(0.2983, 0.3030)	(0.2965, 0.3013)	(0.2965, 0.3018)
H4	0.0000	(-0.0003, 0.0010)*	(0.0339, 0.0495)	(0.0000, 0.0000)*
Case T = VC	**C1**	**C2 SHHCM**	**C2 NHHCM**	**C2 CLHHM**
H1	0.6406	(0.6393, 0.6430)*	(0.7457, 0.7510)	(0.6429, 0.6493)
H2	0.1693	(0.0420, 0.0497)	(0.0859, 0.0970)	(0.0719, 0.0810)
H3	0.1244	(0.0960, 0.1031)	(0.1302, 0.1374)	(0.0983, 0.1056)
H4	0.1485	(0.0427, 0.0493)	(0.0776, 0.0871)	(0.0699, 0.0793)
H5	0.1322	(0.1065, 0.1106)	(0.1309, 0.1368)*	(0.1036, 0.1087)
H6	0.0546	(0.0091, 0.0162)	(0.0383, 0.0572)*	(0.0231, 0.0344)
		C3 SHHCM	**C3 NHHCM**	**C3 CLHHM**
H1	0.6406	(0.6504, 0.6537)	(0.7229, 0.7257)	(0.6504, 0.6543)
H2	0.1693	(0.1578, 0.1696)*	(0.1158, 0.1259)	(0.1587, 0.1683)
H3	0.1244	(0.1076, 0.1152)	(0.1225, 0.1304)*	(0.1042, 0.1121)
H4	0.1485	(0.1127, 0.1226)	(0.0970, 0.1059)	(0.1091, 0.1188)
H5	0.1322	(0.1001, 0.1052)	(0.1252, 0.1312)	(0.0986, 0.1037)
H6	0.0546	(0.0490, 0.0637)*	(0.0550, 0.0696)	(0.0450, 0.0591)*

TABLE 17: CIs for homophily indices: C1, C2 and C3.
N = 50, $\alpha = 0.05$; two-tailed. *C1 lies in CI

Summary. Comparing case 2 and case 3 with case 1, in general it might be concluded that the sampled networks of case 3 are more equal to the network of case 1 than the sampled networks of case 2. This would indicate that normal graph models are preferred over hypergraph models. However, as already mentioned, when comparing the degree sequences, the input for case 2 contains an aspect which is not present in the input for case 3 that might be a crucial element of a hypergraph network, the dimension per vertex. This feature is defined as the average dimension of the hyperedges a specific vertex is part of. More precisely, for a vertex v_i, $i = 1, 2, ..., n$, the average dimension $k_{avg,i}$ is defined as

$$k_{avg,i} = \frac{1}{d_i} \sum_{j=1}^{m} \mathbb{1}(v_i \in e_j) k_j. \tag{19}$$

The average dimension sequence k_{avg} has size n, the number of vertices. Since, this measure is present in the input of case 3, but not in the input of case 3, the cases can actually not be compared fairly. Therefore, it cannot be concluded whether normal graph models are preferred over hypergraph models. However, we might advocate to use hypergraph models, since it is difficult to transform a normal graph into a hypergraph. When transforming a

normal graph to a hypergraph, one needs to analyse the normal graph for cliques, which is a complex computational problem (Bomze, Budinich, Pardalos, & Pelillo, 1999). Based on this section, it can be concluded that the representation of a collaboration network by a hypergraph is better than a normal graph, since measures of the network can be captured better in this way. Furthermore, hypergraph models could also better be used than normal graph models, since once a graph is configured it is hard to transform the normal graph back to a hypergraph.

4.3 Added value of heterogeneity function

In order to be able to transform the hypergraph models into heterogeneous hypergraph models in Section 3.4, certain assumptions were made in order to make the hypergraph models applicable to networks with heterogeneous nodes. The main idea for the heterogeneous hypergraph model is based on the idea of stochastic block models (SBMs). In addition, the formula in Equation 15 is proposed to facilitate the extension. In this section, we test whether these ideas are useful to create a heterogeneous hypergraph. In order to test this, the three hypergraph models are implemented with as input for Ω a matrix with equal probabilities. This means that the types have no preferences when connecting with different types of nodes. The sampled graphs are referred to as non-heterogeneous hypergraphs. The measures that are used to compare the graphs are the vertex degree distribution, dimension, hyperedge degree distribution, modularity and homophily.

Vertex Degree. The degree and the dimension are constrained by the models, therefore the degree and the dimension do not differ between the heterogeneous and non-heterogeneous hypergraphs, except for the degree sequences of the sampled networks of the CLHHM models. When comparing the confidence intervals of the degree sequences of the heterogeneous and non-heterogeneous hypergraphs of the networks with project role and value chain position node types, we see that for the heterogeneous graphs 736 and 738 out of the 783 cases, the vertex degrees are within the CIs. So 6.0% and 5.7% of the cases do not contain the original vertex degrees, respectively. For non-heterogeneous graphs this is 747 and 735 out of the 783 cases. So 4.6% and 6.1% of the cases do not contain the original vertex degrees, respectively. The percentages are all around 5%, which is explainable by the 95%-CIs.

Hyperedge Degree. Comparing the hyperedge degrees between the heterogeneous and non-heterogeneous hypergraph does not show significant differences. Comparing the KL-div. between the heterogeneous and non-heterogeneous hypergraphs, there is not a clear preference for one of the two. When a preference needs to be given than the heterogeneous hypergraph models are preferred, since the KL-div. is in four out of the six models better for heterogeneous models. The precise measures are given in Appendix B.

Modularity. More interesting are the modularity and homophily indices, since these measures depend on the different node types. The results for modularity are striking (see Table 18). For the CLHHM model it seems that indeed the heterogeneous hypergraph model is adding value. The differences between the CIs of the modularity are significant, and the modularity of the non-heterogeneous hypergraphs are more negative than the modularity of the heterogeneous hypergraphs. The confidence intervals for the modularity of the hypergraphs produced by the SHHCM and NHHCM model are also significantly different. So it can be concluded that the heterogeneity method is adding something to the hypergraph model. It seems that for the SHHCM model, the hypergraph is declustered with the heterogeneous hypergraph model, and for the NHHCM model, the model is clustered with the hypergraph model. We see that for the SHHCM model, the modularity increases when there are no preferred types to collaborate with.

The algorithm of the SHHCM is working such that nodes are picked from a set of remaining stubs based on the preference nodes have for other nodes types. When there is no preference for a certain node type, then the chances for picking a specific node are equally distributed. When there are in total more nodes of a specific type (there is a dominant type), then these nodes are picked more often in the beginning. Over time this effect will reduce, since the set with remaining stubs will contain less nodes with the dominant node type, since these nodes are already picked at the beginning. The nodes that are picked at the beginning are of the same types, and therefore they form a cluster with the dominant node types. This could be seen for example in the results of the sampled networks with value chain position types. in these networks there was also a dominant type. In practice, this is a strange result, since one would expect that clusters of the same type of project partners will not arise when there is no preference between different project partners. Fortunately, we see that the modularity sampled networks of the NHHCM decreases when there are no preferred node types. This means that the reshuffling method of the NHHCM is correcting the mistake in the SHHCM for non-heterogeneous graph models. The reshuffling method is working such that when two hyperedges are selected that belong to the same clusters, this does not have consequences for the clusters. When there is a preference between nodes, a network with low modularity will become clustered due to the NHHCM model by the opposite line of reasoning as for a network without preferences between nodes. In practice, this is also what we want, since it is logical to assume that project partners that prefer each other are going to form clusters. The footnote should be made that we should pick node types, for which we know this behaviour appears. Hopefully, this can be seen from the homophily indices. Overall the CIs of the modularity for the SHHCM and NHHCM are strongly indicating that the heterogeneous hypergraph model is working well with respect to the modularity.

T	**Meas.**	**H/N**	**Original**	**SHHCM**	**NHHCM**	**CLHHM**
PR	Q	H	0.101	(-0.010, 0.002)	(0.044, 0.055)	(-0.024, -0.012)
		N		(0.042 , 0.053)	(0.003, 0.014)	(-0.046, -0.033)
VC	Q	H	0.090	(0.001 , 0.010)	(0.245, 0.255)	(-0.020, -0.012)
		N		(0.277 , 0.290)	(0.002, 0.010)	(-0.043, -0.035)

TABLE 18: CIs for modularity of heterogeneous (H) and non-heterogeneous (N) hypergraphs.
N = 50, $\alpha = 0.05$; two-tailed. *Original value is in CI.

Homophily. The results for the CIs of the homophily indices are shown in Table 19. Most CIs are significantly different from each other, when comparing the heterogeneous and non-heterogeneous hypergraphs models. In comparison, the heterogeneous hypergraphs are better at imitating the original hypergraph, even though the homophily indices of the original hypergraph are often not lying within the CIs. This is an indication that the heterogeneous hypergraph models are better than the non-heterogeneous hypergraph models. For the SHHCM, the CIs for the non-heterogeneous hypergraph models are higher than the the heterogeneous hypergraph models and the opposite is true for the NHHCM. Furthermore, we see that all CIs of the non-heterogeneous SHHCM model are higher than the CIs of the NHHCM, and in many cases the CIs drop to almost zero, especially in the models with value chain position node types. This confirms the explanation for the differences in modularity as explained when discussing the modularity results. When we choose as input that there are preferences between nodes, then the SHHCM model produces a network with no clusters and it might be that in some cases the homophily indices

are wrong. The NHHCM model corrects these mistakes with the heterogeneous reshuffling method. In addition, when we choose as input that there are no preferences between nodes, then the SHHCM model produces a clustered network which might result in wrong homophily measures. The NHHCM model corrects this by setting the modularitites and the homophily indices to values closer to zero. This indicates that NHHCM is a much better model for homophily measures than the SHHCM model. The CLHHM model is slightly better for heterogeneous graphs than for non-heterogeneous graphs.

T	Meas.	H/N	Orig.	SHHCM	NHHCM	CLHHM
PR	H1	H	0.2568	(0.2353, 0.2445)	(0.2432, 0.2544)	(0.2292, 0.2398)
		N		(0.2788, 0.2905)	(0.2027, 0.2123)	(0.2110, 0.2217)
	H2	H	0.4544	(0.4235, 0.4319)	(0.4078, 0.4158)	(0.4200, 0.4285)
		N		(0.4270, 0.4345)	(0.3946, 0.4024)	(0.4030, 0.4134)
	H3	H	0.3640	(0.3645, 0.3737)	(0.3581, 0.3657)*	(0.3612, 0.3704)*
		N		(0.3731, 0.3809)	(0.3588, 0.3666)*	(0.3608, 0.3694)*
	H4	H	0.0000	(0.0328, 0.0858)	(0.1622, 0.2341)	(0.0013, 0.0068)
		N		(0.1500, 0.1889)	(0.0041, 0.0138)	(0.0008, 0.0228)
VC	H1	H	0.6409	(0.6286, 0.6347)	(0.7220, 0.7270)	(0.6198, 0.6267)
		N		(0.7509, 0.7565)	(0.6086, 0.6137)	(0.6122, 0.6196)
	H2	H	0.1280	(0.0983, 0.1273)	(0.0910, 0.1101)	(0.0770, 0.0930)
		N		(0.1351, 0.1498)	(0.0331, 0.0427)	(0.0353, 0.0502)
	H3	H	0.1548	(0.1338, 0.1456)	(0.1478, 0.1591)*	(0.1289, 0.1443)
		N		(0.2092, 0.2215)	(0.0974, 0.1075)	(0.1169, 0.1288)
	H4	H	0.1844	(0.1116, 0.1329)	(0.0808, 0.0942)	(0.1002, 0.1226)
		N		(0.1386, 0.1523)	(0.0375, 0.0467)	(0.0425, 0.0537)
	H5	H	0.1374	(0.1574, 0.1671)	(0.1900, 0.2010)	(0.1489, 0.1583)
		N		(0.2674, 0.2789)	(0.1456, 0.1531)	(0.1525, 0.1621)
	H6	H	0.0629	(0.0396, 0.0795)*	(0.0383, 0.0632)*	(0.0306, 0.0577)
		N		(0.1122, 0.1394)	(0.0040, 0.0120)	(0.0052, 0.0126)

TABLE 19: CIs for homophily indices of heterogeneous (H) and non-heterogeneous (N) hypergraphs.
N = 50, $\alpha = 0.05$; two-tailed. *Original value is in CI.

Summary. When comparing heterogeneous hypergraph models with non-heterogeneous hypergraph models, the modularity and homophily measures give strong indications that the NHHCM and the CLHHM are good models for modelling heterogeneous hypergraphs. We see that the sampled networks of the NHHCM are correcting the sampled networks of the SHHCM based on the propensity matrix Ω. The vertex degree distribution and the vertex degree distribution are not significantly different between the non-heterogeneous and heterogeneous hypergraph models.

5 Discussion

Contributions. This thesis has several contributions. First, it starts by giving insight into the University-Industry collaboration network (UICN) for nanotechnology R&D research. We show that the degree sequence follows a power-law, the hyperedge degree does not follow a specific pattern, the centralities show the most important partners in the network

and it does not have to be that the nodes with the highest degree have the highest centrality measures. The modularity shows that the network is slightly clustered for both project role types and value chain position types. Previous research is often focused on the effect of these roles on value creation and not on the effect of these roles on the network itself (Raesfeld et al., 2012; Oukes, 2018). Second, the field of hypergraph models is relatively undeveloped and in this research a short overview is given of the main contributions in this field. The overview by Battiston et al. (2020) is more extended, but quite technical, therefore not as accessible for all fields of science. Third, this thesis gives an insight in the advantages of hypergraph models, in which degree and dimension are constrained. Furthermore, the advantages are explained of constraining the hypergraph models with the exact vertex degree distribution as in the SHHCM and NHHCM and the advantages of constraining the hypergraph models with the expected degree distribution are explained. Fourth, some concepts or ideas in this thesis are not mathematically proven, but developed based on general ideas. Since the field of hypergraphs is quite unexplored, some concepts or ideas do not exist in literature. So, it requires creativity to find define these concepts and ideas. Equation 9, Equation 11 and Equation 15 are results of such ideas. Equation 9 and Equation 11 are representing the equations for the homophily index variables a_{ij}, based on a_i for normal graph models. Equation 15 defines the probability an vertex joins an already existing hyperedge. The heterogeneous hypergraph models are mainly based on this last equation. Fifth, this thesis makes a contribution to the existing hypergraph models by presenting models that take into consideration the heterogeneity of nodes by introducing a method that is based on a simplified form of a stochastic block model. In the literature, sometimes heterogeneity in graphs is not possible due to combinatorial computational issues (Veldt, Benson, & Kleinberg, 2023; Casiraghi, 2019). Furthermore, methods are often really extended and not easily applicable (Eikmeier et al., 2018; Giroire et al., 2021). We introduce a much simpler method, which is also a good model for clustering known node types.

Contributions to business. It has three contributions for business. First, this thesis makes the theory about hypergraphs accessible for application in business, where existent literature is not laying this bridge. Second, it gives a good explanation of why hypergraphs are good representations for collaboration networks. In mathematics this argument was already present (Benson et al., 2016; Ghoshal et al., 2009), but applied to business this was less clear. Third, it provides a model for business that could be used for initializing an evolving collaboration network model, like an agent based model. In business it is important to initialise properly (Grimm et al., 2020) and the focus is often on value creation for companies (Raesfeld et al., 2012; Oukes, 2018). In order to have models that could verify these value creation processes, a hypergraph model would be a good start.

Limitations. This thesis also has certain limitations. First, the data could have been improved, especially in relation with the project role node types. Project role 4 arose purely from a lack of information. Furthermore, some companies also had multiple project roles in different projects. The analysis could be improved by dividing each node into multiple nodes, representing the different departments of a company. Second, even though the measures for hypergraphs presented in this thesis were quite extensive, it could be more extended. General measures were used: vertex degree, dimension, hyperedge degree, eigenvector, betweenness and closeness centrality, modularity and homophily. Betweenness and closeness centrality were not used to compare the models with each other. Finally, the SHHCM and NHHCM model and the effect of these models could have been investigated better. We found an interesting result of the SHHCM and NHHCM model that

the NHHCM model corrects the networks resulting from the SHHCM model related to heterogeneity.

Future Research. In future research, the effect of the the models on the betweeness and closeness could be tested. Furthermore, it is preferred to find measures that capture the complete structure of a hypergraph. In this thesis, the hyperedge degree was the only measure based on hypergraph, specifically. Still, this measure was based on a specific vertex and not on the complete network. Second, in the contributions we stated that some concepts or ideas in this thesis are not mathematically proven, but made up based on general ideas. In future research, the mathematical proofs and or more specific research about the correctness of the formulations of this thesis could be established. Third, the effect of a dominant node type or the effect of a barely available node type could be researched. Furthermore, the exact effect of the reshuffling method could be analysed. It could also be chosen to find or create other models that removes selfloops in heterogeneous hypergraphs. In addition, in future research, the models could be adapted such that other cases could be tested. Based on the results in Section 4.2, it would be interesting to research the effect of constraining the network by the average dimension per vertex. This variable would prevent that a project partner that normally only takes part in small group collaboration, from being assigned to a project with a large dimension. Furthermore, testing the model when changing the formula as presented in Equation 15 would also be interesting. Changing this equation into a multiplicative equation might be useful. Moreover, in future research the connection to the performance of the projects based on hypergraph models could be made. The performance of the projects of the collaboration network is also presented in this thesis in Figure 8. This future research makes use of weighted hypergraphs, where the weight represents the performance of the project. Using the results of this thesis as a basis for studying evolving collaboration networks may be especially useful in business. Finally, in this thesis we saw that normal graphs fail to reproduce the dimension (or project size) distribution of a collaboration network. In normal graphs, finding cliques is a difficult process. Therefore, hypergraph models are superior over normal graph models, since, by definition, the dimension distribution is incorporated in hypergraph models. In future research, the superiority of hypergraphs can be proved even better.

6 Conclusion

This thesis is set out to answer the question of how mathematical heterogeneous hypergraph models could be successfully applied in the field of collaboration networks in business. We compared three heterogeneous hypergraph models. Furthermore, the use of hypergraph models and the correctness of the heterogeneity elements in the hypergraph model are tested. Our findings are as follows. First, we found that the NHHCM model is the best model when comparing the SHHCM, NHHCM and CLHHM, in the case that there is not a dominant type available. Otherwise the SHHCM model performs best. In addition, we found that a normal graph is not a good representation of a collaboration network. It replicates some measures of the collaboration network, but a hypergraph does this better. Moreover, it could not be stated whether hypergraph models are better models than normal graph models. We also found that the heterogeneous hypergraph models significantly improve the original SHHCM, NHHCM and CLHHM, even though the heterogeneous hypergraph models are based on equations that are only based on an intuition of heterogeneous graphs and are not mathematically proven.

References

Alaluusua, K., Avrachenkov, K., Kumar, B. V., & Leskelä, L. (2023). Multilayer hypergraph clustering using the aggregate similarity matrix. In *International workshop on algorithms and models for the web-graph* (pp. 83–98). Cham: Springer Nature Switzerland.

Albert, R., & Barabási, A. L. (2002). Statistical mechanics of complex networks. *Reviews of Modern Physics*, *74*(1), 47–97. doi: 10.1103/RevModPhys.74.47

Avin, C., Lotker, Z., Nahum, Y., & Peleg, D. (2019, aug). Random preferential attachment hypergraph. In *Proceedings of the 2019 ieee/acm international conference on advances in social networks analysis and mining, asonam 2019* (pp. 398–405). Association for Computing Machinery, Inc.

Battiston, F., Amico, E., Barrat, A., Bianconi, G., Ferraz de Arruda, G., Franceschiello, B., ... others (2021). The physics of higher-order interactions in complex systems. *Nature Physics*, *17*(10), 1093–1098.

Battiston, F., Cencetti, G., Iacopini, I., Latora, V., Lucas, M., Patania, A., ... Petri, G. (2020). Networks beyond pairwise interactions: Structure and dynamics. *Physics Reports*, *874*, 1–92.

Benson, A. R. (2019). Three Hypergraph Eigenvector Centralities. *SIAM Journal on Mathematics of Data Science*, *1*(2), 293–312.

Benson, A. R., Gleich, D. F., & Leskovec, J. (2016). Higher-order organization of complex networks. *Network Science*, *353*(6295), 163–166.

Bhagat, S., Cormode, G., & Muthukrishnan, S. (2011). *Node Classification in Social Networks*. Boston, MA: Springer US.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics*, 1-12.

Bomze, I. M., Budinich, M., Pardalos, P. M., & Pelillo, M. (1999). The maximum clique problem. *Handbook of Combinatorial Optimization: Supplement Volume A*, 1–74.

Bonacich, P. (2007). Some unique properties of eigenvector centrality. *Social networks*, *29*(4), 555–564.

Bonacich, P., Cody Holdren, A., & Johnston, M. (2004). Hyper-edges and multidimensional centrality. *Social Networks*, *26*, 189–203.

Brandes, U. (2001). A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, *25*(2), 163–177.

Busatto, G., & Hoffmann, B. (2001). Comparing notions of hierarchical graph transformation. *Electronic Notes in Theoretical Computer Science*, *50*(3), 310–317.

Casiraghi, G. (2019, dec). The block-constrained configuration model. *Applied Network Science*, *4*(1), 123.

Chodrow, P. S. (2020). Configuration models of random hypergraphs. *Journal of Complex Networks*, *8*(3), 1–26.

Chodrow, P. S., Eikmeier, N., & Haddock, J. (2023). Nonbacktracking spectral clustering of nonuniform hypergraphs. *SIAM Journal on Mathematics of Data Science*, *5*(2), 251–279.

Chung, F., & Lu, L. (2002). Connected Components in Random Graphs with Given Expected Degree Sequences. *Annals of Combinatorics*, *6*, 125–145.

Currarini, S., Jackson, M. O., & Pin, P. (2009). An Economic Model of Friendship: Homophily, Minorities, and Segregation. *Econometrica*, *77*(4), 1003–1045.

Dong, X., Frossard, P., Member, S., Vandergheynst, P., & Nefedov, N. (2012). Clustering With Multi-Layer Graphs : A Spectral Perspective. *IEEE Transactions on Signal*

Processing, *60*(11), 5820–5831.

Dong, X., Frossard, P., Member, S., Vandergheynst, P., & Nefedov, N. (2014). Clustering on Multi-Layer Graphs via Subspace Analysis on Grassmann Manifolds. *IEEE Transactions on signal processing*, *62*(4), 905–918.

Drewes, F., Hoffmann, B., & Plump, D. (2002). Hierarchical graph transformation. *Journal of Computer and System Sciences*, *64*(2), 249–283.

Eikmeier, N., Ramani, A. S., & Gleich, D. (2018, nov). The HyperKron Graph Model for Higher-Order Features. In *2018 ieee international conference on data mining (icdm)* (pp. 941–946).

Failla, A., Citraro, S., & Rossetti, G. (2023). Attributed stream hypergraphs: temporal modeling of node-attributed high-order interactions. *Applied Network Science*, *8*(1), 1–19.

Gao, J., Zhao, Q., Ren, W., Swami, A., Ramanathan, R., & Bar-Noy, A. (2015). Dynamic Shortest Path Algorithms for Hypergraphs. *IEEE/ACM Transactions on Networking*, *23*(6), 1805–1817.

Ghosh, R., & Lerman, K. (2019). Structure of Heterogeneous Networks. *International Conference on Computational Science and Engineering*, *4*(1), 98–105.

Ghoshal, G., Zlatić, V., Caldarelli, G., & Newman, M. E. J. (2009, jun). Random hypergraphs and their applications. *Physical Review E*, *79*(6), 1–11.

Giroire, F., Nisse, N., Ohulchanskyi, K., Sulkowska, M., & Trolliet, T. (2022, apr). *Preferential attachment hypergraph with vertex deactivation* (Tech. Rep.). Cornell University.

Giroire, F., Nisse, N., Trolliet, T., & Sulkowska, M. (2021, mar). *Preferential attachment hypergraph with high modularity* (Tech. Rep.). Cornell University.

Grimm, V., Railsback, S. F., Vincenot, C. E., Berger, U., Gallagher, C., Deangelis, D. L., ... Ayllón, D. (2020). The ODD protocol for describing agent-based and other simulation models: A second update to improve clarity, replication, and structural realism. *Journal of Artificial Societies and Social Simulation*, *23*(2).

Kamiński, B., Poulin, V., Prałat, P., Szufel, P., & Théberge, F. (2019). Clustering via hypergraph modularity. *PLOS ONE*, *14*(11), 1–15.

Kim, J., & Lee, J.-g. (2015). Community Detection in Multi-Layer Graphs : A Survey. *IEEE Transactions on signal processing*, *44*(3).

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, *22*(1), 79–86.

Kumar, T., Vaidyanathan, S., Ananthapadmanabhan, H., Parthasarathy, S., & Ravindran, B. (2020). A New Measure of Modularity in Hypergraphs: Theoretical Insights and Implications for Effective Clustering. *Studies in Computational Intelligence*, *881*, 286–297.

Kuznetsov, S. O., Panov, A. I., & Yakovlev, K. S. (2020). Artificial Intelligence. In S. O. Kuznetsov, A. I. Panov, & K. S. Yakovlev (Eds.), *Artificial intelligence* (pp. 1–485). Cham: Springer International Publishing.

Landry, N. W., & Restrepo, J. G. (2020). The effect of heterogeneity on hypergraph contagion models. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *30*(10), 1–17.

Lee, J., Lee, Y., Oh, S. M., & Kahng, B. (2021). Betweenness centrality of teams in social networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *31*(6).

Mäkinen, E. (1990, jan). How to draw a hypergraph. *International Journal of Computer Mathematics*, *34*(3-4), 177–185.

Mcpherson, M., Smith-lovin, L., & Cook, J. M. (2001). Birds of a Feather : Homophily in

Social Networks Author(s). , *27*(2001), 415–444.
Newman, M. (2010). *Networks: An Introduction*. Oxford university press.
Newman, M. E. (2003). Mixing patterns in networks. *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, *67*(2), 13.
Newman, M. E. (2005). A measure of betweenness centrality based on random walks. *Social networks*, *27*(1), 39–54.
Newman, M. E. (2006a). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, *103*(23), 8577–8582.
Newman, M. E. (2006b). Finding community structure in networks using the eigenvectors of matrices. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, *74*(3), 1–19.
Oukes, T. (2018). *Network Position and Related Power*. Enschede: Ipskamp Printing.
Pan, W., Dong, W., Cebrian, M., Kim, T., Fowler, J. H., & Pentland, A. S. (2012, mar). Modeling Dynamical Influence in Human Interaction: Using data to make better inferences about influence within social systems. *IEEE Signal Processing Magazine*, *29*(2), 77–86.
Papanikolaou, N., Lambiotte, R., & Vaccario, G. (2023). *Fragmentation from group interactions: A higher-order adaptive voter model* (Tech. Rep.). Cornell University.
Raesfeld, A. V., Geurts, P., Jansen, M., Boshuizen, J., & Luttge, R. (2012). Influence of partner diversity on collaborative public R&D project outcomes: A study of application and commercialization of nanotechnologies in the Netherlands. *Technovation*, *32*(3-4), 227–233.
Ruhnau, B. (2000). Eigenvector-centrality—a node-centrality? *Social networks*, *22*(4), 357–365.
Saracco, F., Petri, G., Lambiotte, R., & Squartini, T. (2022). *Entropy-based random models for hypergraphs* (Tech. Rep.). Cornell University.
Storm, C. K. (2006). *The zeta function of a hypergraph* (Tech. Rep.).
STW. (2014). *Utilisatierapport 2014* (Tech. Rep.).
Tudisco, F., & Higham, D. J. (2021). Node and edge nonlinear eigenvector centrality for hypergraphs. *Communications Physics*, *4*(1).
Van der Hofstad, R. (2016). *Random Graphs and Complex Networks* (Vol. 43). Cambridge university press.
Van der Hofstad, R., Van Leeuwaarden, J. S., & Stegehuis, C. (2017). Hierarchical Configuration Model. *Internet Mathematics*, 1–25.
Veldt, N., Benson, A. R., & Kleinberg, J. (2023). Combinatorial characterizations and impossibilities for higher-order homophily. *Science Advances*, *9*(1), 1–54.
Von Wrycza, P., Dahlman, E., Larsson, E., Parkvall, S., Sköld, J., & Chapman, T. (2015). *CHAPTER HSPA Evolution: The Fundamentals for Mobile Broadband*. Academic Press.
Wang, C., Yuan, M., Zhang, R., Peng, K., & Liu, L. (2023). Efficient Point-of-Interest Recommendation Services With Heterogenous Hypergraph Embedding. *IEEE Transactions on Services Computing*, *16*(2), 1132–1143.
Wang, J., Rong, L., Deng, Q., & Zhang, J. (2010). Evolving hypernetwork model. *European Physical Journal B*, *77*(4), 493–498.
Zhang, C., Song, D., Huang, C., Swami, A., & Chawla, N. V. (2019). Heterogeneous Graph Neural Network. In *Proceedings of the 25th acm sigkdd international conference on knowledge discovery & data mining* (pp. 793–803). New York, NY, USA: ACM.
Zhang, J., & Luo, Y. (2017). Degree centrality, betweenness centrality, and closeness

centrality in social network. In *2017 2nd international conference on modelling, simulation and applied mathematics (msam2017)* (pp. 300–303).

A Table with variables and abbreviations

Var./Abbrev.	**Description**
A	adjacency matrix
a_i	average number of friendship of node type i with same node types
a_{ij}	average number of friendship of node type i with node type j
$\tilde{a}_{ij}$	average number of friendship of node type i with node type j in hypergraphs
AES	applied and engineering sciences
ASH	attributed stream hypergraph
B	incidence matrix
b	reshuffling function hyperedges
b_i	average number of friendship of node type i with other node types
BC	betweenness centrality
CC	closeness centrality
CI	confidence interval
CM	configuration model
CLHM	Chung-Lu hypergraph model
CLHHM	Chung-Lu heterogeneous hypergraph model
D	diagonal vertex degree matrix
d	degree vector/sequence
D_e	hyperedge degree
	diagonal matrix with edge dimensions
d_e	dimension of edge e
d_v	vertex degree of vertex v
d_{norm}	degree vector/sequence of normalized hypergraph
E	edge set
e	(hyper)edge
E_t	edge set at time t
	hyperedge with dimension t
EC	eigenvector centrality
F_k	mutiset of edges with dimension k
G	graph
GRG	generalized random graph
H	hypergraph
	incidence matrix of a hypergraph
h	sample interval
H_0	initial hypergraph in algorithm
H_i	homophily index of type i node
H_t	hypergraph at time t
HE	hyperedge
HSBM	hypergraph stochastic block model
I	identity matrix
k	dimension vector/sequence
k_{avg}	average dimension sequence
$k_{avg,i}$	average dimension of a vertex v_i
k_i	dimension of edge i

k_{norm}	dimension vector/sequence of normalized hypergraph
KL-div.	Kullbach-Leibner divergence
KPI	key performance indicator
M_i	multiplicity of vertex v_i
m	number of edges
m_Δ	number of parallel hyperedges Δ
MSE	mean squared error
n	number of vertices
NHCM	no-selfloop hypergraph configuration model
NHHCM	no-selfloop heterogeneous hypergraph configuration model
NWO	Nederlandse origanisatie voor wetenschappelijke onderzoek
P	probability function
PA model	preferential attachment model
PR	project role
p_{ij}	probability of edge between vertex i and j
Q	modularity measure
R	uniformly picked stubs
	existing collaboration
R&D	research and development
S	hypergraph in algorithm
s	sample size
SBM	stochastic block model
std	standard deviation
STW	stichting voor technische wetenschappen
SHCM	stub-labeled hypergraph configuration model
SHHCM	stub-labeled heterogeneous hypergraph configuration model
T	Hypergraph adjacency tensor
	vector with node types
t	time
T_v	type of node v
u	vertex
UICN	university-industry collaboration network
V	vertex set
v	vertex
V_i	vertex set of type i
v_i	vertex i
VC	value chain
W	diagonal weight matrix
w	vector with edge-weights
w_i	relative fraction of node type i
x	eigenvector
y	eigenvector
α	acceptance rate
β	parameter beta-centrality
Γ	hyperedge
Δ	hyperedge
λ	eigenvalue

Σ	multiset with stubs
Σ	multiset with stubs of type i
τ	exponential for power law
Ω	block-matrix with preferences between types
Ω_{ij}	probability vertex type i connects with type j

TABLE 20: Table with variables (var.) and abbreviations (abbrev.)

B Hyperedge measures for heterogeneous and non-heterogeneous hypergraphs

T	Meas.	H/N	Original	SHHCM	NHHCM	CLHHM
PR	max	H	242	(248.05, 257.75)	(273.89, 283.90)	(254.81, 265.19)
		N		(255.79, 265.01)	(270.60, 278.56)	(254.63, 264.33)
	min	H	1	(0.040, 0.240)	(-0.016, 0.096)	(0.762, 1.198)*
		N		(0.027, 0.213)	(0.048, 0.352)	(0.987, 1.573)*
	mean	H	93.93	(86.99, 88.10)	(101.08, 101.51)	(90.59, 92.96)
		N		(86.97, 88.09)	(101.78, 102.20)	(90.82, 93.29)
	std	H	61.85	(54.91, 55.78)	(61.87, 62.42)	(54.42, 55.77)
		N		(55.01, 55.85)	(60.84, 61.44)	(54.71, 56.03)
	KL-div	H	0	237.31	200.56	234.75
		N		238.10	206.42	228.75
VC	max	H	242	(250.28, 259.60)	(264.54, 272.38)	(254.75, 264.77)
		N		(233.14, 241.18)	(273.65, 282.51)	(259.03, 269.93)
	min	H	1	(0.014, 0.186)	(0.093, 0.387)	(0.866, 1.294)*
		N		(-0.004, 0.244)	(0.055, 0.265)	(0.831, 1.329)*
	mean	H	93.93	(87.35, 88.55)	(100.94, 101.41)	(90.88, 93.16)
		N		(79.87, 80.82)	(101.59, 101.97)	(91.29, 93.97)*
	std	H	61.85	(54.60, 55.45)	(60.76, 61.44)	(54.92, 56.25)
		N		(50.44, 51.18)	(61.03, 61.63)	(55.13, 56.59)
	KL-div	H	0	239.85	209.14	228.28
		N		262.46	204.12	230.27

TABLE 21: CIs for hyperedge degree heterogeneous (H) and non-heterogeneous (N) hypergraphs.
N = 50, $\alpha = 0.05$; two-tailed. *Original value is in CI.

C Code for Analysis

C.1 Code for Descriptive Analysis

```
% % Desciptive Analysis of R&D Collaboration Data

load('project_info_adj.mat')            % Info per project
load('project_users_adj.mat')           % Info per user
load('project_user_data_adj.mat')       % Combination project
   + users

% List from file.mat: [file{row,column}]

close all

% number of projects

project_number1 = length(unique([project_info_adj{:,1}]));
project_number2 = length(unique([project_user_data_adj{:,1}]))
   ;

% total number of partners

connections1 = sum([project_users_adj{:,78}]);
connections2 = length([project_user_data_adj{:,1}]);

% unique partners

unique_partners = length([project_users_adj{:,1}]);

% distribution of project sizes

[project_size,project_numbers] = groupcounts([
   project_user_data_adj{:,1}]);

[freq_projsize, unique_projsize] = groupcounts(project_size);

% project roles

[freq_roles, roles] = groupcounts([project_user_data_adj{:,
   13}]);
projsize_perpartner = repelem(project_size, project_size);
projroles = [project_user_data_adj{:, 13}];

splitroles = zeros(4,22);

for i = 1:length(projroles)
    if projroles(i) == 1
        splitroles(1,projsize_perpartner(i)) = ...
            splitroles(1,projsize_perpartner(i)) + 1;
```

```
    elseif projroles(i) == 2
        splitroles(2,projsize_perpartner(i)) = ...
            splitroles(2,projsize_perpartner(i)) + 1;
    elseif projroles(i) == 3
        splitroles(3,projsize_perpartner(i)) = ...
            splitroles(3,projsize_perpartner(i)) + 1;
    else
        splitroles(4,projsize_perpartner(i)) = ...
            splitroles(4,projsize_perpartner(i)) + 1;
    end
end

splitroles1 = splitroles;
splitroles = splitroles./sum(splitroles);
splitroles(isnan(splitroles)) = 0;

bar1 = splitroles(1,:);
bar2 = splitroles(2,:);
bar3 = splitroles(3,:);
bar4 = splitroles(4,:);

bar23 = bar2 + bar3;
bar231 = bar23 + bar1;
bar2314 = bar231 + bar4;

% changing project roles of one company

company_projrol = zeros(unique_partners, 4);
projroles ; % roles companies of all projects
projcomp = [project_user_data_adj{:,2}];
[unique_projcomp, ~, num_projcomp] = unique(projcomp);
projroles(isnan(projroles)) = 4;
projroles(projroles == 0) = 4;
for i = 1:length(projroles)
    company_projrol(num_projcomp(i),projroles(i)) = ...
    company_projrol(num_projcomp(i),projroles(i)) + 1;
end

doubleroles = 0;
for i = 1:unique_partners
    if sum(length(find(company_projrol(i,:)))) == 2
        doubleroles = doubleroles + 1;
    elseif sum(length(find(company_projrol(i,:)))) == 3
        doubleroles = doubleroles + 1;
    elseif sum(length(find(company_projrol(i,:)))) == 4
        doubleroles = doubleroles + 1;
    end
end

```

```
92
93  % distribution of technology field number
94
95  techfield = [project_info_adj{:,32}];
96  [freq_techfield, unique_techfield] = groupcounts(techfield);
97
98  % distribution of application number
99
100 application = [project_info_adj{:,42}];
101 [freq_application, unique_application] = groupcounts(
        application);
102
103 % involvement, product development, revenues generated
104
105 involvement = [project_info_adj{:,8}];
106 productdev = [project_info_adj{:,9}];
107 revenues = [project_info_adj{:,10}];
108
109 [unique_involvement, ~,num_involvement] = unique(involvement);
110 [unique_productdev, ~,num_productdev] = unique(productdev);
111 [unique_revenues, ~,num_revenues] = unique(revenues);
112
113 [freq_involvement, ~] = groupcounts(num_involvement);
114 [freq_productdev, ~] = groupcounts(num_productdev);
115 [freq_revenues, ~] = groupcounts(num_revenues);
116 cat1 = categorical({'A'});
117 cat2 = categorical({'B'});
118 cat3 = categorical({'C'});
119 cat4 = categorical({'n.a.'});
120
121 % budget and external financing
122
123 budget = [project_info_adj{:,16}];
124 extfinance = [project_info_adj{:,17}];
125 extfinance(isnan(extfinance)) = 0;
126
127 [sort_budget, index_budget] = sort(budget, 'descend');
128 sort_extfinance = extfinance(index_budget);
129 sort_budget = [sort_budget; 0];
130 sort_extfinance = [sort_extfinance; 0];
131
132 % % Descriptives Participants
133 % Operating income
134
135 SZ_OI = [project_users_adj{:,33}];
136 [unique_OI, ~,num_OI] = unique(SZ_OI);
137 unique_OI = [unique_OI(1:7); 'n.a.'];
138 num_OI(num_OI>7) = 8;
139 perm_num_OI = [7 5 4 2 3 1 6 8];
```

```
permunique_OI = unique_OI(perm_num_OI);

[freq_OI, ~] = groupcounts(num_OI);
permfreq_OI = freq_OI(perm_num_OI);

x = categorical(permunique_OI);
x = reordercats(x,string(x));
```

C.2 Code for Network Analysis

```
% % Network Analysis of R&D Collaboration Data
%
% List from file.mat: [file{row,column}]

% % Input variables
% V = set of vertices per hyperedge.
% E = set with the numbers of the hyperedges that correspond to the
% vertices.
% G = graph that is build based on V and E.
% n = number of vertices
% m = number of hyperedges
% T = type vector with types for each node

function [Measures] = network_analysis(E,V,G,n,m,T)

% % Degree distribution: vertex degree, dimension, hyperedge degree
% % Vertex degree

Measures = dictionary(1:13,{[]});

D = degree(G,m+1:m+n);
[freq_degree, unique_degree] = groupcounts(D');

Measures{1} = D;
Measures{2} = freq_degree;
Measures{3} = unique_degree;

% % Dimension

[project_size,project_numbers] = groupcounts(E);
[freq_projsize, unique_projsize] = groupcounts(project_size);

Measures{4} = freq_projsize;
Measures{5} = unique_projsize;

% % Hyperedge degree

```

```
HE = zeros(max(E),1);
for i = 1:max(E)
    N = neighbors(G,i);
    A = [];
    for j = N'
        A = [A; neighbors(G,j)];
    end
    HE(i) = length(unique(A)) - 1;
end

[freq_HE, unique_HE] = groupcounts(HE);

Measures{6} = freq_HE;
Measures{7} = unique_HE;

% Centrality: betweenness, closeness and eigenvector

d = distances(G,min(num_company_nodes):max(num_company_nodes),
    ...
    min(num_company_nodes):max(num_company_nodes));

Sigma = cell(n, n);

for i = 1:n
    Sigma{i,i} = {i + min(num_company_nodes) - 1};
    for j = 1:i-1
        if isinf(d(i,j))
            Sigma{i,j} = {NaN};
            Sigma{j,i} = {NaN};
        else
            Sigma{i,j} = allpaths(G,i + min(num_company_nodes)
                - 1, ...
                j + min(num_company_nodes) - 1,'MaxPathLength'
                    ,d(i,j));
            Sigma{j,i} = Sigma{i,j};
        end
        disp([num2str(i),' and ', num2str(j)])
    end
end

d = d/2;

% % Closeness centrality
CC = NaN(max(num_company_nodes) - min(num_company_nodes) + 1,
   1);
Inf_shortpath = NaN(max(num_company_nodes) - min(
   num_company_nodes) + 1, 1);
for i = 1: max(num_company_nodes) - min(num_company_nodes) + 1
    Inf_shortpath(i) = sum(d(i,:) == Inf);
```

```
82      d(i,d(i,:) == Inf) = 0;
83      CC(i) = 1/sum(d(i,:));
84  end
85  CC(CC == Inf) = 0;
86
87  % % Betweenness Centrality
88
89  Sigma_v = zeros(n, n, n);
90  Sigma_st = zeros(n,n);
91
92  for i = 1:n % start point shortest path
93      Sigma_st(i,i) = 1;
94      Sigma_v(i,i,i) = 1;
95      for j = i+1:n % end point shortest path
96          Sigma_st(i,j) = length(Sigma{i,j});
97          Sigma_st(j,i) = length(Sigma{j,i});
98          for k = 1:n % v in shortest path
99              for l = 1:length(Sigma{i,j})
100                 if iscell(Sigma{i,j}) && any(Sigma{i,j}{l} ==
                        k+406)
101                     Sigma_v(i,j,k) = Sigma_v(i,j,k) + 1;
102                     Sigma_v(j,i,k) = Sigma_v(j,i,k) + 1;
103                 end
104             end
105         end
106     end
107     disp([num2str(i),' and ', num2str(j), ' and ', num2str(k)
            ])
108 end
109
110 BC = NaN(n, 1);
111
112 for v = 1:n
113     BC(v) = sum(sum(Sigma_v([1:v-1 v+1:783], [1:v-1 v+1:783],
            v)))/...
114     sum(sum(Sigma_st([1:v-1 v+1:783], [1:v-1 v+1:783])));
115 end
116
117 % % Vector Centrality
118 H = zeros(m,n);
119
120 for i = 1:m
121     H(i, V(E == i) - m) = 1;
122 end
123
124 [eigenvector1, lambda1] = eigs(H*H', 1);
125 [eigenvector2, lambda2] = eigs(H'*H,1);
126
127 Measures{8} = H;
```

```
128 Measures{9} = eigenvector1;
129 Measures{10} = eigenvector2;
130
131 % % Modularity
132
133 De = diag(project_size);
134 I = eye(m);
135 d = D';
136
137 DD = (De-I)^-1;
138 DD2 = H'*DD;
139
140 A = H'*DD*H;
141 P = d*d'/sum(d);
142 P(1:n+1:end)=0;
143 A(1:n+1:end)=0;
144 dkron = zeros(n,n);
145
146 for i = 1:n
147     for j = 1:n
148         if T(i) == T(j)
149             dkron(i,j) = 1;
150         end
151     end
152 end
153
154 Q = 1/(2*m) * sum(sum((A-P).*dkron));
155 Measures{11} = Q;
156
157 % Assortative mixing (homophily)
158
159 xx = groupcounts(E);
160 index1 = 0;
161 a = zeros(max(T),max(T));
162
163 for i = 1:length(xx)
164     types = T(V(index1 + 1:index1 + xx(i))- m);
165     freq = groupcounts(types, 1:max(T)+1, "IncludeEmptyGroups
        ",true);
166     for ii = 1:max(T)
167         a(ii,ii) = a(ii,ii) + (2/(xx(i)*(xx(i)-1)))*(freq(ii)
            -1)*freq(ii);
168         for jj = 1:ii-1
169             a(ii,jj) = a(ii,jj) + (2/(xx(i)*(xx(i)-1)))*freq(
                ii)*freq(jj);
170             a(jj,ii) = a(jj,ii) + (2/(xx(i)*(xx(i)-1)))*freq(
                ii)*freq(jj);
171         end
172     end
```

```
173     index1 = index1 + xx(i);
174 end
175 a = 1./groupcounts(T).*a;
176 
177 H_index = diag(a)./sum(a,2);
178 w = groupcounts(T)/sum(groupcounts(T));
179 
180 Measures{12} = [H_index w];
181 
182 
183 % % Omega
184 
185 a_norm = a ./ sum(a,2);
186 Measures{13} = a_norm;
187 
188 end
```

D Code for Models

D.1 Chodrow I Model

```
% % Chodrow I: Heterogeneous hypergraph stub-matching
function [S] = Chodrow1(d,k,T,Omega)
    % Degree and Dimension Check
    if sum(d) ~= sum(k)
        error('NOTE: Degrees and dimensions are not congruent')
    end

    TT = max(T);

    % Initialization
    S = dictionary(1:length(k),{[]});
    Sigma = [repelem(1:length(d),d); zeros(1,sum(d))];

    index1 = 1;

    for i = 1:length(d)
        Sigma(2,index1:index1+d(i)-1) = 1:d(i);
        index1 = index1 + d(i);
    end

    for j = 1:length(k)
        Rnum = datasample(1:length(Sigma(1,:)),1);
        R = Sigma(:,Rnum);
        Sigma(:,Rnum) = [];
        if ~ismember(T(R(1)),T(Sigma(1,:)))
            Omega(:,T(R(1))) = 0;
            Omega = Omega./sum(Omega,2);
        end

        for i = 1:k(j)-1
            Pt = zeros(max(T),1);
            for ii = 1:TT
                Pt(ii) = sum(Omega(T(R(1,:)), ii))/length(R
                    (1,:));
            end

            c = histc(T(Sigma(1,:)),1:max(T));

            PP = Pt(T(Sigma(1,:)));
            P = 1./c(T(Sigma(1,:))).*PP';

            K = randsample(1:length(Sigma(1,:)), 1, true, P);
            R = [R Sigma(:,K)];
            Sigma(:,K) = [];
            if ~ismember(T(R(end)),T(Sigma(1,:)))
```

```
                Omega(:,T(R(end))) = 0;
                Omega = Omega./sum(Omega,2);
            end
        end

        S{j} = R;
    end
end
```

D.2 Chodrow II Model

```
% % Chodrow II: Markov Chain Monte Carlo for hypergraph
   configuration models
function[Ht] = Chodrow2(d,k,T,Omega,h,s)
    S = Chodrow1(d,k,T,Omega);
    Ht = dictionary(0:s*h,{[]});
    Ht{0} = S;

    for t = 1:s*h
        S = Ht{t-1};
        R = randsample(numEntries(S), 2);
        Delta = S{R(1)};
        Gamma = S{R(2)};
        Shat = S;
        [Shat{R}] = deal([],[]);
        Deltahat = [];
        Gammahat = [];
        Gammanew = Gamma;
        Deltanew = Delta;
        for i = length(Delta(1,:)):-1:1
            if ismember(Delta(1,i),Gammanew(1,:))
                [~, P1] = ismember(Delta(1,i),Gammanew(1,:));
                Deltahat = [Deltahat Delta(:,i)];
                Deltanew(:,i) = [];
                Gammahat = [Gammahat Gammanew(:,P1)];
                Gammanew(:,P1) = [];
            end
        end
        Rem_Stubs = [Deltanew Gammanew];
        for i = 1:length(Deltanew(1,:))
            Pt = zeros(1,max(T));
            for j = 1:max(T)
                Pt(j) = sum(Omega(T(Delta(1,:)),j))/length(
                   Delta(1,:));
            end
            P = Pt(T(Rem_Stubs(1,:)));
            P = P/sum(P);
            K = randsample(1:length(Rem_Stubs(1,:)), 1, true,
               P);
```

```
            Deltahat = [Deltahat Rem_Stubs(:,K)];
            Rem_Stubs(:,K) = [];
        end
        Gammahat = [Gammahat Rem_Stubs];
        Shat{R(1)} = Deltahat;
        Shat{R(2)} = Gammahat;
        mdelta = 0;
        mgamma = 0;
        for i = 1:numEntries(S)
            if isequal(sort(Delta(1,:)),sort(S{i}(1,:)))
                mdelta = mdelta + 1;
            end
            if isequal(sort(Gamma(1,:)),sort(S{i}(1,:)))
                mgamma = mgamma + 1;
            end
        end
        alpha = (2^(length(Delta)-length(Deltanew)))/(mdelta*
           mgamma);

        if rand(1) <= alpha
            Ht{t} = Shat;
        else
            Ht{t} = Ht{t-1};
        end
    end
end
```

D.3 Chung-Lu Model

```
% % Chung-Lu Model
function [S] = ChungLu(d,k,T,Omega)
    p = d / sum(d);
    S = dictionary(1:length(k),{[]});

    for i = 1:length(k)
        S{i} = [find(mnrnd(1,p)==1)];
    end

    Undef_Stubs = k - 1;
    for i = 1:sum(k)-length(k)
        stub = mnrnd(1,p);
        p_het = zeros(1,length(k));
        for j = find(Undef_Stubs > 0)
            p_het(j) = sum(Omega(T(S{j}),T(stub==1)))/length(S
               {j});
        end
        p_het = p_het/sum(p_het);
        if length(find(Undef_Stubs>0)) > 1
```

```
19              R = randsample(find(Undef_Stubs > 0),1,true,p_het(
                    p_het>0));
20          else
21              R = find(Undef_Stubs > 0);
22          end
23          S{R} = [S{R} find(stub == 1)];
24          Undef_Stubs(R) = Undef_Stubs(R)-1;
25      end
26  end
```

E Code for Execution

E.1 Code Main Execution File

```
% % Visualisation of the network
load('project_info_adj.mat')            % Info per project
load('project_users_adj.mat')           % Info per user
load('project_user_data_adj.mat')       % Info project + users

project_nodes = [project_user_data_adj{:,1}];
company_nodes = [project_user_data_adj{:,2}];

[unique_project_nodes, ~, num_project_nodes] = unique(
    project_nodes);
[unique_company_nodes, ~, num_company_nodes] = unique(
    company_nodes);
num_company_nodes = num_company_nodes + max(num_project_nodes)
    ;

G0 = graph(num_project_nodes,num_company_nodes); % Original
    graph

load('T_projrole.mat')
load('T_VC.mat')

% Input variables

d = degree(G0, min(num_company_nodes):max(num_company_nodes));
k = groupcounts(num_project_nodes)';
T = T_VC';

n = length(d);
m = length(k);
M0 = network_analysis(num_project_nodes,num_company_nodes,G0,n
    ,m,T');

Omega = M0{13};
Omega(4,4) = 0.000001;
Omega(4,3) = Omega(4,3) - 0.000001;
h = 125;
s = 40;
runs = 50;

M1_VC = dictionary(1:runs,{[]});
M2_VC = dictionary(1:runs,{[]});
M3_VC = dictionary(1:runs,{[]});
Graph1_VC = dictionary(1:runs,{[]});
Graph2_VC = dictionary(1:runs,{[]});
Graph3_VC = dictionary(1:runs,{[]});

```

```
% Results

for r = 1:runs
    S1 = Chodrow1(d,k,T,Omega);
    S2 = Chodrow2(d,k,T,Omega,h,s);
    S3 = ChungLu(d,k,T,Omega);

    V1 = zeros(1,sum(d));
    V2 = zeros(1,sum(d));
    V3 = zeros(1,sum(d));
    E1 = zeros(1,sum(d));
    E2 = zeros(1,sum(d));
    E3 = zeros(1,sum(d));
    index1 = 1;
    index2 = 1;
    index3 = 1;

    for i = 1:length(k)
        V1(index1:index1+length(S1{i}(1,:))-1)= S1{i}(1,:);
        V2(index2:index2+length(S2{h*s}{i}(1,:))-1)= S2{h*s}{i
           }(1,:);
        V3(index3:index3+length(S3{i}(1,:))-1)= S3{i}(1,:);
        E1(index1:index1+length(S1{i}(1,:))-1) = i;
        E2(index2:index2+length(S2{h*s}{i}(1,:))-1) = i;
        E3(index3:index3+length(S3{i}(1,:))-1) = i;
        index1 = index1+length(S1{i}(1,:));
        index2 = index2+length(S2{h*s}{i}(1,:));
        index3 = index3+length(S3{i}(1,:));
    end

    V1 = V1 + max(E1);
    V2 = V2 + max(E2);
    V3 = V3 + max(E3);

    G1 = graph(V1, E1);
    G2 = graph(V2, E2);
    G3 = graph(V3, E3);

    Graph1_VC{r} = {V1, E1};
    Graph2_VC{r} = {V2, E2};
    Graph3_VC{r} = {V3, E3};

    % Network Analysis

    if max(V3) < n+m
        G3 = addnode(G3, n+m - max(V3));
    end

    n = length(d); % Number of vertices
```

```
    m = length(k); % Number of edges

    Measures1 = network_analysis(E1',V1',G1,n,m,T');
    Measures2 = network_analysis(E2',V2',G2,n,m,T');
    Measures3 = network_analysis(E3',V3',G3,n,m,T');

    M1_VC{r} = Measures1;
    M2_VC{r} = Measures2;
    M3_VC{r} = Measures3;

    disp(['Run ', num2str(r)])
end
```

E.2 Code Function Normal Graphs

```
% FUNCTION TO CREATE A NORMAL GRAPH
%
% INPUT:
% k          := dimension sequence
% Vertex1    := the vertex vector of the graph

function [VN1, EN1, GN1, E_normunique, G_norm] = ...
    clique_normalgraph(k, Vertex1)

E1_norm = zeros(sum(k.*(k-1)),1);
E2_norm = zeros(sum(k.*(k-1)),1);

index1 = 1;
index2 = 1;

for i = 1:length(k)
    E1_norm(index1:index1 + k(i)^2 - 1) = ...
        repelem(Vertex1(index2:index2+k(i)-1), k(i));
    E2_norm(index1:index1 + k(i)^2 - 1) = ...
        repmat(Vertex1(index2:index2+k(i)-1), k(i), 1);

    index1 = index1 + k(i)^2;
    index2 = index2 + k(i);
end

E_norm = [E1_norm E2_norm];

for i = 1:length(E_norm(:,1))
    if E_norm(i,1) > E_norm(i,2)
        B = E_norm(i,2);
        E_norm(i,2) = E_norm(i,1);
        E_norm(i,1) = B;
    end
end
```

```
35
36 E_normunique = unique(E_norm,"rows");
37 E_normunique(diff(E_normunique, [], 2) == 0,:) = [];
38 G_norm = graph(E_normunique(:,1) - length(k), ...
39     E_normunique(:,2) - length(k));
40 E_normunique2 = E_normunique';
41 VN1 = E_normunique2(:) - length(k) + (1/2)*length(E_normunique
      (:));
42 EN1 = repelem(1:length(E_normunique(:,1)),2)';
43
44 GN1 = graph(VN1,EN1);
45 end
```

E.3 Code Execution File for Normal Graphs

```
1  % % Execution file for normal graphs
2
3  % Loading files
4  load('Graph1.mat')
5  load('Graph2.mat')
6  load('Graph3.mat')
7  load('T_projrole.mat')
8  load('T_VC.mat')
9
10 % Variables
11 runs = length(keys(Graph1));
12 T = T_VC';
13
14 M1_VChypernorm = dictionary(1:runs,{[]});
15 M2_VChypernorm = dictionary(1:runs,{[]});
16 M3_VChypernorm = dictionary(1:runs,{[]});
17 Graph1_VChypernorm = dictionary(1:runs,{[]});
18 Graph2_VChypernorm = dictionary(1:runs,{[]});
19 Graph3_VChypernorm = dictionary(1:runs,{[]});
20
21 for i = 1:runs
22     % % Algorithm 1
23     k1 = groupcounts(Graph1{i}{2}');
24     V1 = Graph1{i}{1}';
25     [VN1, EN1, GN1, ~, ~] = clique_normalgraph(k1, V1);
26     Graph1_VChypernorm{i} = {VN1', EN1'};
27
28     n1 = length(T);
29     m1 = max(EN1);
30     Measures_norm1 = network_analysis(EN1,VN1,GN1,n1,m1,T');
31     M1_VChypernorm{i} = Measures_norm1;
32
33     % % Algorithm 2
34     k2 = groupcounts(Graph2{i}{2}');
```

```
    V2 = Graph2{i}{1}';
    [VN2, EN2, GN2, ~, ~] = clique_normalgraph(k2, V2);
    Graph2_VChypernorm{i} = {VN2', EN2'};

    n2 = length(T);
    m2 = max(EN2);
    Measures_norm2 = network_analysis(EN2,VN2,GN2,n2,m2,T');
    M2_VChypernorm{i} = Measures_norm2;

    % % Algorithm 3
    k3 = groupcounts(Graph3{i}{2}');
    V3 = Graph3{i}{1}';
    [VN3, EN3, GN3, ~, ~] = clique_normalgraph(k3, V3);
    Graph3_VChypernorm{i} = {VN3', EN3'};

    n3 = length(T);
    m3 = max(EN3);

    if max(VN3) < n3+m3
    GN3 = addnode(GN3, n3+m3 - max(VN3));
    end

    Measures_norm3 = network_analysis(EN3,VN3,GN3,n3,m3,T');
    M3_VChypernorm{i} = Measures_norm3;

    disp(['Run ', num2str(i)])
end
```