UNIVERSITÀ
DI TRENTO

Department of Information Engineering and Computer Science

Bachelor's Degree in
Computer Science

Final dissertation

Hypergraph Summarization

Node aggregation-based method addressed with motifs analysis

Supervisor
Alberto Montresor
Quintino Francesco Lotito

Student
Stefano Genetti

Academic year 2021/2022

Acknowledgements

I dedicate this work to my mother Alessandra and my father Roberto. Their affection has profoundly encouraged me in achieving my scholastic and academic results. I love you.

I dedicate this work to Elisabetta and Aurora. I wish you all the best.

I dedicate this work to my cousins Massimo, Lia, Alessandro, Anna, Serena, Francesco, Emanuele, Marco, Gabriele, Camilla, Giacomo, Isabella, Beatrice, Sara, Steven, Jessica, Danilo, Gianluca, my uncles and aunts Francesca, Luciano, Beniamino, Grazia, Enrico, Lucia, Mario, Nicoletta, Fabrizio, Luisa, Ferdinando, Patrizia, Vito, Sonia, Giorgio.

I thank professor Alberto Montresor for giving me the opportunity to contribute to this research project.

I thank Francesco Lotito for supporting me with the work.

I thank Luisa for her invaluable help.

I thank my University friends, Erica, Fabrizio, Francesco, Giovanni, Laurence, Matteo, Pietro, Simone, Vittoria, for their sympathy, their smiles, their important support and their precious advice.

Contents

Abstract

Complex networks are systems made up of a large number of units interconnected by non-trivial patterns of interactions. Multidisciplinary researchers use networks in order to study complex systems. A complex system is a system made by a large number of single units (individuals, components or agents) interacting in such a way that the behaviour of the system is not a simple combination of the behaviours of the single units. In particular, some collective behaviours emerge without the need for any central control. In order to understand the relevance of networks, it is sufficient to think for a moment about the fascinating fact that actually complex networks are all around us. Social systems, the human brain, the Internet and the World Wide Web are all examples of complex networks. Finally, it is not difficult to understand that they have become one of the hottest research fields in science. In fact, if we want to master the interconnected world we live in, we need to understand the structure of the networks around us.

Networks have originally been understood as a collection of nodes, representing elementary units of the system, and edges, describing the existence of interactions between pairs of such units. Applications to real-world systems, however, require the possibility to describe more details of an interaction, like for example: directed edges to describe the origin and destination of a message; edge weights, to highlight the intensity of an interaction. In more recent years, mathematical tools have been formalized and developed to analyze temporal networks, where interactions are not static but unfold in the temporal dimension. Similarly, many works have recently considered the case of interacting systems where units can be connected by links of different nature, and which can be effectively represented in terms of multilayer networks.

Although over the past decades, a great variety of complex systems have been successfully described with these kind of models, in face-to-face human communication, chemical reactions and ecological systems, interactions can occur in groups of three or more nodes and cannot be simply described just in terms of simple dyads. As a consequence, mathematical frameworks have been proposed in order to describe group interactions explicitly and naturally. Simplicial complexes and hypergraphs are the natural candidates to provide such descriptions.

In parallel, while advances in computing resources have made processing enormous amounts of data possible, human ability to identify patterns in such data has not scaled accordingly. Efficient computational methods for condensing and simplifying data are thus becoming vital for extracting actionable insights. With this in mind, summarizing interconnected data, or graphs, become popular. Graph summarization has various benefits which include for example reduction of data volume and storage or speedup of graph algorithms and queries. Given its advantages, graph summarization has extensive applications, including clustering, classification, community detection, outlier detection, pattern set mining, finding sources of infection in large graphs, and visualization, among others. Consequently the problem of graph summarization has been studied algoritmically in the fields of graph mining and data management, while interactive exploration of the data and appropriate display layouts have been studied in visualization.

However, the literature still lacks of an extension of practical summarization procedures to the higher-order domain. In this work we propose an algorithmic solution to the problem of hypergraph summarization. Our aim is to extend a node aggregation-based graph summarization method to the case of higher order interactions implemented with the hypergraph mathmatical formalism. At the hearth of our algorithm is the condensation of hypergraph nodes into summarized supernodes. We

have identified as proper supernode candidates the fundamental building blocks of networks: higher-order motifs. These are small patterns of higher-order interactions in a network that are statistically over-represented with respect to a null model. The process of summarization, by definition, removes some information from the original data structure. Intuitively, this introduces some uncertainty into any query or analysis that takes the summarized hypergraph as input. To address this problem, we propose formal probabilistic semantics for evaluating structural queries on hypergraph summaries. Motivated by the above, we also study the problem of finding a "good" hypergraph summary. In our problem setting, we present metrics to measure the quality of a summary by the extent to which it alters the results of queries. Finally, we propose a Python implementation which has been tested on a number of real-world datasets coming from different domains. We identify different heuristics to reshape our greedy solution to perform better according to the different dataset characteristics.

This thesis is organized as follows:

- At the beginning we give a brief overview of the recent domain of network science.
- We discuss the limitations of network models able to encode only pairwise interactions, and give a short illustration of the mathematical frameworks to model group interactions.
- After that, we present the solutions introduced by Quintino Francesco Lotito et al. in the paper *Higher-order motif analysis in hypergraphs* to perform higher-order motif discovery [13].
- At this point, we provide a synthetic overview of the state-of-the-art methods proposed by the literature for summarizing graph data. In particular, we strain the attention on a representative node aggregation-based summarization algorithm put forward by Kristen LeFevre and Evimaria Terzi called GraSS (Graph Structure Summarization).
- Then we propose our contribution to the novel problem of hypergraph summarization.
- We evaluate our solution on a number of real-world datasets from different domains. In order to obtain better results, we show some heuristics which have revealed to be adapted for certain frameworks.
- We conclude the thesis discussing briefly about future research directions.

1 Complex networks

The purpose of this chapter is to provide an introduction to some fundamental principles of *network science*. For the sake of clarity and completeness, we also introduce *graph theory* which is the branch of mathematics that deals with the study of graphs, which in turn are the mathematical objects used to represent networks.
The study of networks to understand complex systems made of interacting entities, stands out for its interdisciplinary nature, the generality of the results obtained, and the wide variety of possible applications. We present some properties which are commonly taken into consideration when analyzing a complex network.

1.1 Basic concepts

Imagine to be a small curious fly which is flying around a university. In the morning, before the start of the lessons, you observe that students, researchers, professors and other people start to populate the structure. They start to talk in small groups, usually of two people, then the groups grow in size, they split, merge again, change shape. Some of the people move from one group to another. Some of them know each other already, while others are introduced by mutual friends. What is

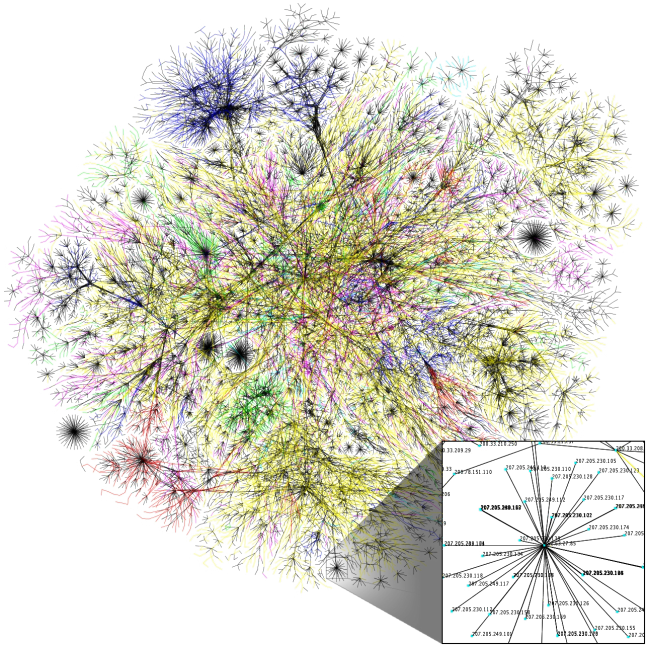


Figure 1.1: Partial map of the Internet based on the January 15, 2005 data found on [26]. Each line is drawn between two nodes, representing two IP addresses. The length of the lines are indicative of the delay between those two nodes. Figure from [29].

more, you could notice that we can somehow partition the groups into communities, since for example professors and administrative staff members do not typically join students' groups. Furthermore each person is different from the others. Hence, some individual are more lively and tend to attract the attention of other people. On the other hand, others are more shy: they stay in smaller groups and prefer to listen to the others. Perhaps you also notice someone in front of a coffee machine. In this room people are organized into more or less regular queues, so that the shape of this groups is different. The social system we have just considered is a typical example of what is known as a *complex system*: a system made by a large number of single units (individuals, components or agents) interacting in such a way that the behaviour of the system is not a simple combination of the behaviours of the single units. With this definition in mind, we clearly understand that complex systems are all around us, permeate all aspects of our life and constitutes the backbone of our modern world.

Over the years, science has abandoned the idea that the collective behaviour of a complex system can be simply understood and predicted by considering the units of the system in isolation [4]. Within this paradigm, networks have emerged as a reference modeling tool for complex systems [1]. The natural framework for the exact mathematical treatment of a complex network is a branch of discrete mathematics known as graph theory and in particular the concept of *graph*. Graphs are used to represent sets of objects and pairwise interactions among them. This mathematical framework is well established, since a lot of problems can be abstracted in terms of graphs.

1.2 Measures, properties and applications

Over the years, the literature has come up with quantitative measures and has highlighted interesting features in order to understand the behaviour and the peculiarities of these complex systems made of interacting entities. For instance, if we look at the connections between the neurons in the brain and construct a similar network whose nodes are neurons and the links are the synapses which connect them, we find that such a network has some special mathematical properties which are fundamental for the functioning of the brain. For instance, it is always possible to move from one node to any other in a small number of steps, and, particularly if the nodes belong to the same brain area, there are many alternative paths between them [9].
Commonly used parameters to study complex system are *centrality measures*. Network centralities

are node-related measures that quantify how "central" a node is in a network. There are many ways, in which a node can be considered so: for example, it can be central if it is connected to many other nodes. This is the case of *degree centrality*. This quantity corresponds to nodes' degree. The *degree* of a vertex v, is the number of neighbors of v. Intuitively, the higher the degree of a node, the more sources of information it has available, and the quicker the information will reach the node, so the more central it is.

Interestingly, most of networks in nature, from social to biological networks, display non-trivial topological features that cannot usually be reproduced by random networks [6].

Scale-free networks The World Wide Web can be represented with a network quite naturally. The vertices of this network are webpages, while the directed links represent hyperlinks pointing from one document to another. Clearly, this is a perfect example of a self-organised technological system: the topology is the result of billions of agents acting independently, so there are no a priori reasons to find large deviations from a random graph. However, the degree distributions of the WWW are totally different from that of a random graph. In fact, in the WWW, most vertices are sparsely connected, while a few vertices have an extremely large number of links and therefore play a crucial role for the functionality of the network. In particular, it has been shown that the degree distribution of this network is *scale-free*, i.e. the probability p_k that a network node has degree k follows a power-law distribution. Interestingly, scale-free functions are ubiquitous in nature and in made-man systems. In fact, the distributions of a wide variety of physical, social and biological phenomena can be well approximated by a power law over a wide range of magnitudes [9].

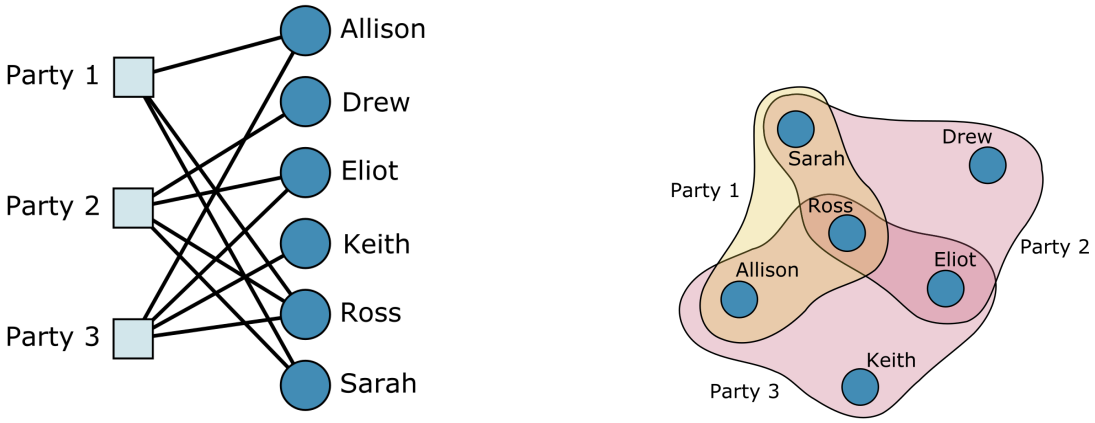
Small-World networks In our life experience, sometimes we discover that we unexpectedly share a common acquaintance with strangers. This is the so called *small-world property* of networks, for which most nodes are not neighbors of one another, but the neighbors of any given node are likely to be neighbors of each other and most nodes can be reached from every other node by a small number of steps. This behaviour is not just a feature of social systems, but can also be found in neural networks of different organisms, and in other biological and man-made networks [9].

Community structure Real-world networks nodes are often organised into *communities*, i.e. clusters of nodes such that nodes within the same cluster are more tightly connected that nodes belonging to two different clusters. In such cases we say that the networks have a *community structure*. The most important point is that nodes in the same network cluster usually share common features. For example tightly connected groups of nodes in the World Wide Web correspond to pages on common topics [9].

2 Beyond pairwise interactions

The concept of network which has been introduced in Chapter 1 has been successfully adopted in vast applications in various domains. However, this model leads quite often to a simplified representation of reality. The fundamental limit of networks is that they capture pairwise interactions only, while many systems display group interactions. Indeed, in social systems, ecology and biology among other examples, many connections and relationships are collective actions at the level of groups of nodes. For instance, three or more species routinely compete for food and territory in complex ecosystems [4].

In this chapter we present mathematical frameworks which have been theorized to overcome this limitation, allowing an explicit representation of *higher-order systems*. In particular we focus on the hypergraph model, providing an overview about the ways it can be described and represented. In addition, we discuss some measures that can be used to characterize and quantify structural properties of this data structure. Finally, we provide possible applications of these notions in order to detail a high-order interacting system.



(a) Bipartite graph

(b) Hypergraph

Figure 2.1: Parallel between the framework of the bipartite graphs (a) and the formalism of the hypergraphs (b). Both model the same system, however hypergraphs explicitly model higher-order interactions and bipartite graphs need to introduce new accessory nodes. Picture from [4].

2.1 Modeling higher-order interactions

So far we have understood how interactions can describe different situations in real systems. Formally, we define an *interaction* as a set $I = [p_0, p_1, ..., p_{k-1}]$ containing an arbitrary number k of basic elements of the system under study, which we indicate as nodes or vertices. Up to this point, we have taken into consideration only *low order systems* in which only self or pair-wise interactions take place. However, in many circumstances, it is more convenient to represent a complex network by means of *higher-order systems* (hereafter usually referred to as HOrSs) which display interactions in groups of more than two elements.
We define an *interaction system* (V, I) as the family of interactions $I = \{I_0, ..., I_n\}$ taking place on a node set V. To aid the intuition, let us make a specific example. Consider the set of friends V = [Allison, Drew, Eliot, Keith, Ross, Sarah] and the set of interactions I = {[Allison, Ross, Sarah], [Drew, Eliot, Ross, Sarah], [Ross, Keith, Eliot, Allison]}. Perhaps, each interaction abstracts a party attended by the involved vertices. Actually, we cannot capture the properties of this interacting system through a graph representation, since it is impossible to explicitly describe group interaction.

Bipartite graphs are an effective way to describe group relationships within the realms of low-order interactions. A bipartite graph is a graph whose vertices can be divided into two disjoint sets U and W; the edgeset E contains only edges (u, w) such that $u \in U$ and $w \in W$. To represent higher-order interactions, one chooses U to coincide with the original nodeset V, i.e. $U = V$, and W to coincide with the set of interactions I. The links in the bipartite graph connect a node (in V) to the interactions in which it takes part (Figure 2.1). Clearly with this approach the entire information is preserved. However, the nodes of the original system do not interact directly with each other anymore. Rather, their relation is always mediated by the interaction layer, which is of a different nature from the node layer itself. This constitute an additional complexity which needs to be taken into account when applying this formalism.

Hypergraphs provide the most general and unconstrained description of higher-order interactions. Formally, a hypergraph is a pair $H = (V, E)$ where V is the set of the vertices and $E \subseteq \mathcal{P}(V)$ is the set of the *hyperedges*. A hyperedge e is a subset of the power set of V, and can link any number of vertices. Each hyperedge specifies which nodes partecipate in which way within an interaction. As shown in Figure 2.1, this formalism is definitely the most natural and appropriate way to describe our toy example of interacting system.

	Party 1	Party 2	Party 3
Allison	1	0	1
Drew	0	1	0
Eliot	0	1	1
Keith	0	0	1
Ross	1	1	1
Sarah	1	1	0

(a) Incidence matrix

	Allison	Drew	Eliot	Keith	Ross	Sarah
Allison	0	0	1	1	2	1
Drew	0	0	1	0	1	1
Eliot	1	1	0	1	2	1
Keith	1	0	1	0	1	0
Ross	2	1	2	1	0	2
Sarah	1	1	1	0	2	0

(b) Adjacency matrix

Figure 2.2: Matrix representations of the interacting system illustrated in Figure 2.1

2.2 Hypergraph representations and measures

In the interest of applying hypergraphs to understand complex systems made of interacting entities and to make predictions about future states, it is crucial to introduce novel representations and significant measures to exploit interesting properties.

The definition of incidence matrix can be easily extended to the case of higher-order interactions. In the case of hypergraphs, we define the incidence matrix as a $n \times m$ matrix I, where n is the total number of nodes, while m is the number of hyperedges. The entry $I_{i\alpha}$ in row i and column α is 1 if node i belongs to interaction α, and zero otherwise.
Similarly to the case of common graphs, hypergraphs can further be properly detailed by means of an adjacency matrix A. While for simple graphs there can be at most one edge connecting a pair of nodes i and j, for HOrSs there can be more than one hyperedge α containing the two nodes. The adjacency matrix of a HOrS is then a $n \times n$ matrix whose elements a_{ij} are the number of hyperedges that contain both i and j.

Perhaps the degree of the nodes and the *order* of the hyperedges, are the first measures one can use to study the properties of HOrSs. The degree $d(v)$ of a vertex v is the number of edges that contain it. On the other hand the order, usually referred also as *size* of a hyperedge e, corresponds to the number of vertices linked by e.
In addition to these local measures, it is useful to introduce also the concept of *intersection profile*. From the incidence matrix I, one can define the intersaction profile of a HOrS as $P = I^T I$, which is an $m \times m$ matrix, whose elements $P_{\alpha\beta}$ count the number of vertices in common between two hyperedges α and β and m is the number of hyperedges.

2.3 Applications

Non-pairwise interactions are common in various types of systems in the real world. For instance, modeling higher-order interactions can play a key role in studying *ecological networks*, in which groups of more than two species compete for a common prey [11]. Furthermore, we can adopt the formalisms introduced in this chapter to map higher-order correlations in neuronal functional patterns [24]. In particular, scientists have realized the importance of hypergraphs to describe affiliation data [27]. *Affiliation networks*, also known as *membership networks*, are social networks representing the affiliation of a set of n actors to a set of m events, or social occasion. Hence, the case illustrated in Figure 2.1, is an example of affiliation network. Authorship of scientific articles is a particularly interesting type of affiliation networks. In this case, the two sets of nodes represent scientists and their publications, respectively. About this domain, Quan Xiao points out the modelling limitations of traditional networks based on classical graph theory with the increase in network complexity [31]. Indeed, co-authored publications often involve groups of authors rather than just two. As a result, representing collaborations with ordinary network such that all nodes are connected with each other, inevitably leads to loss of information. Actually, in this case, it is difficult to distinguish whether it is co-author in pairs or multi-author collaboration. With the aim of describing this scenario, Quan Xiao proposes Figure 2.3 [31].

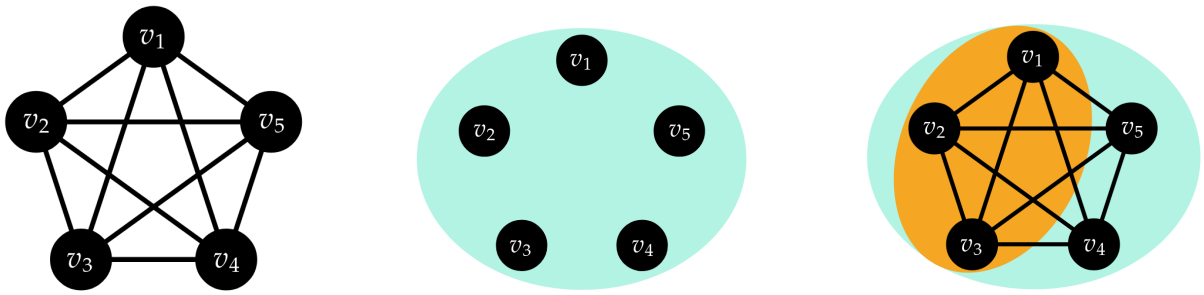


Figure 2.3: v_1,v_2,v_3,v_4,v_5 are five authors of papers. (a) In this case each two of the five authors collaborated, as a result 10 papers are written totally; (b) Five authors writing one paper collaboratively; (c) In this case each two of the five authors collaborated, and the five authors wrote one paper collaboratively, and what is more, authors 1, 2, and 3 wrote another paper collaboratively. Figure from [31].

3 Higher-order network motifs

While many networks in nature share some global properties such as power-law degree distributions, small-world property and community structure, networks from different domains tend to display differences in their local structure. In this chapter, we introduce the notion of networks motifs, patterns of connectivity at the network microscale.
In what follows we discuss the results obtained by Lotito et al. with the aim of extending the concept of motif in order to characterize the local structure of systems that involve group interactions [13].

3.1 Network motifs

So far we have written about methods to analyse complex networks by means of global features. In 2002, Milo et al. introduced the notion of *network motifs* in order to characterize the local structure of complex networks. Network motifs are small patterns of interactions that appear in an observed network at a frequency that is statistically-significantly higher than in a randomized network [18].
In their work, Milo et al. developed an algorithm for detecting network motifs and applied it to several networks from biochemistry (transcriptional gene regulation), ecology (food webs), neurobiology (neuron connectivity), and engineering (electronic circuits, World Wide Web). Each network was scanned for all possible n-node subgraphs, and the number of occurrences of each subgraph was recorded. They analyzed only small patterns of interactions ($n = 3,4$), since overall the enumeration problem of all the connected size-n subgraphs of a network is very expensive from a computational point of view. Their work starts taking into consideration complex systems where interactions can be conveniently represented by directed edges between nodes (Figure 3.1 A). In the paper they report 13 types of three-node connected subgraphs which have been considered in this process (Figure 3.1 B).

According to the definition provided at the beginning of this section, when evaluating the over- or under-expression of each motif in a network, we need to compare their occurrences in real-world networks with respect to an ensemble of randomized networks. To generate this latter we rely on a model known in the mathematical literature as the *configuration model*. The configuration model is a method to describe ensembles of random graphs with N nodes, K edges, and a given degree sequence, so that the degree of each node is predefined. This technique allow us to reproduce the single-node statistics of the observed network of our interest.
In practice, a simple way to generate a graph in the ensemble defined by a given degree sequence $K = \{k_1, k_2, ..., k_N\}$, consists in assigning to each node i a number of *half-edges*, also known as *stubs*,

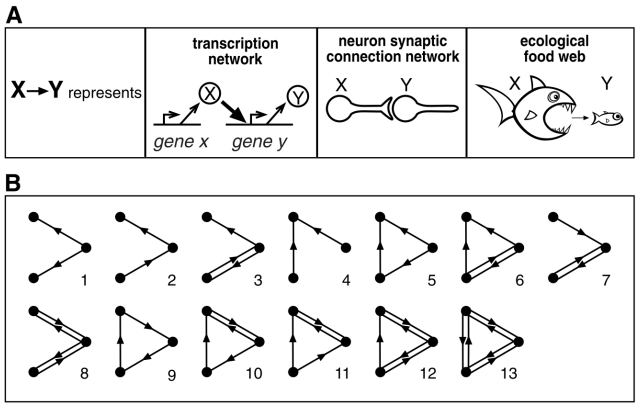


Figure 3.1: (A) Examples of interactions represented by directed edges between nodes. (B) All 13 types of three-node connected subgraphs. Figure from [18].

equal to its degree k_i. A graph in the ensemble is then formed by matching, at random with uniform probability, pairs of half-edges together, until all K edges of the graph are formed.
The statistical measures commonly used to evaluate the significance of the number of motifs occurrences with respect to an ensemble of randomized networks, are the *z-score* (Z_i), the *relative abundance* (Δ_i) and the *Significance Profile* (SP_i^{Δ}).

$$Z_i = \frac{N_{real_i} - \langle N_{rand_i} \rangle}{std(N_{rand_i})} \quad ; \quad \Delta_i = \frac{N_{real_i} - \langle N_{rand_i} \rangle}{N_{real_i} - \langle N_{rand_i} \rangle + \epsilon} \quad ; \quad SP_i^{\Delta} = \frac{\Delta_i}{\sqrt{\sum \Delta_i^2}}$$

In both the equations i refers to the i-th motif; N_{real_i} is the number of times subgraph type i appears in the network; $\langle N_{rand_i} \rangle$ is the mean of its appearances in the randomized network ensemble; $std(N_{rand_i})$ is the standard deviation of its appearances in the randomized network ensemble; ϵ ensures that $|\Delta|$ is not misleadingly large when the subgraph appears very few times in both the real and random networks; SP is a normalized vector made of the statistical measures related to each motif.
These quantities can be used in order to compare the local structure of networks from different fields. This analysis can reveal the emergence of "superfamilies" of networks, meaning clusters of networks which display similar local structure. This suggests that motifs can define broad classes of networks, each with specific types of elementary structures. Indeed, motifs reflect the underlying processes that generated each type of network; for example, food webs evolve to allow a flow of energy from the bottom to the top of food chains, whereas gene regulation and neuron networks evolve to process information. Actually, information processing seems to give rise to significantly different structures than does energy flow [17].
An exemplification of this concept is illustrated in Figure 3.2. The chart illustrates the SP of the 13 possible directed connected triads for networks from two different fields. The first one represents three WWW networks of hyperlinks between Web pages related to university, literature, or music. The second domain is about three social networks in which nodes represent people in a group and edges represent positive sentiment directed from one group member to another, based on questionnaires. Notably, in both cases triads 9, 10, 12, 13 occurs more frequently, on the other hand triads 4, 5, 6 are more infrequent. This similarity between World Wide Web and social networks suggests that they belong to the same "superfamily" and as a consequence classical models of social structural organization may also be used to understand WWW structure.

3.2 Higher-order motif analysis in hypergraphs

In the paper *Higher-order motif analysis in hypergraphs* [13], Lotito et al. propose a generalization of the notion of motifs to the framework of the higher-order networks represented by hypergraphs. Higher-order network motifs are small patterns of higher-order interactions that appear in an observed hypernetwork at a frequency that is statistically-significantly higher than in a randomized hypernet-

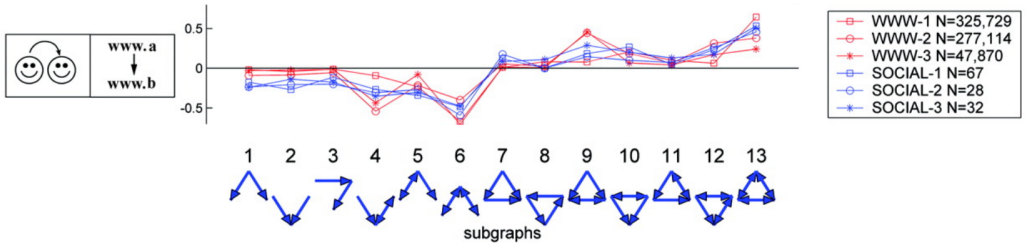
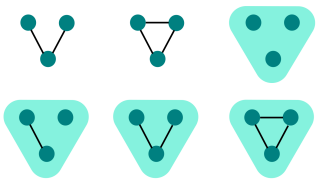


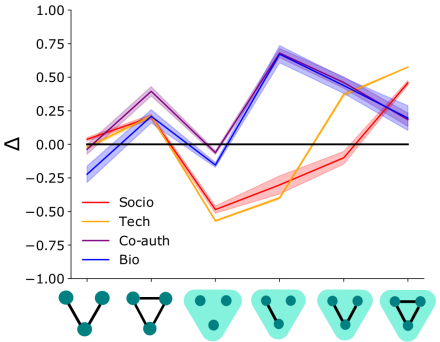
Figure 3.2: *SP* of the 13 possible directed connected triads for networks from different fields: WWW hyperlinks between Web pages in the www.nd.edu site (WWW-1), pages related to literary studies of Shakespare (WWW-2), pages related to tango, specifically the music of Piazzolla (WWW-3); and social networks, including inmates in prison (SOCIAL-1), sociology freshmen (SOCIAL-2), college students in a course about leadership (SOCIAL-3). Figure from [17].

work.
While there is no closed form to express the number of higher-order motifs as a function of the order n, in their work, Lotito et al., demonstrate a lower- and upper-bound on this number. Let m be the number of all the possible non-isomorphic connected hypergraphs with n vertices, then: $m = \mathcal{O}(2^{2^n-n-1})$; $m = \Omega(\frac{2^{2^n-2n}}{n!})$. Two finite hypergraphs are *isomorphic* if they are equivalent modulo relabeling of the vertices. An *isomorphism class* is a collection of mathematical objects isomorphic to each other. Due to the combinatorial explosion of the number of possible patterns given n nodes, in their work Lotito et al. focus on the analysis of motifs of order 3 and 4. In particular there are 6 possible patterns of higher-order interactions which involve three nodes (Figure 3.3a), while considering 4 nodes, there exist 171 different higher-order motifs.

In Algorithm 1 it is reported the pseudocode which is proposed by Lotito et al. as an efficient algorithm to solve the problem of counting the frequency of the higher-order motifs of order 3 of a network. Each different motif represents an isomorphism class. The problem of motifs counting can be interpreted as enumerating all the possible connected sub-hypergraphs of size n (in this case $n = 3$) and assigning an isomorphism class to each of them. In line 3 a hash map is initialised and used as a counter of the occurrences of each isomorphism class. In line 4 we iterate over all the hyperlinks of order 3 and then recover the inner pairwise links to build the motif, whose configuration is stored in *motif*. In order to check the presence of a hyperedge efficiently we can hash every hyperedge of the hypergraph: in this way it is possible to check the existence of an hyperedge in constant time. At this point we rely on a prior algorithm named ESU [28] in order to discover motifs involving 3 nodes that are composed only by pairwise relations, ignoring higher-order interactions which have been already taken into account. As reported in the iterative loop in line 17, every time ESU outputs a subgraph, the triplet of nodes could have been counted already in the previous step because of an overlap between a pairwise motif and a hyperlink of order 3. In this case the triplet is discarded.
For higher-order motifs of order 4, the situation is similar, albeit there are some more details to take into account (Algorithm 2).
What is more in *Higher-order motif analysis in hypergraphs* [13], analogously to what has been described above regarding low order network motifs, a set of available network datasets from different domains have been analyzed in order to identify higher-order families of hypergraphs, characterized by their set of higher-order pattern of local interactions. In particular they have been used the parameters introduced in Section 3.1 for the purpose of studying the over- and under-expression of the patterns of higher-order interactions involving three or four nodes. With the aim of generating randomized networks needed to assess the statistical significance of each motif, it has been adopted a generalization of the classical dyadic configuration model to hypergraphs as proposed by Chodrow et al. [5]. In this manner we can highlight the different frequencies of the higher-order motifs in the different domains, allowing to understand the relative structural importance of certain patterns of interactions. For example, considering higher-order motifs involving three nodes, we can notice that in the social and technological domain the motif composed by an hyperlink of size 3 and a triangle



(a) Enumeration of all the six possible patterns of higher-order interactions involving three nodes. Figure from [13].



(b) Differences in the expressions of the higher-order 3-nodes motifs in the different domains. Figure from [13].

of dyadic edges is strongly over-expressed, suggesting that people interacting in groups also tend to interact individually (Figure 3.3b).

4 Graph summarization

As technology advances, the amount of data that we generate and our ability to collect and archive such data both increase continuously. Daily activities like social media interaction, web browsing, product and service purchases, itineraries, and wellness sensors generate large amount of data, the analysis of which can immediately impact or lives. This abundance of generated data and its velocity call for data summarization. While data summarization techniques have been studied extensively, only recently has summarizing interconnected data, or graphs, become popular [12]. The purpose of this chapter is to provide a synthetic overview of the state-of-the-art methods proposed by the literature for summarizing graph data.

4.1 Graph summarization methods

Overall, the notion of graph summary is not well defined. A summary is application-dependent and can be defined with respect to various goals: it can preserve specific structural patterns, focus on some network entities, preserve the answers to graph queries, or maintain the distributions of graph properties. In a nutshell, the key objective of graph summarization include query efficiency and approximate computations, compression and data size reduction, static or temporal pattern discovery, visualization and interactive large-scale visual analytics, influence analysis and understanding, entity resolution, and privacy preservation.
In order to face the difficult and multifaceted problem of graph summarization, the literature has proposed several methods which can be categorized based on the type of data handled and the core techniques employed. In this sense, we report a possible structured classification of the main graph summaries types whose peculiarities are covered in the following. Actually, in this thesis we strain the attention on the problem of summarization of static graphs, which can be formally described as follows: given a static graph G or its adjacency matrix A, find a summary graph or a set of structures or a compressed data structure to concisely describe the given graph. In what follows we resume the methods which are commonly adopted in order to solve this latter problem.

Grouping-Based Methods

Grouping-based methods are among the most popular techniques for summarization. These methods are distinguished into two main categories:

1. Node-Grouping Methods. Some approaches employ existing clustering techniques to find clusters that then map to supernodes. Others recursively aggregate nodes into supernodes, connected via

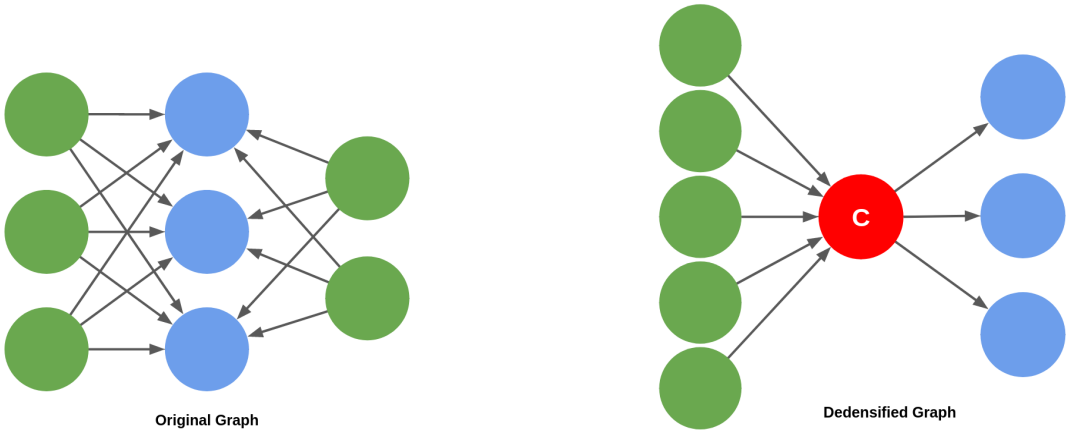


Figure 4.1: Example of graph dedensification. Many edges are removed after the addition of the compressor node C, which connects to the high-degree blue nodes. Figure from [15].

superedges, based on an application-dependent optimization functions. We refer to the former as "Node clustering-based methods" and to the latter as "Node aggregation-based methods".

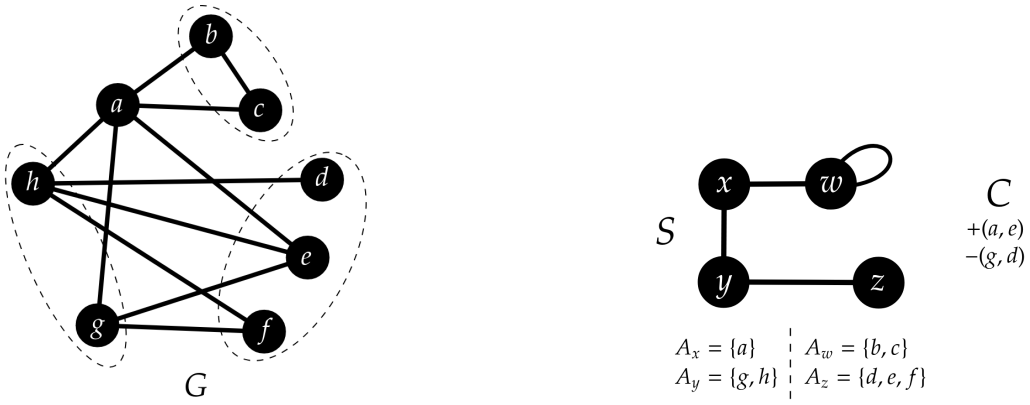
(a) Node clustering-based methods. Although the goal of clustering is not graph summarization, the outputs of clustering algorithms can be easily converted to non-application-specific summaries. In essence, a small representation of the input graph can be obtained by mapping all the nodes that belong to the same cluster/community to a supernode and linking them with superedges with weight equal to the sum of the cross-cluster edges or else the sum of the weights of the original edges (Newman and Grivan [21], Yang and Leskovec [32], Low et al. [14]).

(b) Node aggregation-based methods. One representative algorithm of this category is GraSS [10], which targets accurate query handling. This summarization method supports queries on the adjacency between two nodes, as well as the degree and the eigenvector centrality of a node. We discuss about this summarization method in Section 4.3.

2. Edge-Grouping Methods. Unlike node-grouping methods that group nodes into supernodes, edge-grouping methods aggregate edges into *compressor* or *virtual nodes* to reduce the number of edges in a graph. Note that in this section, "compression" does not refer to bit-level optimization, as in the following section but rather to the process of replacing a set of edges with a node. Following the assumption that high-degree nodes are surrounded by redundant information that can be synthesized and eliminated, Maccioni and Abadi introduce *graph dedensification* [15]. This last is an edge-grouping method that compresses neighborhoods around high-degree nodes, accelerating query processing and enabling direct operations on the compressed graph. An example of this process is illustrated in Figure 4.1.

Bit Compression-Based Methods

The goal of these approaches is to minimize the number of bits needed to describe the input graph, where the summary consists of a model for the input graph and its unmodeled parts. One representative algorithm of this category has been proposed by Navlakha et al. [20]. Given a graph $G = (V_G, E_G)$, the representation for it $R = (S, C)$ has two parts: the first is a graph summary $S = (V_S, E_S)$ (much smaller than the input) that captures the important clusters and relationships in the input graph, while the second is a set of corrections C that helps to recreate the original graph, if necessary. For the sake of clarity, Figure 4.2 shows a sample graph (Figure 4.2a) and its representation (Figure 4.2b). These works employ the *two-part Minimum Description Length (MDL)* code [23], whose goal is to minimize the description of the given graph G and the model class in terms of bits. This principle has its roots in information theory. It roughly states that the best theory to infer from a set of data is the one which minimizes the sum of (i) the size of the theory, and (ii) the size of the data when encoded



(a) Original Graph

(b) Two-part Representation

Figure 4.2: The two part graph representation. The LHS shows the original graph, while the RHS contains the graph summary (S), corrections (C), and the supernode mapping. Figure from [20].

with the help of the theory. In the setting described above, the data is the input graph G, the theory is the summary S and the corrections C essentially represent the encoding of the data in terms of the theory.

Simplification-Based Methods

Simplification-based summarization methods streamline the original graph by removing less "important" nodes or edges, resulting in a sparsified graph. As opposed to the methods above, here the summary graph consists of a subset of the original nodes and/or edges. A representative work on node simplification-based summarization techniques is OntoVis [25], a visual analytical tool that relies on node filtering for the purpose of understanding large, heterogeneous social networks in which nodes and links respectively represent different concepts and relations. OntoVis uses information that relates nodes and edges, such as the degree of nodes of specific type, to semantically prune the network.

Influence-Based Methods

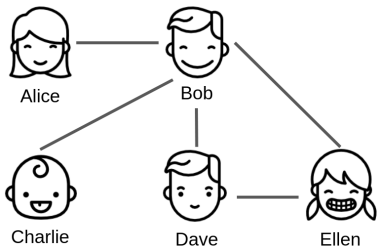
Influence-base methods seek to find a compact, high-level description of the influence dynamics in large-scale graphs to understand the patterns of influence propagation at a global level. Usually such methods formulate graph summarization as an optimization process in which some quantity related to information influence is maintained. These summarization methods have been mostly applied on social graphs, where important influence-related questions arise. Community-level Social Influence (CSI) [16] is a representative work that focuses on summarizing social networks via information propagation and social influence analysis.

4.2 Graph summarization benefits and applications

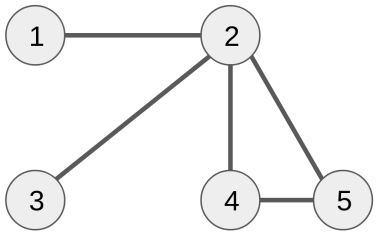
Graph summarization has various benefits, which include the following:

- *Reduction of data volume and storage*: graphs of real-world datasets are often massive. For example, as of August 2017, the Facebook social network had 2 billion users, and more than 100 billion emails were exchanged daily. Summarization techniques produce small summaries that require significantly less storage space than their original counterparts. Graph summarization techniques can decrease the number of I/O operations, reduce communication volume between clusters in a distributed setting, allow loading the summary graph into memory, and facilitate the use of graph visualization tools.

- *Speedup of graph algorithms and queries*: while a plethora of graph analysis methods exist, many cannot efficiently handle large graphs. Summarization techniques produce smaller graphs that maintain the most salient information from the original graph. The resultant summary graph can be queried, analyzed and understood more efficiently with existing tools and algorithms.



(a) Social network graph.



(b) Graph obtained from a social network replacing user names with meaningless integer pseudonyms.

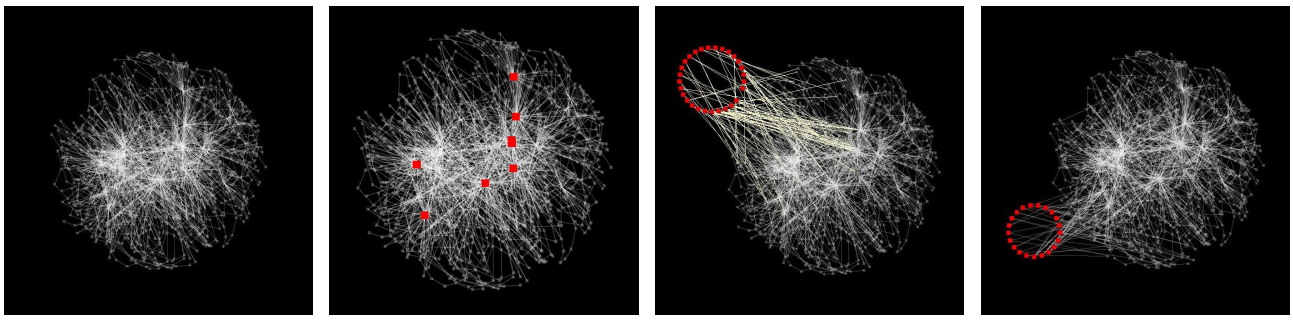
Figure 4.3: Figures from [10].

- *Interactive analysis support*: summarization allows to handle information extraction and speed up user analysis. The resultant graph summaries make it possible to visualize datasets that are originally too large to load into memory.

- *Noise elimination*: real graph data are frequently large scale and considerably noisy with many hidden, unobserved, or erroneous links and labels. Such noise hinders analysis by increasing the workload of data processing and hiding the more "important" information. In this sense, summarization serves to filter out noise and reveal patterns in the data.

- *Privacy and Anonymity*: privacy and anonymity have emerged as important problems when publishing social network graphs. Recent work has observed that removing known identifiers (e.g., Name, SSN) is often not enough to prevent re-identification [3] [7] [19]. As a simple example, consider the social network graph in Figure 4.3a, and suppose that we replace it with the graph in Figure 4.3b, replacing user names with meaningless integer pseudonyms. Now consider an attacker who knows that Bob is in the de-identified graph. If the attacker has some simple information about the graph topology surrounding Bob (e.g., Bob has 4 neighbors), then it is easy for the attacker to locate Bob. Interestingly, Hay et al. have newly demonstrated that a graph summarization approach is sufficient to prevent this particular attack, even in the case of an adversary who has strong structural background knowledge (i.e., knows the entire network topology surrounding the target node) [7].

Given its advantages, graph summarization has extensive applications. As an example we consider the experimental results reported by Koutra et al. [8]. In their work they develop an efficient and effective algorithm called VoG (Vocabulary-based summarization of Graphs) to summarize and understand large real-world graphs. The main idea is to construct a "vocabulary" of subgraph-types that often occur in real graphs (e.g., starts, cliques, chains), and from a set of subgraphs, find the most succinct description of a graph in terms of this vocabulary. The success is measured by means of the Minimum Description Length (MDL) principle, for which a subgraph is included in the summary if it decreases the total description length of the graph. In this sense, the best summary of a graph is the set of subgraphs that describes the graph most succinctly, i.e., compresses it best, and, thus, helps a human understand the main graph characteristics in a simple, non-redundant manner. The motivation behind VoG is that people cannot easily understand cluttered graphs, whereas a handful of simple structures are easily understood, and are often meaningful. In Figure 4.4 we report the results of VoG on the Wikipedia Controversy graph. The nodes are editors, and editors share an edge if they edited the same part of the article. The original graph is depicted in Figure 4.4a. Evidently, no clear pattern emerges and thus a human would have hard time understanding this graph. On the contrary, in Figures 4.4b, 4.4c and 4.4d, we can appreciate the most important structures (i.e., structures that save the most bits) discovered by VoG. In particular, in Figure 4.4b, with red color, there are shown the centers of the most important "stars". Further inspection shows that these centers typically correspond to administrators who revert vandalisms and make corrections. Moreover, Figure 4.4c and Figure 4.4d give the two most important near-bipartite-cores. Manual inspection shows that these



(a) Original Wikipedia Controversy graph. No structure stands out.

(b) VoG: 8 out of the 10 most informative structures are stars whose centers are in red.

(c) VoG: the most informative bipartite graph. It represents an "edit war" between factions (one of them, in the top-left red circle) changing eachother's edits.

(d) VoG: the second most informative bipartite graph. It represents another "edit war" between vandals (bottom left circle of red points) and responsible editors (in white).

Figure 4.4: Summarization and undestranding of the most informative, from an information theoretic point of view, structures of the Wikipedia Controversy graph. Results obtained by applying VoG summarization algorithm. Figure from [8].

correspond to edit wars: two groups of editors reverting each others' changes. For clarity, the members of one group are denoted by red nodes (left), and the edges to the other group are highlighted in pale yellow.

4.3 GraSS: Graph Structure Summarization

In their paper, Kristen LeFevre and Evimaria Terzi, propose a node aggregation-based summarization methods referred to as GraSS (Graph Structure Summarization) [10]. Moreover, they suggest a formal semantics for answering queries on summaries of graph structures. Their work has been a source of great inspiration to achieve our contributions presented in the next chapter. Therefore in this section we succinctly expose the core idea behind the summarization method in question.

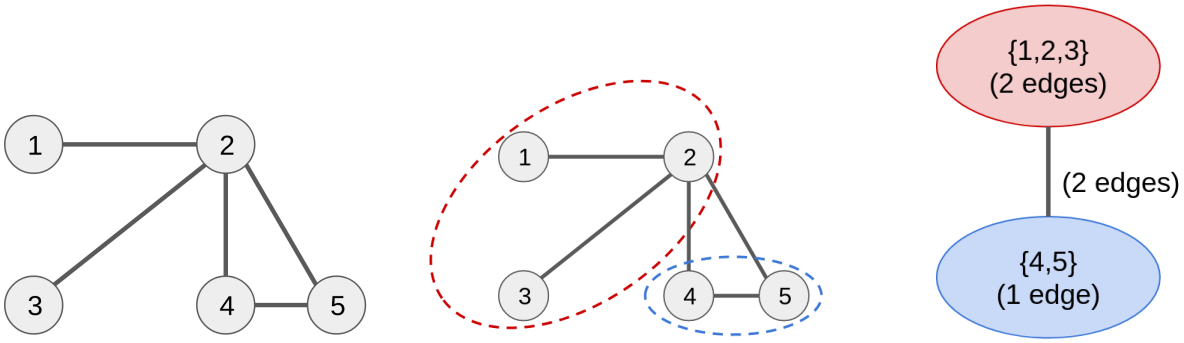
The approach is essentially described in Figure 4.5. The input is a graph $G(V, E)$ that is simple, undirected, and unweighted. As usual, V denotes the set of n nodes $V = \{v_1, ..., v_n\}$ and E denotes the set of edges among these nodes. Given such an input graph $G(V, E)$ a summary $S(G)$ consists of:

1. A *partition* of the nodes of V into parts $\mathbf{V}(V) = \{V_1, ..., V_k\}$, such that $V_i \subseteq V$ and $V_i \cap V_j = \emptyset$, for $i, j \in \{1, ..., k\}$ and $i \neq j$. They refer to each group of nodes V_i as a *supernode* of the summary S.

2. For every supernode $V_i \in \mathbf{V}$, summary S describes the number of edges within the nodes in the supernode. That is, it counts the number of edges in the input graph G that have both their endpoints in the nodes of V_i. For supernode V_i they denote this number by E_i. That is, $E_i = |\{e(u, v)|u, v \in V_i, e(u, v) \in E\}|$.

3. For every pair of supernodes V_i, $V_j \in \mathbf{V}$, summary S also gives the number of edges across the two supernodes. That is, it counts the number of edges in the input graph G that have one of their endpoints in a node of V_i and their other endpoint in a node in V_j. For two supernodes V_i and V_j, they denote this number by E_{ij}. $E_{ij} = |\{e(u, v)|u \in V_i, v \in V_j, e(u, v) \in E\}|$.

Note that for any input graph $G(V, E)$ there exist many possible summaries S. In the paper the set of all possible summaries that can be extracted from an input graph $G(V, E)$ is denoted by $\mathcal{S}(G)$. At the end of this section we overview the basic idea which is operated by LeFevre and Terzi for finding such partition.



(a) Input graph (b) Partitioned graph (c) Graph summary

Figure 4.5: Graph summarization example. Figure from [10].

Replacing original graph G with summary S introduces uncertainty with respect to the structure of G. Intuitively, as mentioned above, given a summary, there are many possible original graphs that could have produced the summary. Following the *principle of indifference* [2], in the absence of additional information, it is reasonable to assume that each such *reconstruction* is equally likely. To capture the uncertainty introduced by summarization, the semantics of a query Q on S should be defined, conceptually, with respect to the set of all valid reconstructions. The key is to utilize an *expected value semantics*. Intuitively, using expected value semantics, the answer to a query Q is the expected result, given a distribution over possible reconstructions. In the absence of additional information, the principle of indifference suggests that it is reasonable to assume a uniform distribution. In formal terms, let $\mathcal{R}(S)$ denote the set of all valid reconstructions from summary S, and let Q denote a query on G with a boolean (0 or 1) or real-valued response. Under expected value semantics, the answer to Q is defined to be the real number e such that: $e = \frac{\sum_{G \in \mathcal{R}(S)} Q(G)}{|\mathcal{R}(S)|}$. Using expected value semantics, in the paper LeFevre and Terzi, demonstrated that some queries can be answered in closed form. Furthermore, they observe that many other interrogations can be answered heuristically, with reasonable accuracy.
By way of illustration, one of the simplest and most common graph-structure queries is the adjacency query, which simply asks: given graph $G(V, E)$ and two nodes $u, v \in V$, does there exist an edge $(u, v) \in E$? Given a summary S, the *expected adjacency matrix* captures the answers to all possible adjacency queries under expected value semantics. The expected adjacency matrix is formally defined in the following way. Let S be a summary graph. The expected adjacency matrix $\bar{A}$ for S is a $|V| \times |V|$ matrix, where all entries are real numbers in the range $[0, 1]$ defined as: $\bar{A}(u, v) = \frac{|\{G(V,E) | G \in \mathcal{R}(S), (u,v) \in E\}|}{|\mathcal{R}(S)|}$. Given a graph summary, it can be proved that each of the entries in the expected adjacency matrix is easily computed in closed form.

In GraSS the *quality* of a summary is measured based on how well the summary $S(G)$ describes the input graph G. Given an input graph G, the goal is to find the summary $S(G)$ such that G's adjacency matrix A and the expected adjacency matrix $\bar{A}$ of summary $S(G)$ are as similar as possible. This intuition is captured by the *reconstruction error*. Intuitively, the reconstruction error measures the average absolute error, resulting from using summary S rather than G, across all possible adjacency queries. The notion is formally defined as follows. Let $G(V, E)$ be an input graph described by a $(|V| \times |V|)$ adjacency matrix A. Let S be a summary of G, and let A be the expected adjacency matrix for S. The (normalized) reconstruction error of S is defined with respect to G as follows: $Re(A|\bar{A}) = \frac{1}{|V|^2} \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} |\bar{A}(i, j) - A(i, j)|$.

We conclude this section describing the technique put forward by LeFevre and Terzi in order to perform graph summarization. In the paper, they give several algorithmic alternatives to face the problem. We limit ourselves describing the essence of their approach. The task is achieved using a greedy algorithm. The computational process starts with a summary graph in which each node is

placed in a separate supernode. In each step, the two supernodes that cause the largest reduction in the objective function are merged. The algorithm repeats until a stopping condition is met. The procedure sets out two main stopping conditions:

1. The easiest case is when the maximum number of supernodes k in the output summary is specified as a constraint. In this instance, we stop the greedy execution when the summary graph contains k super-nodes.

2. Otherwise, if the maximum number of supernodes k in output summary is unknown, the stopping condition states that there is no merging of super-nodes that can improve the objective function. This latter can be defined on the basis of the reconstruction error. As an alternative the objective function can take into account the total number of bits required to encode the summary graph and the input graph $G(V, E)$ given the summary. This idea mirrors the Minimum Description Length principle allude above, where in this case $G(V, E)$ represents the input data, while the summary S of the graph is the model.

5 Hypergraph summarization

Nowadays, it is frequent to address huge datasets which are tough to be conveniently represented. What is more, modeling numerous interacting entities leads to noise and complications in capturing interesting peculiarities of the network. Although several methods have been proposed with the aim of concisely describe low-order graphs, to the best of our knowledge, the literature still lacks of an extension of practical summarization procedures to the higher-order domain. In this chapter we propose our solution to the novel problem of hypergraph summarization. In particular our contribuition can be summarized as:

1. Problem formulation: we introduce a possible way to formalize the process of hypergraph summarization.

2. Algorithmic implementation: we develop an algorithmic method to solve the problem at issue.

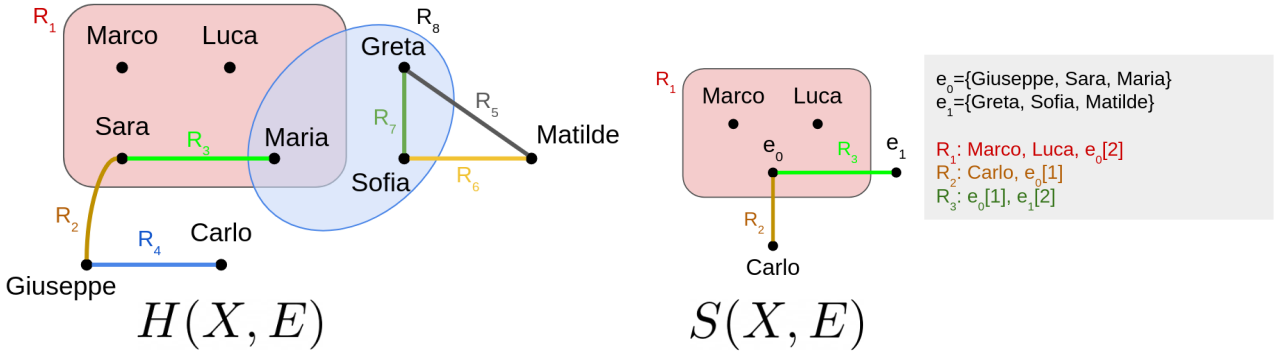
3. Experiments on real datasets: we evaluate our implementation on a set of freely available network datasets from different domains.

5.1 Node aggregation-based hypergraph summarization

Our aim is to introduce a method for the purpose of summarizing a hypergraph. To this end, we propose a node aggregation-based hypergraph summarization method. The input is the representation of a network of entities modelled with a hypergraph $H(X, E)$ made up of a finite set of vertices X linked together through higher-order hyperedges $E \subseteq \mathcal{P}(X)$ (Figure 5.1a).

Given such an input hypergraph $H(X, E)$ a summary $S(H)$ consists of:

1. A *partition* of the nodes of X into parts $\mathbf{X}(X) = \{X_1, ..., X_k\}$, such that $X_i \subseteq X$ and $X_i \cap X_j = \emptyset$, for $i, j \in \{1, ..., k\}$ and $i \neq j$. We refer to each group of nodes X_i as a *supernode* of the summary S. The core of our summarization process is to build supernodes from higher-order motifs. Our baseline algorithm visits the hypergraph and condenses nodes into supernodes whenever a higher-order motif is found in the data structure. In the following of this chapter we discuss about more convenient and deterministic greedy heuristics which have been tested in order to achieve better quality results. For the sake of finding out the motifs we are interested in, we execute the algorithm for higher-order motif discovery implemented by Quintino Francesco Lotito et al. in their work [13]. In the procedure, each supernode is associated with a unique identifier and we map each node of the primary hypergraph with the supernode it belongs to. As explained in Chapter 3, the combinatorial explosion of higher-order motifs makes intractable



(a) Input hypergraph $H(X, E)$. X={Marco, Luca, Sara, Maria, Giuseppe, Carlo, Greta, Matilde, Sofia}; E={R_1{Marco, Luca, Sara, Maria}, R_2{Giuseppe, Sara}, R_3{Sara, Maria}, R_4{Giuseppe, Carki}, R_5{Greta, Matilde}, R_6{Sofia, Matilde}, R_7{Greta, Sofia}, R_8{Maria, Greta, Sofia}}.

(b) Hypergraph summarization output applying baseline algorithm to the input illustrated in Figure 5.1a.

Figure 5.1

their storing and indexing in memory for high orders, as a result we focus on the analysis of the higher-order motifs of order 3 and 4. The choice of obtaining supernodes from motifs is interestingly profitable since they are the fundamental building blocks at the network microscale and so, natural candidates to become a cluster.

2. For every supernode $X_i \in \mathbf{X}$, we conveniently keep track of its category. This last provides hints about the internal configuration of the supernode. For example, we have a distinct category for each of the six possible patterns of higher-order interactions involving three nodes.

3. Each supernode is populated by entities which interacts with other nodes in the primary hypergraph. These entities are replaced and represented by their supernode in the summarized data structure. For each supernode we indicate the cardinality by which it participates to the relationships. This quantity is rigorously expressed by the intersection profile between the target supernode and the hyperedge. In other words we count the number of vertices in common between the hyperedge and the supernode. For example, in Figure 5.1b supernode e_0 shares two nodes with relationship R_1.

A more formal explanation of the proposed summarization method is reported in Algorithm 3.
Note that for any input hypergraph $H(X, E)$ there exist many possible summaries S. We denote the set of all possible summaries that can be extracted from an input hypergraph $H(X, E)$ with $\mathcal{S}(G)$. Figure 5.1b illustrates a possible output of the summarization process given the hypergraph in Figure 5.1a as input. In this case, the execution has first discovered the motif populated by Giuseppe, Maria, Sara that have been condensed into supernode e_0. Then, the computational process has visited the fully connected triplet Greta, Matilde, Sofia that are summed up into supernode e_1. At this point, no more higher-order motif can be found considering the remaining entities. As a consequence, all the nodes which do not belong to any supernode yet (Marco, Luca, Carlo), become supernodes themselves. The summary includes three hyperedges: R_1, R_2, R_3. Marco and Luca are involved in relationship R_1, whereas Carlo is a member of R_2. Supernode e_0 partecipates with two nodes in R_1, with one node in R_2 and with one node in the hyperedge R_3. On the other hand e_1 intersects relation R_3 with two components. In Section 5.3 we reason about how to evaluate the obtained summary. With these metrics we are able to compare more sophisticated variants with respect to the baseline approach.

5.2 Query answering according to expected value semantics

The process of summarization, by definition, removes some information from the original hypergraph. Intuitively, this introduces some uncertainty into any query or analysis that takes the

summarized hypergraph as input. To address this problem, we extend the formal probabilistic semantics covered in Chapter 4 for evaluating structural queries on hypergraph summaries. Intuitively, given a summary, there are many possible original hypergraphs that could have produced the summary. Following the principle of indifference, in the absence of additional information, it is reasonable to assume that each such reconstruction is equally likely. Using this assumption, we define the expected-value response for some query Q. Given a summary S we conceptually answer to each query considering all the possible reconstructions $R(S)$. The purpose of this section is to discuss about the queries which have been considered in our work. In this regard, a useful future research direction is to continue to search for closed-form and heuristic solutions to other queries both with local and global scope.

Expected adjacency matrix Given a hypergraph $H(X, E)$ and two nodes $u, v \in X$, the adjacency between them qualitatively indicates how much u and v are reciprocally adjacent. In other words this quantity measures how many hyperedges are shared between u and v. In order to answer to this interrogation on the given summary S, we build an expected adjacency matrix similarly to what has been treated in Chapter 4. Theoretically, given a summary S, the expected adjacency matrix is properly defined considering all the possible reconstructions $R(S)$. However, for the sake of efficiency and tractability, we introduce a closed form formula to answer to this question. Given a summary S and two nodes $u, v \in S$, in order to compute the entry $\bar{A}(u, v)$ of the expected adjacency matrix, we distinguish between three cases:

1. $u = v$ then $\bar{A}(u, v) = 0$.

2. u and v belong to distinct supernodes. In particular, u is part of supernode U while v is a member of supernode V. These supernodes could share some common hyperedges $\mathbf{R} = \{R_1, R_2, ..., R_n\}$, i.e. U and V are both involved in each $R_i \in \mathbf{R}$. Supernode U partecipates to relationship R_i with an intersection profile iu while the membership of supernode V in R_i is characterized by intersection profile iv. For all $R_i \in \mathbf{R}$ there exist two discrete random variables X and Y such that:

$$X = \begin{cases} 1 & u \in R_i \\ 0 & u \notin R_i \end{cases} \quad ; \quad Y = \begin{cases} 1 & v \in R_i \\ 0 & v \notin R_i \end{cases}$$

   Since both X and Y are independent random variables then:

$$E[XY] = E[X]E[Y]$$

   $E[X]$ express the probability by which node u participates in relation R_i. Simmetrically $E[Y]$ quantifies the likelihood that node v is involved in relation R_i. We can simply compute these values as follows:

$$E[X] = \frac{iu}{|U|} \quad ; \quad E[Y] = \frac{iv}{|V|}$$

   Where $|U|$ and $|V|$ indicate the cardinality of supernode U and V respectively. At the end of the day we can calculate the adjacency value between u and v as:

$$\bar{A}(u, v) = \sum_{\forall R_i \in R} E[XY]_{R_i}$$

3. u and v both belong to the same supernode V. The value $\bar{A}(u, v)$ takes into consideration both the *internal contribution* related to the relationships inside the supernode and the *external contribution* caused by the hyperedges where V is involved. Since these contributions are independent one with respect to the other, the overall expected value can be calculated as:

$$\bar{A}(u, v) = internalContribution + externalContribution$$

   (a) V is a supernode obtained from a 3-nodes motif. The internal contribution depends on the internal configuration of the motif from which the supernode has been obtained. Remember that we know the internal layout of the supernode since we store the corresponding motif

category. The computation can be solved taking into account pairwise and order-three relations separately. We denote with X the discrete random variable which indicates how many pairwise interactions inside V are shared between u, v. On the other hand Y is the discrete random variable which refers to how many order-three interactions inside V are in common between u and v. $cardRel2$ and $cardRel3$ are the number of pairwise interactions in V and the number of order-three hyperedges in V respectively. Overall, the internal contribution can be calculated as:

$$internalContribution = E[X] + E[Y] = \frac{cardRel2}{3} * cardRel3$$

What is more, we need to take into account the external contribution due to the participation of V in a number of hyperedges $\mathbf{R} = \{R_1, R_2, ..., R_n\}$. V intersects each $R_i \in \mathbf{R}$ with intersection profile $iv \in \{1, 2, 3\}$:

$$externalContribution = \begin{cases} 0 & iv = 1 \\ \frac{1}{3} & iv = 2 \\ 1 & iv = 3 \end{cases}$$

The total external contribution is given by the summation of the external contributions related to each $R_i \in \mathbf{R}$.

(b) V is a supernode obtained from a 4-nodes motif. V can host pairwise, order-three and order-four interactions. u and v can participate to these relations independently. Actually, a node can belong to n pairwise hyperedges and this do not affect the probability it appears also in 3-nodes hyperedges. As a consequence, we separately consider the contributions resulted from order two, order three and order four interactions. In this regard, we introduce three independent discrete random variables X, Y, Z, which respectively indicates how many pairwise interactions are there between u, v, the number of 3-nodes hyperedges containing u, v and the number of 4-nodes hyperedges which include u, v. Overall:

$$E[X] = \frac{cardRel2}{6} \quad ; \quad E[Y] = \frac{cardRel3}{2} \quad ; \quad E[Z] = cardRel4$$

$$internalContribution = E[X] + E[Y] + E[Z]$$

Moreover, V takes part to a number of relations $\mathbf{R} = \{R_1, R_2, ..., R_n\}$. V partecipates to each relation with $iv \in \{1, 2, 3, 4\}$.

$$externalContribution = \begin{cases} 0 & iv = 1 \\ \frac{1}{6} & iv = 2 \\ \frac{1}{2} & iv = 3 \\ 1 & iv = 4 \end{cases}$$

The total external contribution is given by the summation of the external contributions related to each $R_i \in \mathbf{R}$.

The rigorous proof of the above formula is based on probability manipulations and is omitted for space. In the algorithm we can agglomerate these formulas into a single closed form equation.

For the sake of clarity, in Figure 5.2a we report an example of adjacency matrix and expected adjacency matrix given hypergraph in Figure 5.1a and the summary in Figure 5.1b.

Expected adjacency degree centrality Given a hypergraph $H(X, E)$ and the corresponding adjacency matrix A, the *adjacency degree centrality* of a node $n \in X$ is defined as:

$$ad(n) = \sum_{\forall j \in X} A(n, j)$$

	Marco	Luca	Sara	Maria	Giuseppe	Carlo	Greta	Matilde	Sofia
Marco	0	1	1	1	0	0	0	0	0
Luca	1	0	1	1	0	0	0	0	0
Sara	1	1	0	2	1	0	0	0	0
Maria	1	1	2	0	0	0	1	0	1
Giuseppe	0	0	1	0	0	1	0	0	0
Carlo	0	0	0	0	1	0	0	0	0
Greta	0	0	0	1	0	0	0	1	2
Matilde	0	0	0	0	0	0	1	0	1
Sofia	0	0	0	1	0	0	2	1	0

(a) Adjacency matrix of the hypergraph in Figure 5.1a.

	Marco	Luca	Sara	Maria	Giuseppe	Carlo	Greta	Matilde	Sofia
Marco	0	1	2/3	2/3	2/3	0	0	0	0
Luca	0	0	2/3	2/3	2/3	0	0	0	0
Sara	1	2/3	0	1	1	1/3	2/9	2/9	2/9
Maria	2/3	2/3	1	0	1	1/3	2/9	2/9	2/9
Giuseppe	2/3	2/3	1	1	0	1/3	2/9	2/9	2/9
Carlo	0	0	1/3	1/3	1/3	0	0	0	0
Greta	0	0	2/9	2/9	2/9	0	0	4/3	4/3
Matilde	0	0	2/9	2/9	2/9	0	4/3	0	4/3
Sofia	0	0	2/9	2/9	2/9	0	4/3	4/3	0

(b) Expected adjacency matrix calculated given the summary in Figure 5.1b.

	R1	R2	R3	R4	R5	R6	R7	R8
Marco	1	0	0	0	0	0	0	0
Luca	1	0	0	0	0	0	0	0
Sara	1	1	1	0	0	0	0	0
Maria	1	0	1	0	0	0	0	1
Giuseppe	0	1	0	1	0	0	0	0
Carlo	0	0	0	1	0	0	0	0
Greta	0	0	0	0	1	0	1	1
Matilde	0	0	0	0	1	1	0	1
Sofia	0	0	0	0	0	1	1	1

(a) Incidence matrix of the hypergraph in Figure 5.1a.

	R1	R2	R3
Marco	1	0	0
Luca	1	0	0
{Sara,Maria,Giuseppe}	2/3	1/3	1/3
Carlo	0	1	0
{Greta,Matilde,Sofia}	0	0	2/3

(b) Expected incidence matrix calculated given the summary in Figure 5.1b.

Analogously, given a summarized hypergraph $S(X, E)$ and the corresponding expected adjacency matrix $\bar{A}$, the *expected adjacency degree centrality* of a node $n \in X$ is defined as:

$$\overline{ad}(n) = \sum_{\forall j \in X} \bar{A}(n, j)$$

Expected incidence matrix Given a hypergraph $H(X, E)$ and its summary $S(X_s, E_s)$, the purpose of the *expected incidence matrix* $\bar{I}$ is to state, given a node $n \in X$ and a hyperedge $h \in E_s$, what is the probability of $n \in h$. By definition node n is part of a supernode N in S. N intersects h with a given intersection profile in. Consequently, the entry of the expected incidence matrix $\bar{I}(n, h)$ can be simply computed as:

$$\bar{I}(n, h) = \frac{in}{|supernode|}$$

where $|supernode|$ is the cardinality of N, which in our work is between one and four. For the sake of clarity, in Figure 5.3a we report an example of incidence matrix and expected incidence matrix given hypergraph in Figure 5.1a and the summary in Figure 5.1b.

Expected degree centrality Analogously to the case of graphs, degree centrality is a relevant way to measure the importance of a node. In the hypergraph field, this quantity indicates the number of hyperedges which contain the target node. Given a summarized hypergraph $S(X, E)$, the *expected degree centrality* of a vertex $v \in X$ is determined by the hyperedges inside the supernode V which includes v and by the interactions in which V is involved. We indicate with $in\_d(V)$ the portion of expected degree centrality because of the internal configuration of the supernode V. On the other hand the expected amount of relationships which contain node v due to the hyperedges in which supernode V partecipates, is expressed by the quantity $ex\_d(V)$. Clearly, $ex\_d(V)$ and $in\_d(V)$ are independent one with respect to the other. As a consequece we can calculate the expected degree

centrality of node v as follows:

$$\bar{d}(v) = ex\_d(V) + in\_d(V)$$

Supernode V appears in a number of hyperedges $\mathbf{R} = \{R_1, R_2, ..., R_n\}$. V intersects each set of nodes $R_i \in \mathbf{R}$ with a certain intersection profile iv. We can compute $ex\_d(V)$ taking into consideration the contribution of each R_i. The contribution of each external hyperedge R_i can be simply calculated as follows:

$$ex\_d(V) = \frac{iv}{|supernode|}$$

Where $|supernode|$ is the cardinality of the target supernode. Alternatively, we can compute $ex\_d(V)$ summing all the entries of the row which represents node V in the expected incidence matrix. We indicate with X the discrete random variable which counts the number of pairwise interactions attended by v inside its supernode V. Y is the discrete random variable which represent the number of 3-nodes hyperedges inside supernode V where we find node v. Finally, Z is the discrete random variable which enumerate the amount of 4-nodes relationships inside V which incorporate node v. Clearly, X,Y and Z are independently distributed one with respect to the other. Given this, it is possible to demonstrate that:

$$in\_d(V) = \frac{2 * cardRel2}{|supernode|} + \frac{3 * cardRel3}{|supernode|} + \frac{4 * cardRel4}{|supernode|}$$

The rigorous proof of the above formula is based on probability manipulations and is omitted for space. As an example, given the hypergraph $H(X, E)$ in Figure 5.1a and the resulting summary in Figure 5.1b, the degree centrality of Giuseppe is 2 while the expected degree centrality is $\frac{8}{3}$.

5.3 Summary quality evaluation

In our work, we also study the problem of finding a "good" hypergraph summary. In our problem setting, we introduce three metrics which can be used in order to evaluate the output of the hypergraph summarization process:

1. *Accuracy*: measure the quality of a summary based on how well the summary S describes the input hypergraph H.

2. *Conciseness*: measure the amount of space that is saved using the summary S instead of the input hypergraph H.

3. *Responsiveness*: operating on the summary S rather than using the original hypergraph H, leads to better query responsiveness. This quantity measures the improvements as regards the answering time to the queries on the data collection.

For each metric, one can define several measures to assess the obtained results. The purpose of this section is to present the quantities that we propose to evaluate the accuracy, conciseness and responsiveness of a given summary. In this regard, a useful future research direction is to continue to search for useful evaluation measures.

Measuring summary accuracy

- *Reconstruction error*: given an input hypergraph $H(X, E)$, we want to find the summary S such that H's adjacency matrix A and the expected adjacency matrix $\bar{A}$ of a summary S are as similar as possible. This intuition is captured by the *reconstruction error*. Intuitively, the reconstruction error measures the average absolute error (resulting from using summary S rather than H) across all possible adjacency queries.

$$Re(A|\bar{A}) = \frac{1}{|X|^2} \sum_{i=1}^{|X|} \sum_{j=1}^{|X|} |\bar{A}(i,j) - A(i,j)|$$

- *Average degree centrality error*: in the previous section we discuss about degree centrality as a significant way to measure the importance of a node. As well as the reconstruction error, the *average degree centrality error* measures the average absolute error that we commit due to the usage of summary S rather than the original hypergraph H when computing the expected degree centrality.

$$ADCE = \frac{1}{|X|}\sum_{i=1}^{|X|}|d(i) - \bar{d}(i)|$$

  In the formula, $d(i)$ and $\bar{d}(i)$ represent the degree centrality of node i given the hypergraph $H(X, E)$ and the expected degree centrality of node i calculated from the summary of H respectively.

Sometimes, due to the considerable size of hypergraphs obtained from real world datasets, the computation of the reconstruction error and of the average degree centrality error considering every node is computationally intractable. As a result, in order to face this limitation, we suggest to select a subset of the original node set in order to conveniently compute these quantities. A proper subset of the original node set can be selected randomly or as an alternative, adopting more sophisticated criteria.

Measuring summary conciseness

- *Number of nodes reduction*: given a hypergraph $H(X, E)$ and the resulting summary $S(X_s, E_s)$, a first interesting way to quantify the conciseness of S is simply through a comparison between $|X|$ and $|X_s|$. For example, we can notice with ease that the hypergraph in Figure 5.1a has $|X| = 9$ while the output summary has $|X_s| = 5$. In this toy instance we get a 44% node reduction; as we can see in Section 5.4, in larger and more concrete scenarios, the results are more appreciable. This measure is significant since the computational complexity of several algorithm executed on hypergraphs depends on the number of nodes.

- *Compression ratio*: given a hypergraph $H(X, E)$ and the corresponding output of the summarization process $S(X_s, E_s)$, we can evaluate the conciseness of the solution calculating the *compression ratio* $cr(S)$.

$$cr(S) = \frac{|E_s|}{|E|}$$

  The ratio quantifies the hyperedge number reduction which characterizes the summary S with respect to the primary data structure H. For example, given the hypergraph in Figure 5.1a as input and the summary in Figure 5.1b as output, the compression ratio is: $\frac{3}{8} = 0.375$. This means that the summary has 62.5% less hyperedges with respect to the initial hypergraph. Clearly, this reduction has a positive impact on several algorithm which are executed on hypergraphs, whose computational complexity is related to the amount of hyperedges.

Measuring summary responsiveness:

- *Execution time*: one of the purpose of hypergraph summarization is to speed up hypergraph algorithms and queries. In order to evaluate the output of the summarization process with respect to this aspect, we time the execution of a hypergraph algorithm. In particular, we have implemented an algorithm which computes the distance between two nodes. Given a hypergraph $H(X, E)$, the corresponding summary $S(X_s, E_s)$ and two nodes $u, v \in X$, the distance between u and v in H corresponds to the number of hyperedges between u and v in the original hypergraph, while the distance between the same couple of nodes in S corresponds to the number of hyperedges between U and V which are the supernodes which includes u and v in the summarized data structure. With the purpose of computing the execution time of this process we adopt the Python module *time* whose method *time*() returns the number of seconds passed since epoch. Nodes u and v are chosen randomly, hence it is often better to execute the computational process more times and consider an average execution time.

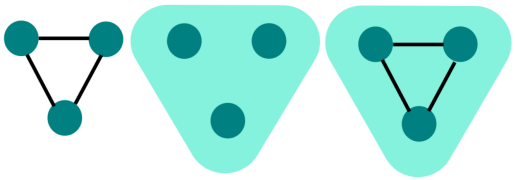


Figure 5.4: Motif categories favoured by heuristic *HE1[3]*. These are the classes which leads to the best knowledge about the internal supernode configuration. For example, in each of these classes we can state with total confidence the amount of hyperedges inside a supernode which are shared between its components.

5.4 Results

In this section we apply the hypergraph summarization algorithm on a set of freely available network datasets from different fields. In order to carry out our experiments, we have studied the following datasets:

- *PACS*: nodes correspond to authors and each hyperedge is the set of authors on a scientific publication in the field of physics.
- *conference*: nodes are participants of a conference. Wearable sensors are exploited to construct a network of proximity contacts among people. Contacts are aggregated in time-windows of 20 seconds. Each hyperedge is a maximal clique in each layer (i.e. each interval) of the temporal network of contacts.
- *Facebook-known-pairs_data_2013*: nodes are students attending a high school. Interactions represent friendship relationships on Facebook.

Using our solution to face the problem of hypergraph summarization, it is really frequent that there exist several motifs in the original hypergraph which share some components. For example, in Figure 5.1a the triplets {Giuseppe, Sara, Carlo} and {Giuseppe, Sara, Maria} are two distinct 3-nodes motifs with two vertexes in common. Both of them are valid candidate supernode. However, this choice is mutually exclusive since, by definition, the summary node set is a partition of the original node set. As a consequence, we can construct a supernode with {Giuseppe, Sara, Maria} or considering the set {Giuseppe, Sara, Carlo}. What is more, we have often to choose between 3-nodes supernodes or the order 4 counterpart. This is the case of the 4-nodes motif {Marco, Luca, Sara, Maria} and the 3-nodes motif {Giuseppe, Sara, Maria}. We have to decide wether condensing the former into a 4-nodes supernode or the latter into a 3-nodes supernode. In order to deal with these alternatives there are no definitive solutions. In this regard, in the following of this section, we report our results applying some greedy heuristics. An open research direction is to seek for new convenient greedy choices in order to achieve better results.

The first approach that has been tested is the baseline algorithm whose pseudocode is reported in Algorithm 3. We refer to this implementation as *HE0[N]* where N is the number of vertexes which populate each supernode bigger than one node in the summary. For example *HE0[3]* takes into consideration only 3-nodes motif and condenses the significant patterns following visit order without making any sophisticated choice. In other words, if the triplet {Giuseppe, Sara, Maria} is discovered before motif {Giuseppe, Sara, Carlo} throughout the hypergraph visit, only {Giuseppe, Sara, Maria} is a supernode in the output summary. In order to evaluate the results of the baseline algorithm, we introduce a first heuristic attempt named *HE1[3]*. Following, this latter approach, we consider only 3-nodes motifs. Whenever there is a collision between two supernode alternatives we give priority to the motif categories reported in Figure 5.4. Actually, these are the classes which allow the best knowledge about the internal supernode configuration. Applying this method on our toy example in Figure 5.1a we would prioritize supernode {Maria, Greta, Sofia} rather than {Sara, Maria, Giuseppe}. In the following we report the results obtained applying these methods on dataset *PACS*, which is the biggest data collection considered in our work.

N	avg error
100	1,81
500	1,55
1000	1,6
5000	1,39
10000	1,42

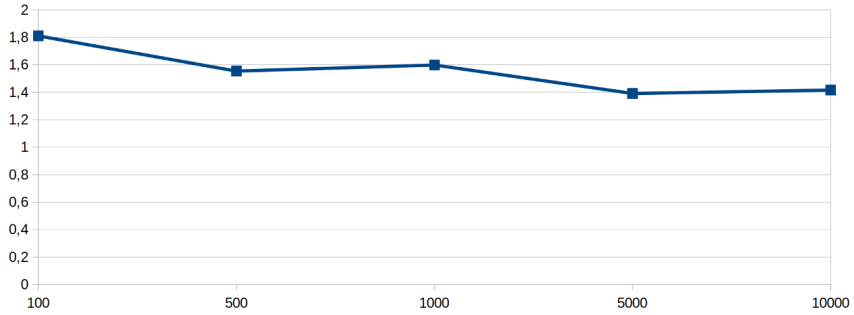


Figure 5.5: Accuracy test. Average degree centrality error. Dataset *PACS*. Heuristic *HE0[3]*. The graph relates the number of nodes (N) randomly selected to perform the computation on the x axis with the average degree centrality error (*avgerror*) due to the usage of the summary instead of the original data structure, reported on y axis.

N	avg error
100	1,53
500	1,22
1000	1,76
5000	1,81
10000	1,46

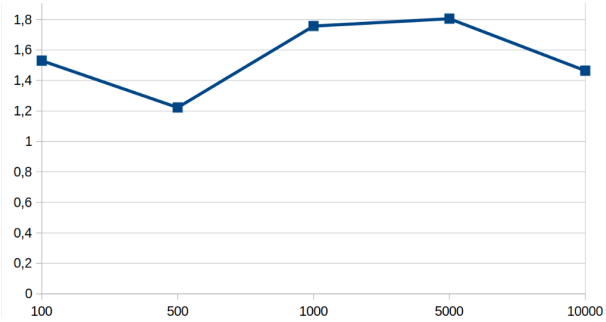


Figure 5.6: Accuracy test. Average degree centrality error. Dataset *PACS*. Heuristic *HE0[4]*. The graph relates the number of nodes (N) randomly selected to perform the computation on the x axis with the average degree centrality error (*avgerror*) due to the usage of the summary instead of the original data structure, reported on y axis.

5.4.1 Dataset *PACS*

Accuracy - Heuristic: *HE0[3]*, *HE0[4]*, *HE1[3]*

In order to evaluate how well the summary S describes the input hypergraph H, we consider the accuracy of the expected degree centrality with respect to the actual degree centrality. Due to the large number of nodes which populate the dataset, we conveniently consider the average degree centrality estimation error among N randomly chosen nodes. Results are reported in Figure 5.5, Figure 5.6 and Figure 5.7. We can observe that with all the heuristics we achieve good summary accuracy. The plotted curve oscillates around the actual error value. Following this behaviour, the line progressively converges on the value that we would appreciate considering the entire node set instead of a random sample. Unfortunately, there is no relevant improvements applying *HE1[3]* with respect to the baseline method.

Conciseness - Heuristic: *HE0[3]*, *HE0[4]*, *HE1[3]*

In the table, the first column reports the target heuristic, the second column reports the number of nodes in the original hypergraph, the third column reports the number of supernodes which populate the output summary, the fourth column reports the number of supernodes with cardinality greather than one, the fifth column reports the achieved compression ratio.

Conciseness - Dataset: *PACS*				
Heuristic	$\|X\|$	$\|X_s\|$	**Supernodes**	**Compression raio**
HE0[3]	316551	282285 (10.825% less nodes)	17133	0.842 (16% less edges)
HE0[4]	316551	**275124 (13.087% less nodes)**	13809	**0.833 (17% less edges)**
HE1[3]	316551	281945 (10.932% less nodes)	14658	0.865 (14% less edges)

N	avg error
100	2,16
500	1,44
1000	1,15
5000	1,41
10000	1,39

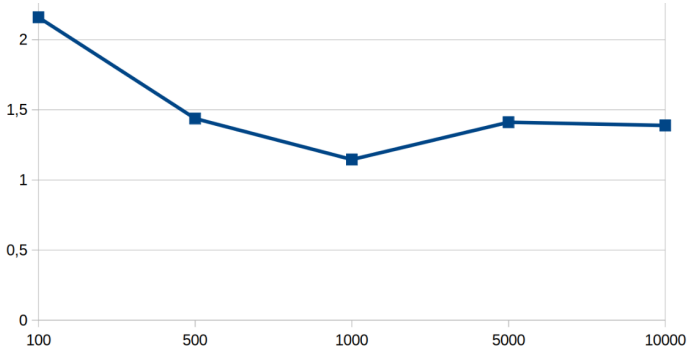


Figure 5.7: Accuracy test. Average degree centrality error. Dataset *PACS*. Heuristic *HE1[3]*. The graph relates the number of nodes (N) randomly selected to perform the computation on the x axis with the average degree centrality error ($avgerror$) due to the usage of the summary instead of the original data structure, reported on y axis.

Responsiveness - Heuristic: *HE0[3]*

We execute the algorithm to compute the distance between two randomly selected nodes in the hypergraph and the algorithm to compute the distance between the two corresponding supernodes in the summarized data structure as explained in Section 5.3. Over the course of this process, we keep track of the time to complete the task. Considering ten executions, the computational procedure takes on average 497.355 seconds in the case of the original hypergraph and on average 446.978 seconds in the case of the summary hypergraph. The best execution returned a result 55 seconds before using the summary rather than the initial hypergraph (11% improvement).

5.4.2 Dataset *conference*

Evaluating the performances resulting from the application of the heuristics that have been discussed so far on *conference* dataset, do not lead to satisfying results in terms of accuracy.

Accuracy - Heuristic: *HE0[3]*

In this case we have fewer nodes, so we can estimate summary accuracy with respect to all the node set rather than considering an average evaluation based on random samples.

Accuracy - Dataset: *conference*		
Heuristic	**Average degree centrality error**	**Reconstruction error**
HE0[3]	22.933	0.306

Conciseness - Heuristic: *HE0[3]*

In the table, the first column reports the target heuristic, the second column reports the number of nodes in the original hypergraph, the third column reports the number of supernodes which populate the output summary, the fourth column reports the number of supernodes with cardinality greather than one, the fifth column reports the achieved compression ratio.

Conciseness - Dataset: *conference*				
Heuristic	$\|X\|$	$\|X_s\|$	**Supernodes**	**Compression raio**
HE0[3]	403	159 (60.546% less nodes)	122	0.976 (2.4% less edges)

Responsiveness - Heuristic: *HE0[3]*

This time we consider 100 executions of the node distance algorithm. The dataset is significantly smaller than *PACS* and as a result the benefits are less rewarding. The computational process takes on average 0.01949 seconds in the case of the original hypergraph and on average 0.01786 seconds in the case of the summary hypergraph.

The reason why we obtain a high average degree centrality error, is because there are a lot of supernodes whose components are structurally too different one with respect to the other. More in depth, we inspect an order three supernode {A,B,C} in the summary. The three components have the following degree centrality values: $d(A) = 3$, $d(B) = 72$, $d(C) = 116$. So different values lead inevitably to an unrepresentative expected degree centrality: $d_{expected}(\{A, B, C\}) = 63.67$.
In order to improve the output of the summarization process, we introduce *motif degree similarity* (MDS) which is a quantity to measure the structural similarity between the components of a supernode. MDS is quantitatively represented by the mathematical variance calculated from the degree centrality measures which characterize the motif components. Subsequently, we introduce a greedy heuristic referred to as *HE2[N]*. This latter computes the motif degree similarity value for each candidate supernode and gives precedence to the clusters distinguished by lower degree centrality variance. The obtained results highlights a significant accuracy improvement because of the introduction of *HE2[N]* heuristic.

Accuracy - Heuristic: *HE2[3]*

Accuracy - Dataset: *conference*		
Heuristic	**Average degree centrality error**	**Reconstruction error**
HE2[3]	2.063	0.265

Conciseness - Heuristic: *HE2[3]*
In the table, the first column reports the target heuristic, the second column reports the number of nodes in the original hypergraph, the third column reports the number of supernodes which populate the output summary, the fourth column reports the number of supernodes with cardinality greather than one, the fifth column reports the achieved compression ratio.

Conciseness - Dataset: *conference*				
Heuristic	$\|X\|$	$\|X_s\|$	**Supernodes**	**Compression raio**
HE2[3]	403	211 (47.643% less nodes)	96	0.981 (1.87% less edges)

Responsiveness - Heuristic: *HE2[3]*
We consider 100 executions of the node distance algorithm. The computational process takes on average 0.01698 seconds in the case of the original hypergraph and on average 0.01653 seconds in the case of the summary hypergraph.

5.4.3 Dataset *Facebook-known-pairs_data_2013*

At this point, we report the results obtained on the Facebook dataset. This time we try also to consider order three and order four motifs at the same time ($HE0[4] + HE0[3]$).

Accuracy - Heuristic: *HE0[3]*, *HE0[4]*, *HE1[3]*, *HE2[3]*, *HE0[4]+HE0[3]*
In this case we have fewer nodes than $PACS$, so we can estimate summary accuracy with respect to all the node set rather than considering an average evaluation based on random samples.

Accuracy - Dataset: *Facebook*		
Heuristic	**Average degree centrality error**	**Reconstruction error**
HE0[3]	5.03	0.119
HE1[3]	3.782	**0.082**
HE2[3]	**0.701**	**0.082**
HE0[4]	5.481	0.134
HE0[4]+HE0[3]	5.506	0.134

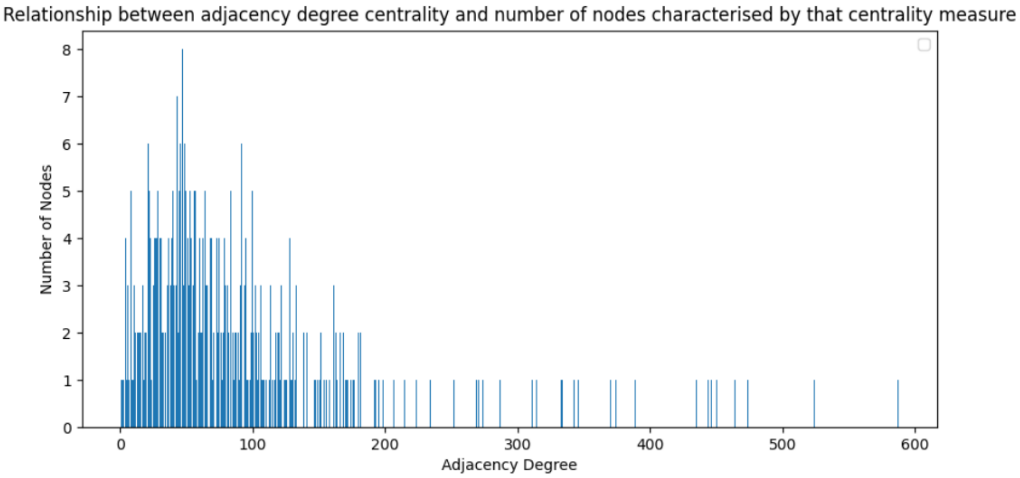


Figure 5.8: Dataset: *conference*. Correlation between adjacency degree centrality (x axis) and number of nodes having that adjacency degree centrality value (y axis).

Conciseness - Heuristic: *HE0[3]*, *HE0[4]*, *HE1[3]*, *HE2[3]*, *HE0[4]+HE0[3]*
In the table, the first column reports the target heuristic, the second column reports the number of nodes in the original hypergraph, the third column reports the number of supernodes which populate the output summary, the fourth column reports the number of supernodes with cardinality greather than one, the fifth column reports the achieved compression ratio.

Conciseness - Dataset: *Facebook*				
Heuristic	$\|X\|$	$\|X_s\|$	**Supernodes**	**Compression raio**
HE0[3]	156	60 (61.538% less nodes)	48	0.925 (7.52% less edges)
HE1[3]	156	74 (52.564% less nodes)	41	0.914 (8.56% less edges)
HE2[3]	156	84 (46.154% less nodes)	36	0.935 (6.47% less edges)
HE0[4]	156	51 (67.308% less nodes)	35	0.909 (9.05% less edges)
HE0[4]+HE0[3]	156	**49 (68.59% less nodes)**	36	**0.907 (9.25% less edges)**

Responsiveness - Heuristic: *HE0[3]*, *HE0[4]*, *HE1[3]*, *HE2[3]*, *HE0[4]+HE0[3]*
We consider 100 executions of the node distance algorithm. The dataset is significantly smaller than *PACS* and as a result the benefits are less rewarding.

Responsiveness - Dataset: *Facebook*		
Heuristic	**Exec. time original hypergraph**	**Exec. time summarized hypergraph**
HE0[3]	0.00219 s	0.00187 s
HE1[3]	**0.00234 s**	**0.00199 s**
HE2[3]	0.00251 s	0.00231 s
HE0[4]	0.00216 s	0.00183 s
HE0[4]+HE0[3]	0.00216 s	0.00182 s

5.4.4 Correlation between degree centrality and average degree centrality error

Finally, our experiments have taken into account the correlation between the nodes having a certain degree centrality in the initial hypergraph and their corresponding average degree centrality error which characterizes them in the output summary. According to what has been discussed in Chapter 1, in most real networks the degree distribution is highly asymmetric: most of the nodes have low degrees while a small but significant fraction of nodes have an extraordinarily high degree [30]. The plots in Figures 5.8, 5.9, highlights this interesting behaviour in the case of the dataset *conference*.

With the aim of studying the average degree centrality error in the summarized data structure as a function of the degree centrality in the original hypergraph, we propose the plots in Figures 5.10, 5.11.

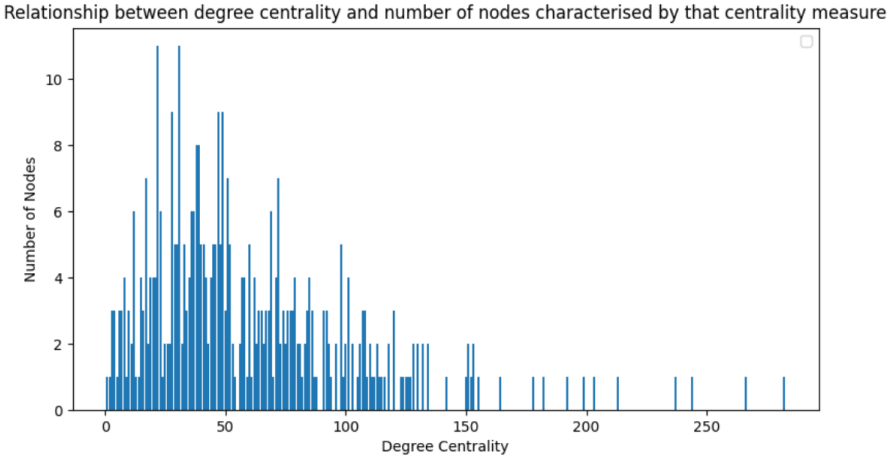


Figure 5.9: Dataset: *conference*. Correlation between degree centrality (x axis) and number of nodes having that degree centrality value (y axis).

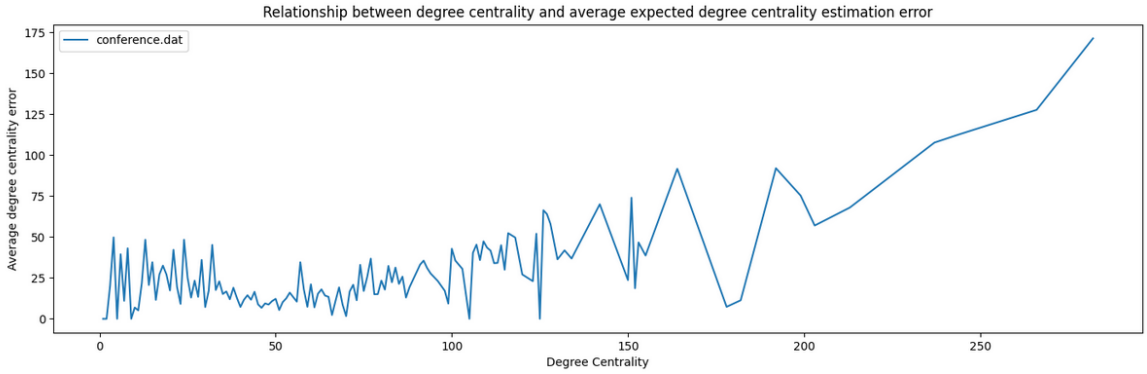


Figure 5.10: Dataset: *conference*. Heuristic: *HE0[3]*. The plot describes the relationship between degree centrality and average degree centrality error.

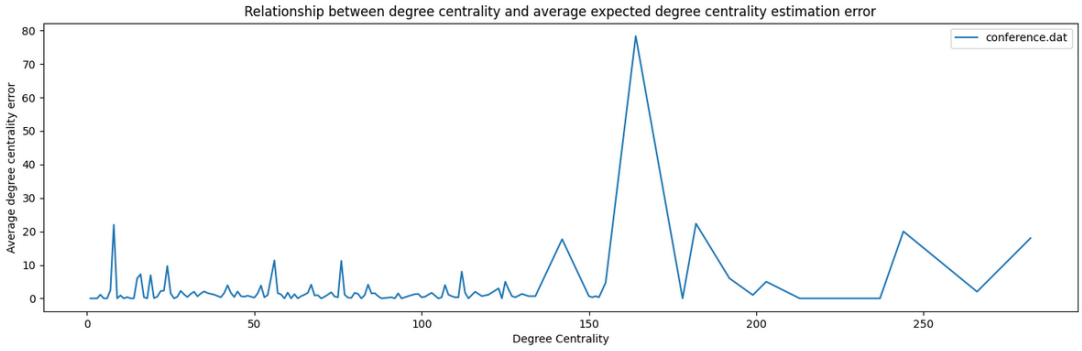


Figure 5.11: Dataset: *conference*. Heuristic: *HE2[3]*. The plot describes the relationship between degree centrality and average degree centrality error.

6 Conclusion and future work

Over the years, the research community has proposed several methods with the aim of solving the graph summarization problem. Graph summarization has various benefits which include for example reduction of data volume and storage or speedup of graph algorithms and queries. Given its advantages, graph summarization has extensive applications, including clustering, classification, community detection, outlier detection, pattern set mining, finding sources of infection in large graphs, and visualization, among others. Despite the success and the wide applicability of summarization in classical pairwise networks, we lacked a generalization of this concept to the framework of the higher-order networks represented by hypergraph. In other words, to the best of our knowledge, there is no prior practical solution to the problem of hypergraph summarization. In this thesis we introduced a first algorithmic solution to the problem of hypergraph summarization. For this purpose, we extended a node aggregation-based graph summarization method to the case of higher order interactions implemented with the hypergraph mathmatical formalism. At the hearth of our algorithm is the condensation of hypergraph nodes into summarized supernodes. We have identified as proper supernode candidates the fundamental building blocks of networks: higher-order motifs. What is more, in order to face the uncertainty which results from the deletion of some information from the original data structure, we proposed formal probabilistic semantics for evaluating structural queries on hypergraph summary. Motivated by the above, we also studied the problem of finding a "good" hypergraph summary. Finally, we proposed the implementation of different heuristics to reshape our greedy solution to perform better according to the different real world dataset peculiarities.

We are confident our proposed solution can open exiting new directions for applications in a number of domains, pushed by the growing need of analyzing higher-order interacting systems.
First of all it would be interesting to test the performance of our solution on further datasets. Probably this would lead to novel greedy heuristics in order to improve the results.
Our node aggregation-based solution can be extended to new approaches which do not take into consideration higher-order network motifs to condense nodes into supernodes. Furthermore, we believe that other graph summarization methods can be extended to the higher order domain.
An interesting research direction is to study the novel problem of hypergraph visualization and the benefits of summarization in this regard.
Moreover, in this work we limit ourselves considering only static hypergraphs. However, in the real world data collections are almost never static, new nodes and new edges can be added any time. It would be useful to understand how our summary can handle dynamic insertions and dynamic deletions of hyperedges and nodes.
The proposed solution does not guarantee output quality. In several applications it is important to be able to rely on quality guarantees. In this regard we suggest to approach this problem starting from the method proposed by Riondato et al. [22].

Bibliography

[1] P. W. Anderson. More is different. *Science*, 1972.

[2] F. Bacchus, A. Grove, J. Halpern, and D. Kohler. From statistical knowledge bases to degrees of belief. *A.I.*, 1996.

[3] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x? anonymized social networks, hidden patterns, and structural steganography. *WWW*, 2007.

[4] Federico Battiston, Giulia Cencetti, Iacopo Iacopini, Vito Latora, Maxime Lucas, Alice Patania, Jean-Gabriel Young, and Giovanni Petri. Networks beyond pairwise interactions: structure and dynamics. *Physics Reports*, 2020.

[5] Philip S Chodrow. Configuration models of random hypergraphs and their applications. *arXiv:1902.09302*, 2019.

[6] Paul Erdős. On random graphs i. *Publ. Math.*, 1959.

[7] M. Hay, G. Miklau, D. Jensen, and P. Weis. Resisting structural re-identification in anonymized social networks. *VLDB*, 2008.

[8] Danai Koutra, U. Kang, Jilles Vreeken, and Christos Faloutsos. Vog: Summarizing and understanding large graphs. *Proceedings of the SIAM international Conference on Data Mining (SDM'14)*, 2014.

[9] Vito Latora, Vincenzo Nicosia, and Giovanni Russo. *Complex Networks Principles, Methods and Applications*. Cambridge university press, 2017.

[10] Kristen LeFevre and Evimaria Terzi. Grass: Graph structure summarization. *Proceedings of the SIAM International Conference on Data Mining (SDM'10)*, 2010.

[11] Jonathan M Levine, Jordi Bascompte, Peter B Adler, and Stefano Allesina. Beyond pairwise mechanisms of species coexistence in complex communities. *Nature*, 2017.

[12] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. Graph summarization methods and applications: A survey. *arXiv:1612.04883v3*, 2018.

[13] Quintino Francesco Lotito, Federico Musciotto, Alberto Montresor, and Federico Battiston. Higher-order motif analysis in hypergraphs. *Commun Phys*, 2022.

[14] Yucheng Low, Danny Bickson, Joseph Gonzalez, Carlos Guestrin, Aapo Kyrola, and Joseph M. Hellerstein. Distributed graphlab: A framework for machine learning and data mining in the cloud. *Proc. VLDB Endow*, 2012.

[15] Antonio Maccioni and Daniel J. Abadi. Scalable pattern matching over compressed graphs via dedensification. *In Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'16)*, 2016.

[16] Yasir Mehmood, Nicola Barbieri, Francesco Bonchi, and Antti Ukkonen. Csi: Community-level social influence analysis. *Machine Learning and Knowledge Discovery in Databases*, 2013.

[17] Ron Milo, Shalev Itzkovitz, Nadav Kashtan, Reuven Levitt, Shai Shen-Orr, Inbal Ayzenshtat, Michal Sheffer, and Uri Alon. Superfamilies of evolved and designed networks. *Science*, 2004.

[18] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: Simple building blocks of complex networks. *Science*, 2002.

[19] A. Narayanan and V. Shmatikov. De-anonymizing social networks. *IEEE Symposium on Security and Privacy*, 2009.

[20] Saket Navlakha, Rajeev Rastogi, and Nisheeth Shrivastava. Graph summarization with bounded error. *Proceedings of the ACM Special Interest Group on Management of Data (SIGMOD'08)*, 2008.

[21] Mark E. J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Phys. Rev.*, 2004.

[22] Matteo Riondato, David García-Soriano, and Francesco Bonchi. Graph summarization with quality guarantees. *Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM'14)*, 2014.

[23] J. Rissanen. Modelling by the shortest data description. *Automatica*, 1978.

[24] JElad Schneidman, Susanne Still, Michael J Berry, and William Bialek. Beyond pairwise mechanisms of species coexistence in complex communities. *Phys. Rev. Lett.*, 2003.

[25] Z. Shen, K.-L. Ma, and T. Eliassi-Rad. Visual analysis of large heterogeneous social networks by semantic and structural abstraction. *IEEE Trans. Vis. Comput. Graph*, 2006.

[26] The opte project. https://www.opte.org/.

[27] Stanley Wasserman and Katherine Faust. Social network analysis : Methods and applications (structural analysis in the social sciences). *Cambridge University Press*, 1994.

[28] Sebastian Wernicke. Efficient detection of network motifs. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 2006.

[29] Internet from wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Internet.

[30] www.sci.unich.it (university of chieti-pescara). https://www.sci.unich.it/.

[31] Quan Xiao. Node importance measure for scientific research collaboration from hypernetwork perspective. *Teh. Vjesn.*, 2016.

[32] Jaewon Yang and Jure Leskovec. Overlapping community detection at scale: A nonnegative matrix factorization approach. *Proceedings of the 6th ACM International Conference on Web Search and Data Mining (WSDM'13)*, 2013.

Attachment A Algorithms

In this attachment we report the pseudocode of the algorithms which are discussed in the thesis.

Algorithm 1 Counting higher-order motifs of order 3

Input: a hypergraph $H = (V, E)$
Output: distribution of the frequency of the motifs of order 3
Let M be the motifs frequency hash map
for all hyperedge e of order 3 in E **do**
  $V^* \leftarrow$ vertices of e
  $motif \leftarrow \emptyset$
  for $e^* \in \mathcal{P}(V^*)$ **do**
    if $e^* \in E$ **then**
      $motif \leftarrow motif \cup e^*$
    end if
  end for
  Let $\mathcal{C}_m$ be the isomorphism class of $motif$
  $M[\mathcal{C}_m] += 1$
end for
$G \leftarrow$ Discard all hyperedges of order 3 from H
$S \leftarrow \text{ESU}(G, 3)$
for all $s = (V^*, E^*) \in S$ **do**
  if V^* not already visited **then**
    Let $\mathcal{C}_m$ be the isomorphism class of s
    $M[\mathcal{C}_m] += 1$
  end if
end for

Algorithm 2 Counting higher-order motifs of order 4

Input: a hypergraph $H = (V, E)$
Output: distribution of the frequency of the motifs of order 4
Let M be the motifs frequency hash map
for all hyperedge e of order 4 in E **do**
  $V^* \leftarrow$ vertices of e
  $motif \leftarrow \emptyset$
  for $e^* \in \mathcal{P}(V^*)$ **do**
    if $e^* \in E$ **then**
      $motif \leftarrow motif \cup e^*$
    end if
  end for
  Let $\mathcal{C}_m$ be the isomorphism class of $motif$
  $M[\mathcal{C}_m] += 1$
end for
$H \leftarrow$ Discard all hyperedges of order 4 from H
for all hyperedge e of order 3 in E **do**
  Let $\mathcal{N}$ be the set of hyperedges adjacent to e
  for all hyperedge n in $\mathcal{N}$ **do**
    $V^* \leftarrow$ vertices of $e \cup n$
    if $|V^*| = 4$ and V^* not already visited **then**
      $motif \leftarrow \emptyset$
      for $e^* \in \mathcal{P}(V^*)$ **do**
        if $e^* \in E$ **then**
          $motif \leftarrow motif \cup e^*$
        end if
      end for
      Let $\mathcal{C}_m$ be the isomorphism class of $motif$
      $M[\mathcal{C}_m] += 1$
    end if
  end for
end for
$H \leftarrow$ Discard all hyperedges of order 3 from H
$S \leftarrow \mathrm{ESU}(G, 4)$
for all $s = (V^*, E^*) \in S$ **do**
  if V^* not already visited **then**
    Let $\mathcal{C}_m$ be the isomorphism class of s
    $M[\mathcal{C}_m] += 1$
  end if
end for

Algorithm 3 Hypergraph summarization baseline algorithm

Input: Hypergraph $H = (X, E)$
Output: Summarized hypergraph S

#Execute higher-order motif analysis as explained in Chapter 3
for all higher-order motif $motif$ in H **do**
  #Check that the nodes in $motif$ have not already been assigned to a supernode
  #If the target nodes do not belong to any partition yet, we create a new supernode
  if new_partition **then**
    #supernode_category[i] indicates the category of supernode labeled i.
    supernode_category[supernode_id] = category of motif $motif$
    #supernodes[i] is the set of nodes which belongs to the supernode identified with integer i
    supernodes[supernode_id]=set()

    for all node $node$ in $motif$ **do**
      supernode_map[$node$] = supernode_id
      supernodes[supernode_id++].add($node$)
    end for
  end if
end for

#Insert all the singleton which do not belong to any supernode yet into new order one supernodes
for all node $node$ in supernode_map **do**
  if supernode_map[$node$]==-1 **then**
    Add $node$ to a new supernode with cardinality equal to one
  end if
end for

#Finally, we build summary hyperedges
for all edge e in E **do**
  if $\exists$ a node $n \in e$ which belongs to a different supernode w.r.t. the others **then**
    #We add a new hyperedge to the summary hypergraph
    hyperedge_dictionary = {} #Dictionary data structure: ids of the supernodes are the keys; intersaction profiles are the values
    for all node v in e **do**
      v_supernode = supernode_map[v]
      if v_supernode in hyperedge_dictionary **then**
        hyperedge_dictionary[v_supernode]+=1
      else
        hyperedge_dictionary[v_supernode]=1
      end if
      summarized_hyperedges.append(hyperedge_dictionary)
    end for
  end if
end for