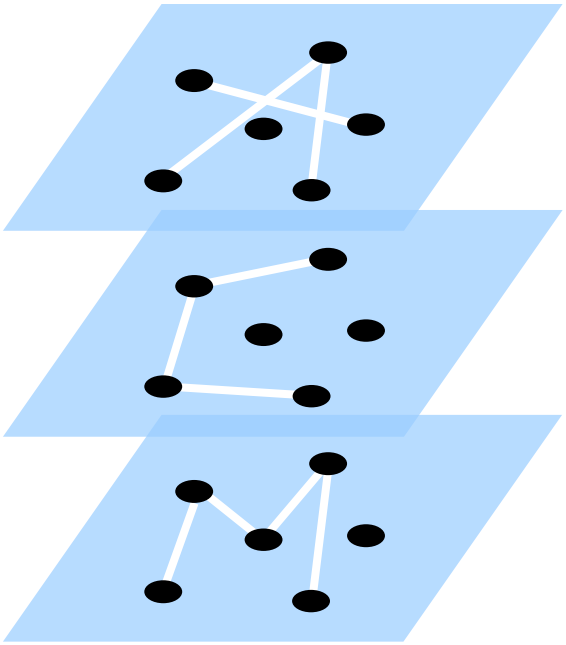
U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

SUBGRAPH PATTERNS IN MULTIPLEX NETWORKS

André Couto Meira

Master Thesis presented to
Faculty of Sciences of the University of Porto in
Network and Information Systems Engineering
2019



SUBGRAPH PATTERNS IN MULTIPLEX NETWORKS

André Couto Meira
Network and Information Systems Engineering
Computer Science Department
2019

Supervisor
Pedro Manuel Pinto Ribeiro, Assistant Professor
Faculty of Sciences, University of Porto

U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

to Maria and all my family

Resumo

As redes são uma estrutura genérica que permitem representar uma grande diversidade de sistemas. Elas são modeladas por grafos matemáticos, que têm a capacidade de conter muitas informações na sua topologia, mas esquematicamente são apenas entidades que de alguma forma estão conectadas entre si. Essa simplicidade explica o porquê de existirem muitas áreas de aplicação uma vez que as conexões são frequentes no mundo real. No entanto, analisar essa arquitetura não é simples. Mesmo que a presença de certas entidades revele informação, é pouco comparado com o que pode ser descoberto se os relacionamentos forem também considerados.

A presença de um subgrafo numa rede, e com que frequência ele aparece, pode ser indicativo de algo extremamente importante. Por esse motivo, a identificação de padrões, e outras características, são áreas relativamente bem exploradas nos últimos anos e mostra que será uma área que continuará a crescer. No entanto, este problema de identificação e contagem de subgrafos, algoritmicamente falando, é um obstáculo difícil sem uma solução polinomial conhecida. Isso faz com que a maioria dos trabalhos combata o problema na forma mais simples, ignorando características que podem e devem começar a ser consideradas.

Camadas são um novo aspeto que tem sido adicionado à estrutura. Estudar e adquirir conhecimento sobre redes com essas características especiais é o principal objetivo desta tese, principalmente através da deteção e contagem de subgrafos. No entanto, tudo o que conhecemos até agora é diferente neste novo contexto, mesmo um simples subgrafo. Portanto, o trabalho realizado deve ser aprimorado para acompanhar a evolução. Após um estudo cuidadoso dos mais diversos métodos de análise de redes, este projeto definiu a sua abordagem através da implementação de alguns conceitos matemáticos importantes e adaptando métodos clássicos.

Após analisar e mostrar algumas técnicas e ideias, que consideramos necessárias para este trabalho, descrevemos em detalhe como obtemos os nossos próprios resultados. Primeiro, contribuímos para a definição de conceitos numa área nova. Uma agregação para identificar subgrafos, voltando atrás para uma redução que permite identificar classes isomórficas é o resumo da abordagem. Permite-nos descobrir padrões que são mostrados com uma poderosa ferramenta de visualização que nós desenvolvemos. Também apresentamos os nossos tempos de execução e avaliamos o nosso processo em redes sintéticas que seguem modelos tradicionais adaptados.

Palavras-chave— multiplex, redes, subgrafos, padrões, motifs

Abstract

Networks are a generic structure that allows representing a great diversity of systems. They are modeled by mathematical graphs, which have the capacity to contain a lot of information in their topology, but schematically they are only entities that somehow are connected to each other. This simplicity explains why there are many areas of application since connections are frequent in the real world. However, analyzing this architecture is not simple. Even if the presence of certain entities reveals information, it is very little related to what can be found if the relationships are considered.

The presence of a subgraph in a network, and how often it appears, may be indicative of something extremely important. For this reason, the identification of patterns, and other characteristics, are relatively well explored areas in recent years and shows that it will be an area that will continue to grow. However, this problem of identification and counting of subgraphs, algorithmically speaking, is a hard obstacle without a known polynomial solution. This causes most of the works to fight the problem in its simplest form ignoring features that the networks present and that can and should start to be considered.

Layers are a new aspect that is being added to the structure. Studying and gaining knowledge about networks with these special features is the main goal of this thesis, especially by detecting and counting of subgraphs. However, everything as we know it so far is different in this new context, even a simple subgraph that is the basis of this work. Therefore, the work done should be improved to follow this evolution of systems. After carefully studying the most diverse methods of network analysis this project defined its new approach by implementing some important mathematical concepts and adapting other methods.

After going through and showing some techniques and ideas, which we consider necessary for this work, we describe in detail how we get our own results. First, we contribute to defining concepts in a novel area. An aggregation to identify subgraphs followed by a backtrack to make a reduction for identify isomorphic classes is the summary of our own approach. It allows us to obtain and discover patterns, which we show in our results section, created with a powerful visualization tool that we have developed. We also present our execution times and we evaluate our process in synthetic networks through classical models adapted to contain layers.

Keywords— multiplex, networks, subgraphs, patterns, motifs

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Around the world there are complex structures that, for various reasons, need to be understood. The composition itself can give important evidence, but certainly it is a small fraction of what can be extracted. Therefore, these complex structures need a model representation capable of embracing all their details so that they can then be studied in detail. Graphs are a good abstract model that has been thoroughly studied, but there are still many open research problems.

1.1 Context

Many systems, from the most diverse areas, can be represented as networks which increases the relevance of the network concept. This kind of organization, with relations being formed, can be found easily. Perhaps we can only perceive it in a very limited set of systems, but if we look around, we can find connections between a lot of entities. The concept is easily detected in social networks, but they are everywhere [9]. The model is used in many fields, including sciences like chemistry and biology [44]. It is also used to represent networks based on references, like publications or web pages [57], and others, not so obvious, like the human brain [10]. Transport systems are another one where the concept of network is easily applicable [24] as in Figure 1.1.

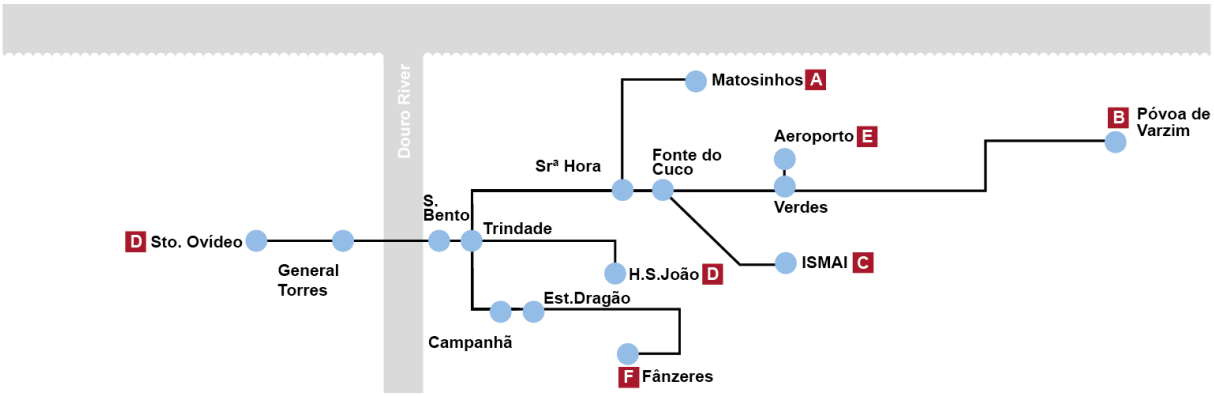


Figure 1.1: This map represents all the Metro lines in the city of Porto, from A to F, with some of the stations indicated as blue circles and connections as black lines.

Graphs represent networks and they are simple structures with a very rich characterization power on their topology. They are an abstract model to represent interactions between individual components and these can contain as much data as necessary. In addition to the nodes and their categories, there is other information, like the interactions and their types, that can tell us more than if two units have a simple connection between them. Studying it as connected components and not as individual elements allows finding the knowledge that may not be perceived at first.

Extracting information from graphs is an area known as *graph mining*, and the crucial importance of this area can be explained by the wide variety of systems that are representable in this way. Therefore, there is an attempt to extract information on various aspects. One core and important primitive is to detect the presence of a subgraph in a graph and also count the number of its occurrences. This is called a *subgraph census* and this is the basis of other important measures. In particular, finding patterns that appear in the network more times than expected.

Each network can have a specific pattern, or a set of patterns, which can give us information about what unusual things happen between the entities. We can understand unusual as something that appears more than the normal, a *network motif*, or something else that does not appear so frequently, an *anti-motif*. The patterns can be seen as a characteristic that identifies the type of the network [35] and this can be important for tasks ranging from health to financial area such as identifying diseases [13] and this is the reason why they are being used to study the networks.

The evolution of the networks means that the research done so far does not fulfill the expectations. There are new features that come up with data growth that are being ignored. Or, characteristics that already existed but were not analyzed due to the not advanced state of the art. That is why it is urgent to think of new approaches to deal with the new demands of networks. Even if the solution is an adaptation of old methods it is important to advance to make sure that the study of the networks is not stopped and, fortunately, that is not happening.

1.2 Research Goals

Discovering all subgraphs in a network and identifying which are different from each other in order to get the frequencies can be crucial tasks. Despite their importance, these tasks are very limited by computational problems. Because of this, usually the search is focused on reduced sizes in networks that have a large number of entities connected between them. This problem, of counting subgraphs in networks, is a computationally hard task closely related to the *subgraph isomorphism problem*, which is known to be NP-complete for general graphs [22]. To further increase the difficulty of the problem, we are interested in a set of subgraphs, and not just one specific subgraph. This is because in the general case we do not know before analyzing which subgraphs are present and we are interested in all of them because anyone can be surprising. Although the problems still exist, they have been reduced because of the research that has been done in past years. However, almost all studied networks are represented as simple as possible where the connections are single, static and unweighted and the entities are all of the same type.

The focus until now is essentially on the simpler networks, which we call traditional or *classic* networks. Nowadays, complex systems have been enriched with different features. The entities can belong to different types and they connect to others in relationships that can also be categorized. **The networks can be divided into layers**, with connections between them, and all of this can change in time. The evolution brings the urgency for adaptations and all should be considered because an aggregation to analyze it as a classic network leads to information loss.

A new form to represent the networks with more detail is a set of layers, which means a kind of division of the network. **We will study this new characteristic**. The analysis of multilayer networks is more than simply analyzing it in each layer with the tools that already exist and then make individually conclusions. We are not interested in the subgraphs present in each layer but in the subgraphs present in the complete network that is not an aggregated one. Actually, to get metrics related to subgraphs, first it is necessary to define what they are in this new context.

New definitions need to be made to concretize the subgraph frequencies and then identify motifs, but since this is a very recent area, many of them are still open. Because of that, we have taken a calculated step and considered a specific type of multilayer network where the edges between layers are very limited, when not ignored, and the set of nodes is the same in each layer. The work done in this field is still scarce and very recent [8, 14, 15, 29, 53] and our main goal is to contribute to this promising new area **identifying and getting frequencies of subgraphs**.

The research becomes more interesting **knowing that a real application can be found**. It is possible if we think that some places have more than one mode of public transportation, as in Figure 1.2. Exchanging transports, which might mean moving to another layer, may have a cost that could be considered if it is significant. For example, in a certain station it can takes time to walk from a metro platform to a train. Some recent studies take into account the presence of multiple layers in transport systems [2], but not for analysis of subgraphs as we do in our work.

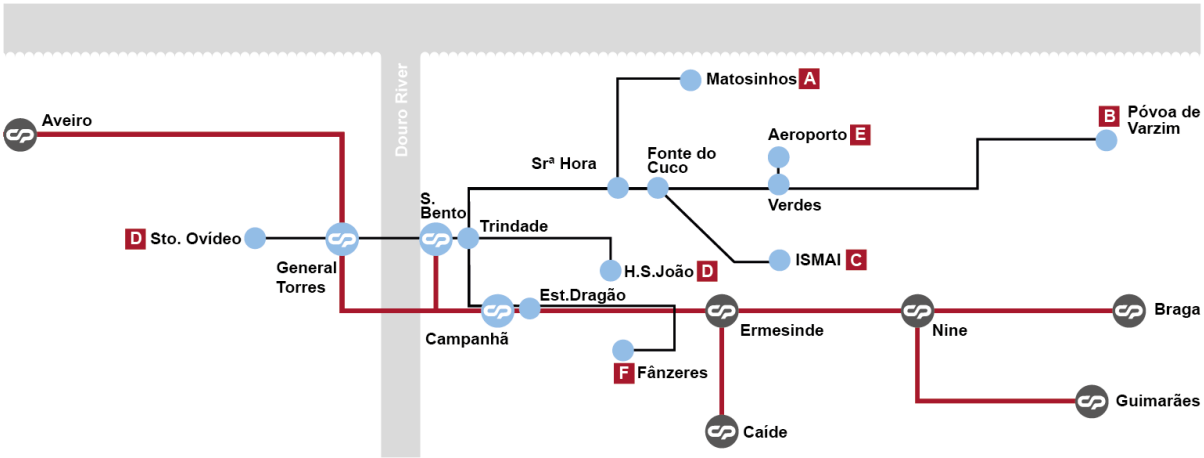


Figure 1.2: A multilayer network for urban transports. One layer corresponds to metro connections (black lines) and the other to train (red lines). Other layers like bus could be added. The only nodes, or entities, represented in the figure are the stations that are at the end of a line, the ones that allow to diverge for different lines and those that have both connections of metro and train.

1.3 Main Contributions

This project is centered in a special type of multilayer network. We are interested in understanding better the information in a multiplex network and according to Kivelä et al. they are "as edge-colored multigraphs, which are networks with multiple types of edges" [30]. This simplification is possible if we consider that each layer corresponds to a certain color. Each connection assumes the type that corresponds to the layer and the connections between the nodes in the different layers, the sets are assumed to be equal, are very limited or totally ignored.

This transformation allows us to advance in solving the problem because it simplifies and brings it closer to networks that are already known and studied. One of our contributions was to develop an entire approach based on the development of new methods with the help of classical algorithms, which we adapted to fulfill our purpose, and the application of mathematical theories. Therefore, **we can assume as our the complete strategy for analyzing multiplex networks**, in particular the search for subgraphs, even using some ideas and adaptations from other authors.

Identification of subgraphs is not possible if we do not know what a subgraph is in this new context. Therefore, **we had the need to redefine and clarify some concepts**. Another example is the isomorphism definition that must be expanded to contemplate the new dimension.

Find subgraphs is the first step when it comes to patterns and only then will come the exact frequencies and the identification of motifs. Therefore, the predominant focus of this thesis is **being able to count subgraphs in multiplex networks**, which has been widely achieved.

Moreover, we are able to show the subgraphs found in an explicit and clear way, by drawing each one using layers or choosing a compact version. The way we present the results allows anyone who wants to use the **visualization tool to easily interpret the subgraphs found**.

Finding motifs after counting on the original network requires generating random networks that are as similar as possible to the original. This allows us to detect what should be considered as normal since if a large number of networks are generated then we can safely know what is expected. We present a **novel method for generating random multiplex networks**, which is fully idealized for layers. Applying the methods of classic networks to each layer separately is not enough because edges may overlap and it is not considered generating little similar networks.

Being in a recent area, we struggle with problems that would not happen in other well studied fields, like the divergence of some concepts in the state of the art. Furthermore, the few real networks already created led to the need to **adapt classic network models with the layers**.

The demand to get a set of networks was to be able to **obtain experimental results**. Our focus was to compare the differences between the algorithms used and realize how scalable our approach is. Synthetic networks were fundamental because they allow to use different parameters.

Still not done, but our intention is to **make our source code available** as soon as possible.

1.4 Thesis Outline

This thesis is structured in five major chapters. A brief description for each one is presented below.

Chapter 1 - Introduction. Provides a brief introduction to the research area, the context of this topic, our research goals, our contributions, as well as the organization of the thesis. In between, we provide two network images of our daily lives, where the second one is multilayer.

Chapter 2 - Preliminaries. Introduces a common graph terminology that will be used throughout the thesis. In particular, apart from classic networks, it presents the necessary definitions of a multiplex network and the problems related to subgraphs in both contexts. In addition, an overview of the work done in this area that is important to complete our approach.

Chapter 3 - Approach. Establishment of the approach used to achieve the goals. Detailed explanation of the methodology used through images and also some pseudo code. Two very similar paths are followed, but each one has three fundamental steps to find subgraphs that are explained in detail. Furthermore, our method to show the results in a way that is visually clear.

Chapter 4 - Experimental Results. The results obtained by each method in a diverse network set are detailed and discussed. In particular a complete section with random networks followed by real networks. Visual results are also shown with interesting subgraphs found.

Chapter 5 - Conclusion and Future Work. A perspective of the work done is given and also what should be done to continue the work, with considerable improvements, and how to expand this area.

Chapter 2

Preliminaries

In this chapter we present several concepts that are essential to better understanding others. The notation used will be the same throughout the document. A network is defined using a mathematical graph and this is the central subject of this section. Graph theory is a well explored area, but the concept of multilayer and all specifications for this field is a little more uncertain. For this reason, an overview is given that covers as much as possible.

2.1 Graph Terminology and Concepts

2.1.1 Single Layer

A graph G is composed by a tuple $G = (V, E)$ where V is a set of vertices or nodes and E is a set of edges connecting pairs of nodes. Each pair has the format (u, v) and $u, v \in V(G)$, so these two nodes are the endpoints of the edge in $E(G) \subseteq V(G) \times V(G)$. The size of the graph is considered the size of the set $V(G)$ and it is denoted by $|V(G)|$. A *k-graph* is a graph of size k.

The *degree* of a node u is the number of connections it has to other nodes, and that means the number of times there is an edge in $E(G)$ where one of the elements of the pair is the element u. The *neighborhood*, denoted as $N(u)$, is composed by the set of nodes that share an edge with u, and they are also called adjacent nodes of u.

In *undirected* graphs the order is irrelevant since the nodes are connected in both directions, but in *directed* graphs the edges are ordered pairs. Because of that, the definition of node degree is divided in two. The *indegree* of u is the quantity of pairs of type $(u, \_) \in E(G)$. Similarly, the *outdegree* is the amount of edges where u is the second element of the pair.

A *subgraph* G_k of a graph G is a k-graph in which $V(G_k) \subseteq V(G)$ and $E(G_k) \subseteq E(G)$. This subgraph is said to be *induced* when $\forall u, v \in V(G_k)$: $(u, v) \in E(G)$ implies $(u, v) \in E(Gk)$ and it is called *connected* if all pairs of nodes are connected by some sequence of edges. The *neighborhood* of a subgraph G_k, denoted by $N(G_k)$ is the union of $N(u), \forall u \in V(G_k)$.

2.1.2 Multiple Layers

A system consisting in multiple levels needs a special representation, more robust than the normal, able to append layers to nodes and edges. We will present the most general notation, capable of defining any type of network, and then specify the one that is the base of our project. The most general concept, based on Kivelä et al. work [30], "allows each node to belong to any subset of the layers" and allows connections between nodes of any layers.

A multilayer network is defined as a quadruplet $M = (V_M, E_M, V, L)$. Starting by the end of the tuple, the network needs a set of layers that is represented by the letter L, and, just like a classic network, it needs a set of nodes V. But since not all nodes have to be present in a layer, as well as edges, information given by the sets V_M and E_M is required.

The simple idea of having one single set of layers is not enough to the most general case because that limits the set of layers. An *aspect* is considered a dimension of the network. For example, in a network where one of the aspects represents the type of the network, and the other represents the time, we would need two sets of layers. To avoid confusion, the term *elementary layer* is used for an element that is in these sets, so it is possible to represent the network with several sets of elementary layers (any number d of aspects). The term *layer* is to refer to a combination of elementary layers - one of each aspect. We can define the sequence $L = \{L_1, \cdots, L_d\}$ as a sequence of sets of elementary layers where each L_i can be $(\alpha_1, \alpha_2, \alpha_3, \cdots)$.

The presence of a node in an elementary layer is possible by combining the set of nodes V and the set of elementary layers $L_1 \times \cdots \times L_d$. The set V_M is composed by the node-layer tuples in $V \times L_1 \times \cdots \times L_d$ such that the node has a copy in the elementary layer. The term *node-layer* is used often and $(u, \alpha_1, \cdots, \alpha_d)$ represents node u on layer represented with d aspects $(\alpha_1, \cdots, \alpha_d)$. The connections are not anymore between nodes but between node-layers tuples because each edge can start and end in different layers. Once this information is contained in the node-layers tuples the edge set E_M is formed by $V_M \times V_M$, a set of pairs of pairs.

A multilayer network M is considered undirected if for all edges of the type $((u, \alpha), (v, \beta)) \in E_M$ it is also true that $((v, \beta), (u, \alpha)) \in E_M$. The graph is directed if this does not happen at least once. It is important to separate the edges that connect nodes in different layers and those that connect a pair that are in the same layer. Considering an edge $((v, \alpha), (u, \beta))$, *intra-layer edges* occur when $\alpha = \beta$ and *inter-layer* when $\alpha \neq \beta$. For the purposes of our project, self-edges are disallowed by requiring that $((u, \alpha), (u, \alpha)) \notin E_M$. The concept of *subgraph*, as well as *induced* and *connected*, are maintained with the difference of having node-layers instead of only nodes.

Kivelä et al. made the observation: "The first two elements in a multilayer network M yield a graph $G_M = (V_M, E_M)$, so one can interpret a multilayer network as a graph whose nodes are labelled in a certain way". This conclusion makes possible to predict that some specific multilayer networks will have a similar representation as using a classic single layer network. This mapping can be seen in Figure 2.1 and in particular a multiplex network, which we discuss below, are represented as well as its conversion.

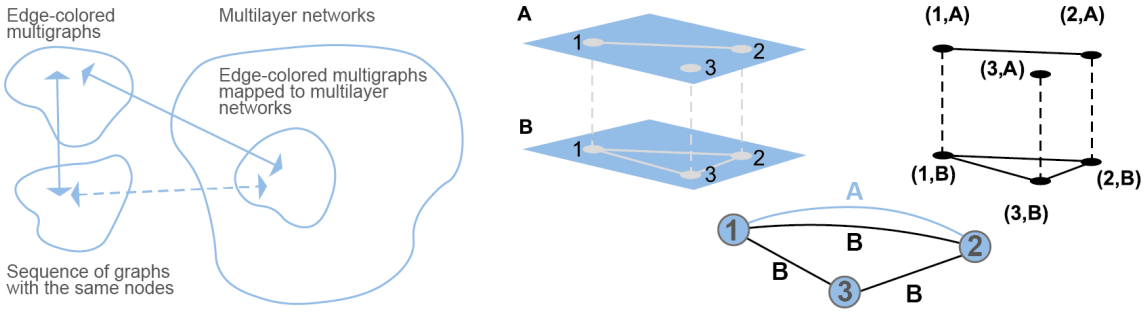


Figure 2.1: Illustration of the idea of characterizing different types of network by finding 'natural' injective maps from one set of network structures to another. A multiplex network, its underlying graph $G_M = (V_M, E_M)$ and the correspondent edge-colored multigraph. Adapted from [30].

There are different specifications of multilayer and the difference between them arise from the application of some constraints. One of them is the *node-aligned* and means that all the layers contain all nodes making $V_M = V \times L_1 \times \cdots \times L_d$. Sometimes, this concept is forced, to easily represent the network, using the *padding* method that adds empty nodes [16]. When the number of aspects is zero the sequence L does not exist and the product $V \times L_1 \times \cdots \times L_d$ is impossible which leads to $V_M = V$. Because of that there will be two equal sets in the quadruplet and one of them becomes redundant. Moreover, the set L is empty and unnecessary. When this happens the multilayer network M reduces to a single layer network $G = (V, E)$. When the number of aspects is one and the concept of node-aligned is present redundancy also occurs. Considering k to be the size of L_1 and $(\alpha_1, \cdots, \alpha_k)$ the elementary layers, V_M is the combination $V \times L_1$ but this information is not necessary since all nodes are present and each one will form k tuples. In this case, a network can be defined as a triplet $M = (E_M, V, L)$ where $L = L_1$ and the concept of elementary-layer becomes the same as layer.

The edges can have also one or more constraints. Two node-layer tuples (u, α) and (v, β) are connected if and only if $u = v$. That is, the two node-layer represent the same node in different layers, which we call *coupling edges*. The other option is when $\alpha = \beta$ and we are talking about connections occurring in the same layer. A network that only contains this type of connections is an intra-layer network. Connections to other layers can be omitted assuming only a relation between copies of nodes. In this case, the representation of the edges can be different, instead of two node-layers $((u, \alpha), (v, \alpha))$ we can have another triplet (u, v, α) with the same information. With this simplification, it is easy to perceive the parallel that exists between the representation of the edge-colored graphs.

The definition of an edge-colored multigraph is a triplet $G = (V, E, C)$, where V is the node set, C is the color set (which is used for assign a type to each edge) and E, the edge set, is contained in $V \times V \times C$. Forcing a node-aligned representation and presence of all coupling edges or none, makes the system a *multiplex network* and the similarities with colored edges are found considering that the set of layers L assumes exactly the same function as the set of colors C.

2.1.3 Representation

2.1.3.1 Adjacency Matrices

Given a graph G it is possible to associate it to the adjacency matrix denominated as $A(G) = G_{Adj}$, with size $|V| \times |V|$, and $G_{Adj}[u][v]$ stores a value from the indicator function. This value can be different according to the type of graph. Since the graph is representing a network N we can conclude that $A(N) = A(G) = G_{Adj}$. The values returned from the indicator function, with arguments u and v indicating the index of the nodes, can be boolean and 1 occurs when $(u, v) \in E(G)$ and is 0 otherwise. The value may also represent the weight of the edge or some label/color to identify the type of the connection. If self-edges are disallowed all the values in the diagonal, from the top left corner to the bottom right corner, are zeros. This diagonal divides a matrix and both parts are equal when the graph is undirected, making a symmetric matrix.

A multiplex network can be expressed as a combination of intra-layer adjacency matrices. Once the network is node-aligned there are no representation problems. Technically, the node-aligned constraint is not a need in the real network because this feature can be guaranteed by adding nodes that are not adjacent to any other. The method is called *padding* and requires an extra attention when studying the results. This information, that can lead to misunderstandings, can be solved creating a new matrix of participation. The size required is $|V| \times |L|$ and in position $[u][\alpha]$ there is a value 1 if the node u participates in the layer α, and 0 otherwise. Assuming that the representations of the same node are always connected to each other, and that these are the only ones that happen between layers, no representation is required for this type of edges. However, they could be represented as an additional matrix of size $|V \times L| \times |V \times L|$, transforming this structure into the same one that is presented in section 2.1.3.2.

2.1.3.2 Supra-Adjacency Matrix

This special type of matrix is popular for representing multilayer networks. This data structure is constructed by concatenating the intra-layer and the inter-layer adjacency matrices. In other words, there will be L lines, each with as many rows as nodes in layer L_i. In position i will be present the intra-layer adjacency matrix and the other positions have matrices storing the connections between the layer L_i and all the others. In this representation, a line or a column can be seen as the complete set of nodes contained in a layer. A new symbol could be used to represent the pair (u, α). The supra-adjacency matrix that represents a multiplex network, node-aligned and with all the coupling edges, yields a block-diagonal structure. The diagonal contains the intra-layer adjacency matrices and all the other blocks correspond to the couplings, with a special format, where only the diagonal is filled, forming identity matrices. Each block of the construction for the general multilayer networks are not necessarily square matrices which allows to represent the absence of nodes. The size needed to the representation is $|V|^2 \times |L|^2$.

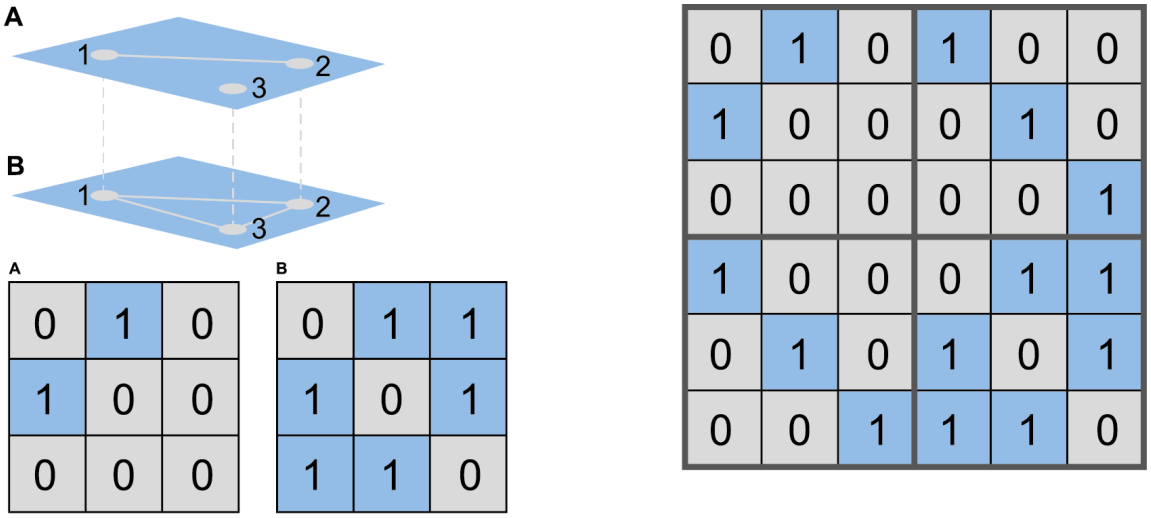


Figure 2.2: A multiplex graph and its correspondent adjacency matrices for each layer and supra-adjacency matrix.

2.1.3.3 Adjacency Lists

The choice of representation affects both the storage and computational time of algorithms to be used next. The methods seen above have a big problem of space. The vast majority of real-world network models are very sparse (they typically have a few connections of the total possible connections for each node) [5]. For sparse graphs an adjacency list [25] would be preferable, because the matrix will contain a lot of zeros. Briefly, in graph theory, an adjacency list is the representation of all edges as an array of list of neighbors for each node [49].

Consider a classic single layer graph. For each node $u \in V$ it is possible to construct a list containing all nodes v such that $(u, v) \in E$. Such list is called an adjacency list for node u. Each node has its own list, and a combination of all the lists is called an adjacency list of G. If the graph is undirected, each edge (u, v) is represented twice. If G is directed, each edge (u, v) is represented exactly once - node v appears in the adjacency list of node u.

The adjacency list of a classical graph has a much smaller memory footprint than the adjacency matrix ($|V| + |E|$ vs $|V|^2$), although this difference may decrease as a graph becomes denser. Another advantage is that adjacency list allows us to traverse the list of neighbors of node u in $\mathcal{O}(|neighbors(u)|)$ (compared with $\mathcal{O}(|V|)$ when using matrices), which is much more efficient for some algorithms that will be presented in the following chapter. When memory allows, one can even have both data structures (lists and matrices) in memory and use one or the other depending on the need graph operation.

Regarding multilayer networks, essentially any previously described matrix based graph representation can be transformed into an adjacency list by converting each matrix row into a list containing only its non-zero elements. The expected result will be a single list of node-layers tuples or a combination of lists, replicating the concept for each layer as in Figure 2.2.

Figure 2.3: A multiplex graph and its correspondent adjacency list. A list for each node instance.

2.2 Subgraph Patterns

Seeing a subgraph as a pattern allows us to think of a panoply of concepts for characterizing networks. But, first we need to define the concepts before being used together. In this case, the last goal is the identification of *network motifs* [34] and then we will discuss some past work in this area. However, it is important to keep in mind that subgraph patterns go beyond this and have other subareas such as *frequent subgraph mining* [27] and *graphlet degree distributions* [38].

2.2.1 Graph Isomorphism Problem

Isomorphism between two graphs occurs when there is a mapping between the nodes of the two graphs. That is, when there is an edge between the two nodes of a graph that edge must exist between the corresponding nodes of the other. Therefore, it consists in finding a bijection between the sets of nodes that preserves the adjacencies.

Definition 2.2.1 (Graph Isomorphism)**.** Given a graph G and a graph H, determine if it is possible to obtain H after permuting the nodes of G.

The Graph Canonization problem is very similar and consists of find a name for each graph ensuring that two equal names only occur when the graphs are isomorphic. This name is usually called *canonical labeling* and, after find it, it is enough to solve the graph isomorphism problem and that is why these two problems are connected.

Definition 2.2.2 (Graph Canonization)**.** Given a graph G and a graph H find a label for each one that are only equal when the graphs are isomorphic.

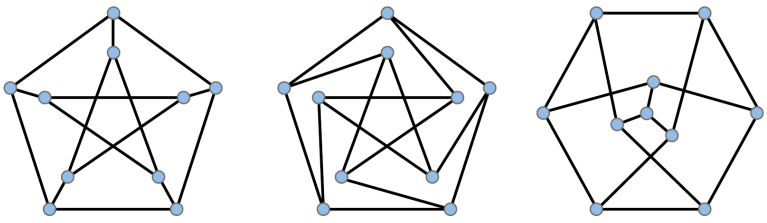


Figure 2.4: Three isomorphic graphs. The same set of ten nodes and the edges in different ways.

2.2.2 Subgraph Census Problem

Another very important related graph primitive is to count the number of times subgraphs appear in a given network. In particular, we want to know which subgraphs are present, with a certain size, and how often each appears. Although deceptively simple in its formulation, the subgraphs census concept lies at the core of several graph mining tasks.

Definition 2.2.3 (Subgraph Census)**.** Given an integer k and a graph G, determine the frequency of all different connected induced k-subgraphs in G.

In order to properly understand this concept, we also need to define what we mean by frequency. We look for differences even if they are minimal. It is as simple as checking if there are nodes or edges that are not shared between two subgraphs. All other edges and nodes can overlap. Figure 2.5 illustrates this definition.

Definition 2.2.4 (Subgraph Frequency)**.** Two occurrences of a subgraph, in a graph G, are different if they have at least one node or edge different.

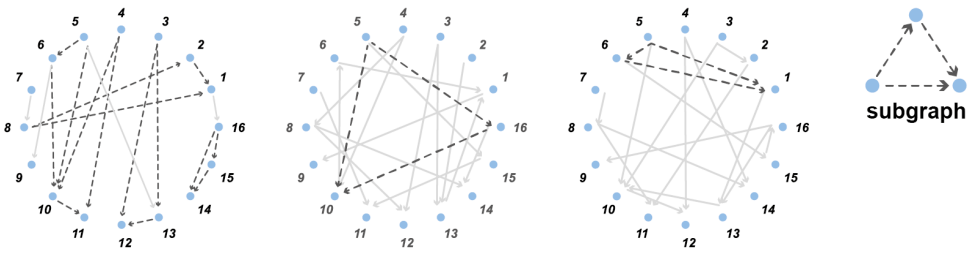


Figure 2.5: Three different networks. In the first one the frequency is five and in the others it is one. The darker and dashed lines represent each match of the pattern. Adapted from [34].

Computing a subgraph census is a computationally very hard task. As said before, even knowing if a subgraph appears at all (*subgraph isomorphism problem*) is already an NP-complete problem [22]. Naturally, determining the exact frequency is even harder. Because of this, typically this operation is limited to small subgraph sizes.

2.2.3 Network Motifs Problem

When analyzing networks and trying to discover what is characteristic of a specific graph, it is important to understand what should be *expected* as "normal". In other words, we must take care to establish a suitable *null model* that captures some features and satisfies a given set of constraints establishing what should be considered *unexpected*. In the particular case of motifs, the idea is to get the frequencies in a set of similar random networks to be the basis of comparison. The most used similarity model is to keep the degree sequence, that is, to generate networks in which all nodes preserve the same (in and out) degree [34], as exemplified in Figure 2.6. This notion can be extended to colored graphs if we maintain the colored degree sequence [42].

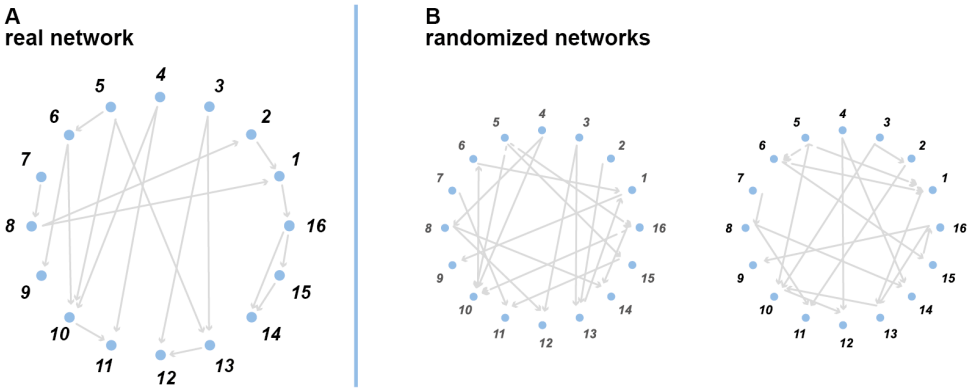


Figure 2.6: Two random directed networks preserving in and out degree. Adapted from [34].

Network motifs were initially defined by R. Milo et al. and were considered simple building blocks of complex networks [34]. Discovering them is a very important task for creating network signatures as we will see ahead because similar motifs are found in graphs representing networks of the same family.

Usually, we do not know the motifs we will find. Because of that, the typically first step consists in choosing a size k and then computing its k-subgraph census on the original network. This results in obtaining all frequencies of subgraphs of size k. Other approaches might look for a specific subgraph or a set of them.

In order to measure their over-representation, a set of similar random networks, according to some null model, is generated and a k-subgraph census is computed on each random network. After that, the frequencies of each subgraph type presented in the original network are compared with the average frequencies of the same class in the random networks.

Those that are present in significantly higher frequency than in the original network can be reported as motifs. Figure 2.7 illustrates this process. On the left we see the original network and highlighted one of the subgraphs returned by the subgraph census function. This pattern is the well-known feed-forward loop. For this pattern to be considered a motif, we compared its frequency on both the original and the set of similar random networks, the four networks on the right side, that preserve the indegree and outdegree of each node. This pattern was returned as motif because in random networks it appeared at most once whereas in the original network it occurred five times.

Definition 2.2.5 (Network Motifs)**.** Small induced subgraphs that appear in a network with a higher frequency than what would be expected, through the following parameters:

1. $Prob(\overline{f_{random}}(G_K) > f_{original}(G_K)) \leq P$ **(Over-representation)**
2. $f_{original}(G_K) \geq U$ **(Minimum frequency)**
3. $f_{original}(G_K) - \overline{f_{random}}(G_K) > D \times \overline{f_{random}}(G_K)$ **(Minimum deviation)**

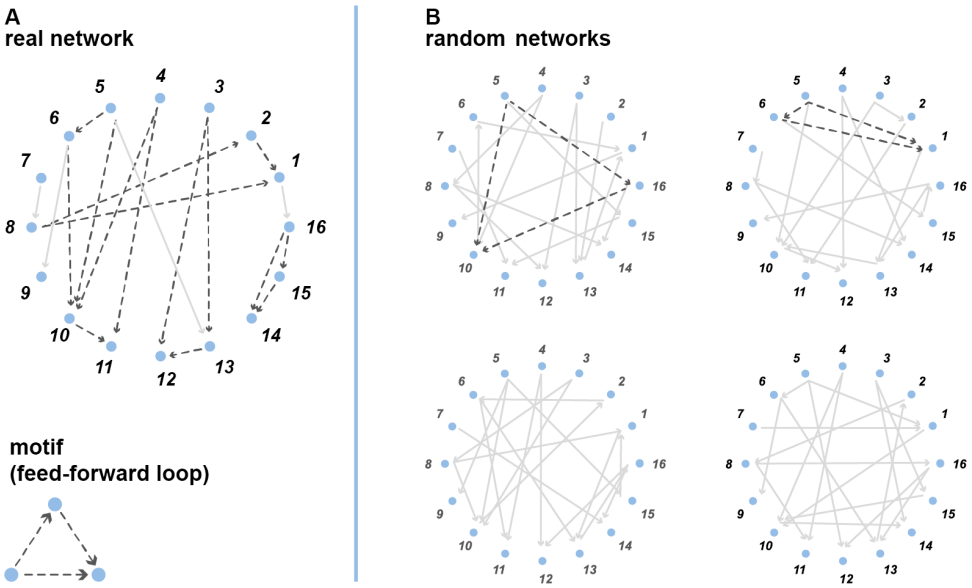


Figure 2.7: Representation of motif detection in a network. Adapted from [34].

The work presented in [35] by Milo et al. allowed to characterize networks based on motifs. In order to do that, the presence of patterns of size three (*triads*) was studied in each of them. It was used normalized z-score, which allows to compare the results obtained from networks of different sizes, to measure the over-representation. Basically, the profile is a vector with the normalized results for each of the subgraphs.

In Figure 2.8 the line represents zero (frequency exactly has expected). Points above the line represent motifs, because they indicate subgraphs that appeared at frequencies higher than expected. On the other hand, points below the line represent patterns that appear less than what was expected. We can observe that different networks display very similar significance profiles and they are already grouped in families of networks.

More details of the networks can be found in [35], but short names are indicative of their type. "TRANSC" indicate transcription networks, "SIGNAL" indicates signal transduction networks, "NEURONS" indicate synaptic connections and the names of languages indicate networks constructed using word adjacency of different texts where two words are connected if they are consecutive in one sentence.

A superfamily that is well understood is the last one, constituted by different semantic networks, where identical profiles occur because words are linked by their synonyms, regardless of the language. They have a very similar motif fingerpints. The other families require extra knowledge in other fields, like biology, to understand, for example, that the presence of the feed-forward loop as motif in some networks is expected.

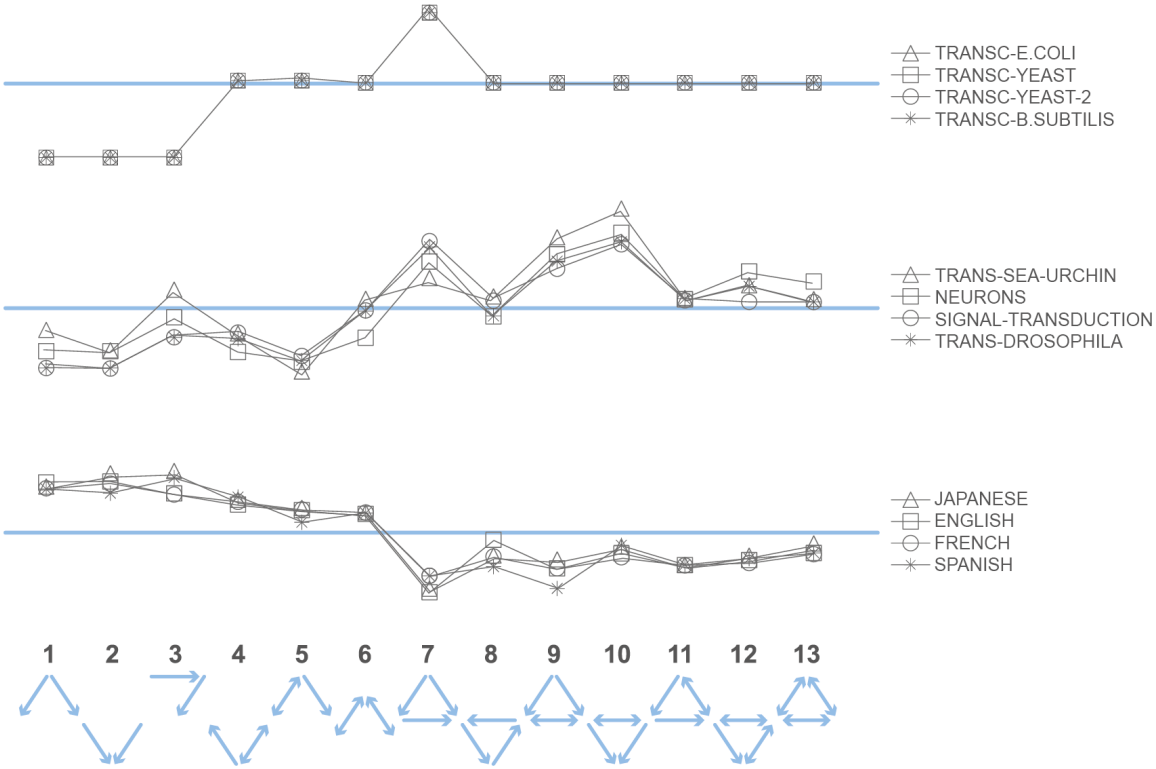


Figure 2.8: The triad significance profile of several networks. Adapted from [35].

2.2.4 Multiplex Problem

The definitions we gave are very basic and suitable for classic single layer networks, but with the necessary changes, they can be applied to other network types, including aspects such as weights, direction or colors. In multiplex networks, although it continues to be applicable, it is more complex since there are subgraphs spanning different layers. Bearing in mind that this is a recent area and that the definitions remain open and can easily be changed we will give our point of view, which is the one we use in the project.

In order to extend the notions to multiplex networks we need to be careful on defining what a subgraph is. Consider that a sequence of adjacent nodes defines a walk on a simple single layer graph. The length of a walk is the number of edges it contains, considering that the next step can be given from each of the nodes visited. A step is the basic component of a walk, that is, the connection between two adjacent nodes. This walk is the process that allows to identify all the subgraphs that are candidates for motifs.

But "how is a walk defined in multiplex networks?" [29]. In these type of networks, the concept is necessarily different because a step can be done to another layer. That is, it is perfectly natural for a step to occur between nodes in different layers [17]. However, it is also possible to consider that each node is represented in different layers and that the next step can be given in any of those without an intermediate step. In other words, we can reach one node in one layer and leave the same node in another as exemplified in Figure 2.9.

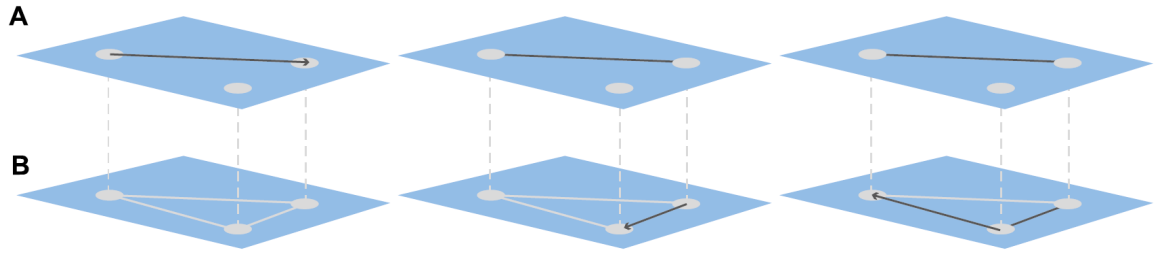


Figure 2.9: Representation of a subgraph in a multiplex network using a walk. In the second set there was a jump between layers without taking a step. Thus, this subgraph is a triangle with edge representation in two layers.

Remembering that a multiplex network is characterized by the presence of the same set of nodes in each layer, the concept of subgraph explained above is easily understood. That is, since there is an instance of each node in all layers it is natural that the transition between layers can be discarded since when reaching a node in a given layer we can think that this node has been reached in all layers. Even a subgraph of size two, which had only one format in a classic network, can now have many shapes, depending on the number of layers, as shown in Figure 2.10.

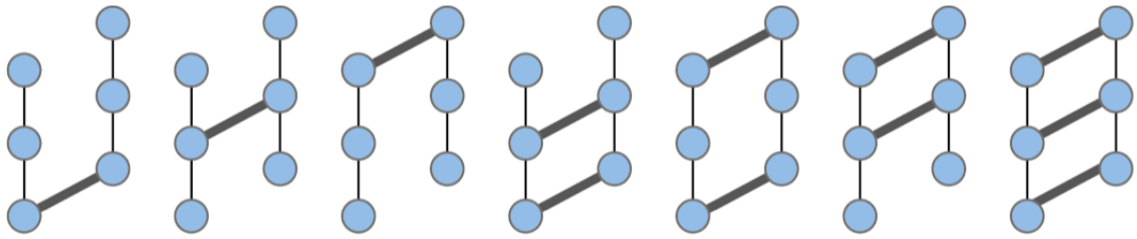


Figure 2.10: All possibilities of subgraphs of size two from a multiplex network with three layers.

However, a new problem arises. When are two graphs isomorphic? Two graphs are said to be isomorphic if after one permutation in one of them the other is obtained. This idea can be expanded when layers are added because they also can be permuted but, in other way, they also can be ignored. Looking at Figure 2.10 and trying to identify the isomorphic class of each of the subgraphs, we realize that the definition needs to be adapted and clarified to this new context.

One of the options is to naturally expand the idea commonly used, that is, to apply a permutation only on the nodes. Thus, in the tuple that represents them, the layers are retained and the node itself can be changed. The name used to refer this isomorphism is *node-isomorphism*.

Layer-isomorphism, like the previous concept, is manifested only by changing one of the elements of the node-layers tuples. In this case the relabel of the element of the tuple that represents layers. Intuitively, these concepts may come together to form more general isomorphism.

Figure 2.11 shows subgraphs of size three, and the class to which they belong, in two types of isomorphism. In one of them only the nodes are allowed to be permuted. In the right column we see a more complex case where permutations of both tuple values, nodes and layers, are allowed.

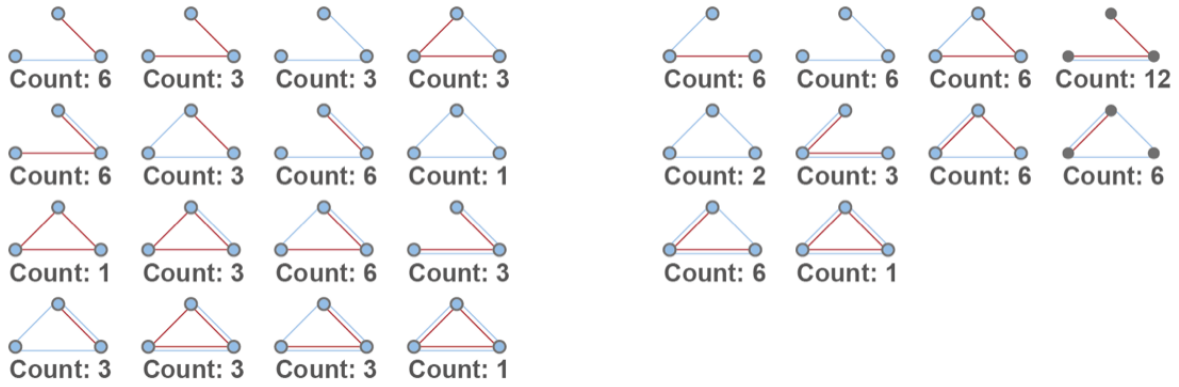


Figure 2.11: Isomorphism classes for multiplex networks with 3 nodes and 2 layers. We show node isomorphisms in the left panel and node-layer isomorphisms in the right panel. The count is the number of networks that are mapped to each class. Adapted from [29]

For motifs detection it is necessary to have other networks to compare the results obtained from the network being studied. It is not possible to determine if a measurement is within the expected range without comparing it with the same measurement in other contexts. Therefore, a null model that captures some features and satisfies a given set of constraints establishing what should be considered unexpected must be established. In networks with only one layer it is usually chosen to maintain the different degrees of each node. In multiplex networks, it is possible to think in the same way to generate the random networks. However, edges can now connect nodes in different layers, and there are different definitions, such as local and global degrees [7] that need to be taken into account.

Considering the degrees of nodes in a multiplex network the local degree assumes the same degree definition as a single layer network but considering only each layer at a time. While the global degree is the sum of the degrees of all instances of a node in the different layers. If the null model only ensures that the global degree is maintained there may happen that a node with links in all layers will have links in only one. Thus, the random networks to be generated can be considerably different. Therefore, maintaining the local degree at each layer ensures a more similar network and ensures that the global degree is maintained. In addition, to complete our null model, the most natural thing to consider is that each generated network maintains the same set of nodes in each layer.

However, as networks are more complex, models can take on new characteristics, in addition to the degree of nodes, to ensure even bigger similarity. Since layers can be seen as an edge characterization then this can be considered important in model generation. One idea, explained in detail in the chapter 3, is to keep also the types of edges that exist. That is, take into account the overlapping of edges in the different layers. If there is only one edge between two nodes that is present in all layers, it can be a good idea to maintain this uniqueness.

2.3 Related Work

This section shows some of the algorithmic ideas used to get frequencies and then discover motifs in different networks. We give a more detailed explanation of some of the main exact, sequential and general algorithms. We start by presenting the work in classical networks (including node-colored networks) and we exemplify algorithms for different conceptual approaches. We also offer a view of some of the latest work in the area of multiplex networks.

2.3.1 Single Layer Approaches

For the purposes of this work we are mainly concerned with exact subgraph census computations. This is because at this phase this is an exploratory research and we are really keen on understanding more perfectly the multiplex motif concept, without noise introduced on the computed frequencies. However, we should note that several non-exact strategies exist for classic networks, that are able to trade precision with execution time, returning approximate frequency values using sampling [12, 40, 48, 56].

We also focus only on sequential algorithms that encapsulate the main ideas used to find motifs and are more easily extendable to the multiplex case. We want to keep it simple whithout extra complexity added by the parallelization itself. We should nevertheless point out that several parallel approaches exist for the classic case [3, 46, 52], and we could consider in the long term parallelizing our approach.

Finally, we concentrate our efforts on describing the more general approaches, again because these are more suitable for extending to the multiplex networks. Some other classical network census methodologies are however more focused only on certain subgraph types, for instance, limiting to undirected and uncolored subgraphs up to a certain size. These methods can therefore take advantage of specific analytical properties that can speedup the computation [1, 26, 33] but it is not important for us at this first moment.

The currently existing exact methods can be divide into three different conceptual categories [40]. The first one, called *network-centric* computes a complete k-census of a network, by enumerating all sets of k connected nodes and using isomorphism tests to determine the subgraph type of each one. The second is focused computing the frequency of a single subgraph given and is therefore called *subgraph-centric*. Computing a k-census would therefore imply calling this method for each possible k-subgraph type. The third one is between the two previous approaches and concentrates on a custom set of subgraphs defined at the beginning, so that the user can indicate explicitly for which subgraphs he wants to have the frequency computed (it could be just a single subgraph, all possible k-subgraphs, or anything in between). We present the well known algorithms for each method category (network-centric: ESU and FaSE; subgraph-centric: Grochow; set-centric: G-Tries) and somehow related and used in this project, but a recent and complete survey can be read [47].

2.3.1.1 Network-Centric

It is natural that the first algorithm to identify motifs appeared in the same year that the concept was launched, 2002, and by the same authors in auxiliary notes to the original paper [34]. It is a network-centric algorithm, basically a recursive backtracking search, that enumerates all subgraphs with a determined size. It starts by choosing an edge and the two nodes that are its the endpoints. Then, other nodes will be added if they have any edges for those nodes that are part of the subgraph already selected. Whenever the size is achieved, the subgraph type (its isomorphism class) is computed and its frequency is incremented. Due to subgraph symmetries, the same subgraph could be found several times, and over-counting was avoided using specialized data structures.

In 2005 progress was made with ESU, a new and more efficient algorithm that avoided the before mentioned symmetry redundancy [55]. It begins by choosing a node (then it is executed starting in all the others) that is added to the current subgraph being generated. It also keeps a list with all the candidate future nodes. A node of this candidate list is then chosen to expand the current subgraph, and each time this happens, the exclusive neighbors (neighbors that are not shared by the other chosen nodes) are added to the list of candidates if they have a greater index than the last chosen node. The fact that they are exclusive and the indexes guarantees that each subgraph is enumerated exactly only once, avoiding redundancy. In the end, for each discovered subgraph occurrence, a highly efficient third-party isomorphism algorithm called *nauty* [32] is used to determine the subgraph type. Figure 2.12 gives an example of a search tree generated by ESU.

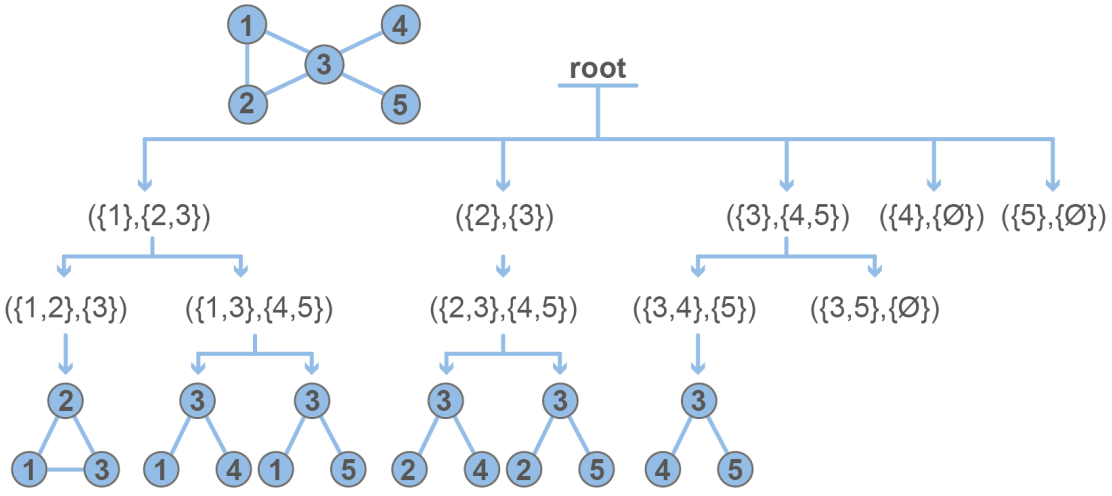


Figure 2.12: Search tree of ESU algorithm looking for subgraphs with size 3. Adapted from [40].

The ideas of ESU were further improved in 2013 with FaSE [36] and Quatexelero [28], that essentially use the same non-redundant enumeration mechanism, but improved upon the isomorphism testing, avoiding one test per occurrences. This is possible grouping many occurrences according to their topology, and doing one single isomorphism test per group, because the subgraphs share identical structures.

FaSE algorithm is based on the ESU and the idea is to take advantage of it adding one node at a time. Whenever a transition occurs from one state to another by adding a new node, that is, whenever it moves to another position in the search tree, this transition should be labeled. That is, if each occurrence is labeled, the joining of labels can lead us to the final state.

As seen before, ESU creates a recursive search tree. Each node of this tree is a distinct state that has two parameters: V_S and V_E. Therefore, if each state is distinct, the conditions are met to create a unique label as well. Therefore, each time a recursive call is made, the identification process can be performed through the current set of nodes.

When a new node is added it is stored the connections to nodes already in the set. Consider now the undirected case, when adding the k-th node and using an adjacency list as data structure. The size will be at most $k-1$ integers where an integer is present if there is an edge to the node with that index. For the adjacency matrix we will always have $k-1$ boolean values.

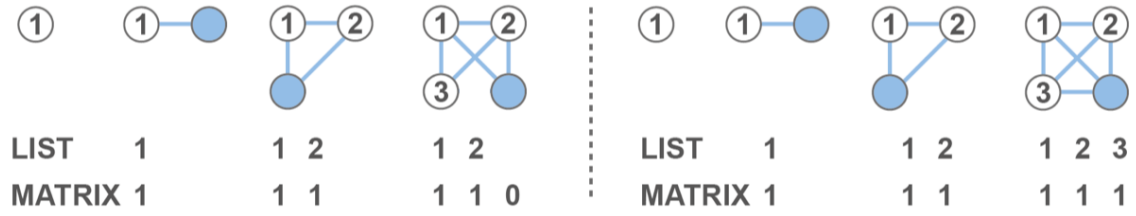


Figure 2.13: Two different valid labels schemes on two example graphs. Blue nodes are the ones being added. Adapted from [36]

.

2.3.1.2 Subgraph-Centric

In 2007 a different idea was implemented, and it does not care about the entire set of k-subgraphs [23]. This new approach, by Grochow and Kellis, focuses only on the frequency of a single given subgraph type and this allowed counting and obtaining larger motifs. Note that this can still be used to obtain a k-census if we first generate all possible non-isomorphic k-subgraph types (using for example the tools from [31]), and then calling the method individually.

The algorithm works essentially as a recursive backtracking process, but the search can be highly pruned by fixing a order in which the nodes can be matched and then only exploring the nodes that completely obey to the required connection to previously matched nodes. This implies that when a set of k nodes matches the query subgraph, we are certain of its type and no further isomorphism test is needed to identify the isomorphic class.

However, because of symmetries, the query graph can match multiple times to a single subgraph of the network. This is avoided by calculating symmetry breaking conditions, which ensure that a single match occurs. The technique of breaking conditions is made with the application of constraints on the labeling of the nodes. The order is by increasing degree and then by increasing neighborhood degree sequence.

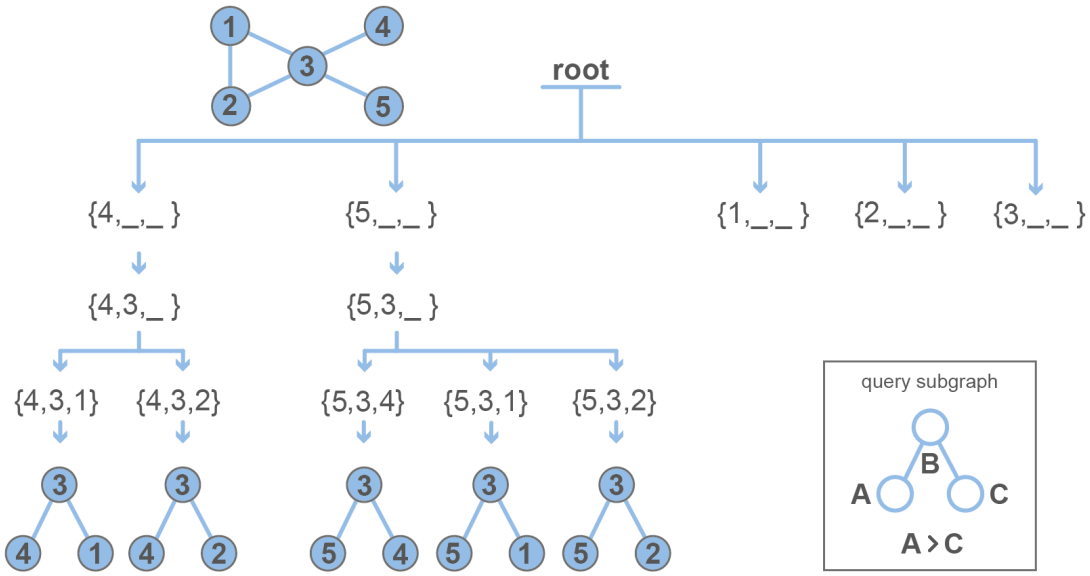


Figure 2.14: Search tree of Grochow and Kellis algorithm looking for subgraphs with size 3. Adapted from [40]. To perform the census using this method we must find all subgraphs with a certain size and then runs it for each one to determine its frequency.

2.3.1.3 Set-Centric

This section focuses on a methodology that is a mixture of the two above. That is, a method that instead of searching for all the subgraphs of a network or searching only for a single one, looks for a customized set of subgraphs as defined before. To complete this task, the g-trie was specifically designed for storing and discovering subgraph frequencies [40, 43].

This structure is akin to a prefix tree (also known as trie), a well known in computer science data structure to store words by taking advantage of words sharing equal prefixes. In this particular case, several graphs can share identical substructures, that is, contain the same subgraphs. When talking about this structure, special attention must be paid to the use of the term node since, like the graphs, trees also have this element. Therefore, in this section the larger structure will always be concatenated to the word node (trie or graph). Each trie-node represents the connections of a graph-node to the graph-nodes already inserted in the parent trie-node making each path in the tree correspond to a different subgraph. The connections are stored as a boolean array and the concept is illustrated by Figure 2.15. Although there are already algorithms to create the tree during execution [36], the initial version requires it to be pre-computed. It is possible, for example, to add all the subgraphs with size k, or just a subset of them. A subgraph is added at one at a time, starting with an empty tree, only the root (must be empty since there are two possible beginnings, a graph-node with or without a connection to itself). At each moment, the tree is visited to check if there is any trie-node containing a graph-node with the same connections as the graph-node that is being inserted. In other words, we look for trie-nodes that contain the same array of connections.

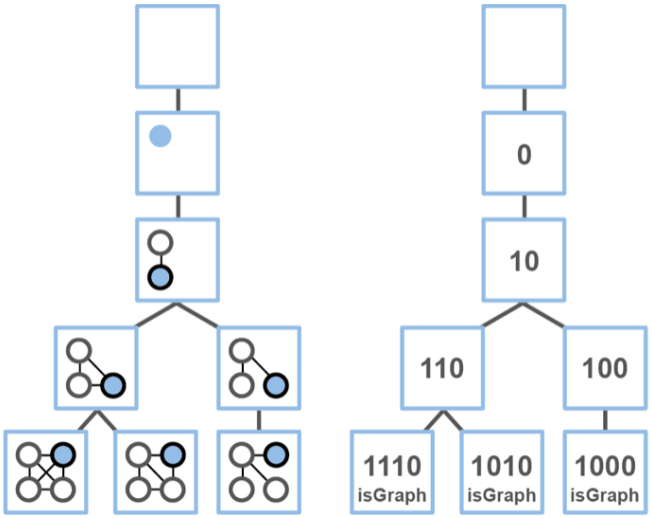


Figure 2.15: A g-trie representing a set of 3 undirected graphs. Each trie-node adds a new graph-node (in blue) to the already existing ones in the ancestor trie-nodes. The 0–1 boolean arrays represent the connections to already existing graph-nodes. Two childs correspond to different graph typologies that emerge from the same subgraph. Adapted from [40].

To ensure that g-trie leafs store different subgraph types, a canonical form is used to represent a graph in a single way, that is, a unique string assigned to each topology. Therefore, two graphs that are isomorphic must have the same canonical form and will always induce the same path from root to leaf. G-tries use the customized canonical form that was developed having in mind this usage, and therefore maximizes the amount of overlapping substructure, leading to a more compressed representation and a small tree [43].

After the g-trie is built, the next step is to get the frequencies of the subgraphs stored in the tree. The algorithm is based on a recursive procedure that expands subgraph sets that correspond to a g-trie path. The properties of the g-trie itself guarantee an highly constrained search, and symmetry breaking conditions ensure that each occurrence is only found once. G-tries can be from on to two orders of magnitude faster than a network-centric algorithm such as ESU and a subgraph-centric algorithm such as the one Grochow and Kellis.

One other advantage of this method is the calculation of frequencies in randomly generated networks. After obtaining the subgraphs contained in the original network, we can construct a g-trie containing only these subgraphs. So, when computing the frequencies on the other networks, we are only spending time and space with the subgraphs that are important. Only those that have been identified on the original network matter because of the definition of motif.

We should also note that g-tries have successfully been coupled with sampling to generate an approximated version able to trade accuracy for speed [41]. Furthermore, parallel versions of g-tries have been proposed for several different paradigms, including a distributed memory implementations using MPI [45], a shared memory implementations for multicores [3] and a MapReduce implementation with Spark [20].

The amount of total different possible subgraphs grows exponentially as we introduce layers. But, as we will see the multiplex networks can be transformed in graphs with colored nodes. Thus, we now present the same structure but with the capacity to support this kind of graphs. Even though it may not be a tool used to search for motifs it is an important data structure and fundamental in this work. We are not limiting the information that we can obtain and although we do not focus on this point, this structure is also able to handle colors on the edges. The authors, in addition to the structure, also give clear definitions of colored motifs and how they can be obtained.

The original approach contains in each trie-node the information relative to a single graph-node. Therefore, the logical extension is to store in each child of the tree the color of the graph-node that is being stored and to keep the connections to the graph-nodes in the same way they were kept. That is, to the boolean array color information is added. To know the color of the nodes of a subgraph, it is necessary to go through the g-trie, as to know the structure. This is what allows to reduce the space needed to store a set of subgraphs. Figure 2.16 illustrates a colored g-trie and how it stores some subgraph.

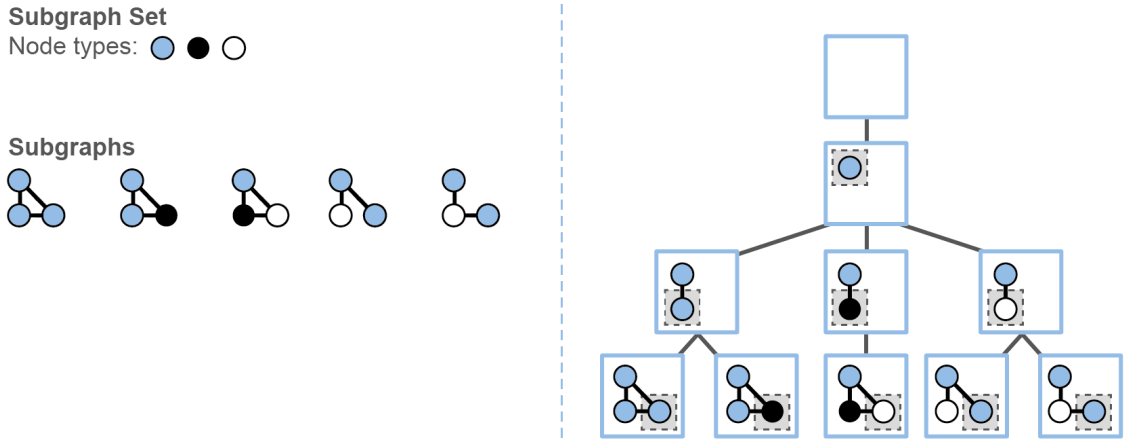


Figure 2.16: A g-trie storing 5 colored subgraphs, with 3 different node colors. Each trie-node adds a new graph-node (inside the small square) to the already existing ones in the ancestor trie-nodes. Adapted from [42].

Due to the combinatorial explosion that colors cause, the number of possibilities for subgraphs of a certain size is much greater. Therefore, it might not possible to generate all the possibilities and insert them into a g-trie as it was done in the classic version. However, one could build the g-trie on the fly, dynamically inserting only the subgraph types that do appear in the network. The occurrence of one subgraph remains different from another if they do not share at least one node and it is enough that the color of a node is not the same even if all the connections are. This has been successfully applied to colored networks and shown to provide new insight, identifying new motifs that were previously indistinguishable with uncolored version [42].Colors leads to a new concept of the similar random network. The idea of maintaining the degree of each node has been adapted to this new context. Thus, each node maintains the degree of edges that connect to nodes of a certain color. That is, each edge connects nodes of the same color.

2.3.2 Multiple Layers Approaches

The approaches used for multiplex networks are scarce and too specific - unless there is one that is not of our knowledge. That is, the results obtained are usually about a very particular context and it is hard to compare with our work where we tried to be as general as possible. Sometimes the work is focused on small size motifs and some of them do not search for subgraphs with more than three nodes. Furthermore, there are projects that consider a certain subgraph format, which they are interested like a feed forward-loop, without worrying about other structures. There are important and central definitions, like multiplex subgraph, that are not equal to ours. However, we provide some information about what exists in this area because there are some ideas that have been reused.

For detecting and analyzing multiplex motifs in large-scale corporate networks [51] the first step done was reduce the graph resulting in a weighted graph. As an intermediate step, the multiplex network is transformed into one with several types of connections between the nodes where each layer corresponds to a different edge type as explained in the Figure 2.1. Then occurs a process which consists of adding the different edges between two nodes by creating a binary label. First an order of the different types of edges (same as layers) present in the network is fixed. Then the creation of the string is to iterate over the types of the edges, in the fixed order, and concatenate 1 case the edge exists between two nodes and 0 otherwise. This binary label, with the size equal to the number of layers, can be interpreted as a weight where this encodes edges types. All the steps are shown in Figure 2.17. One problem is that the ability to represent weighted multiplex networks is lost. Another problem is that when a subgraph is identified and then transformed the most effective isomorphism test, nauty [32], does not support weighted networks.

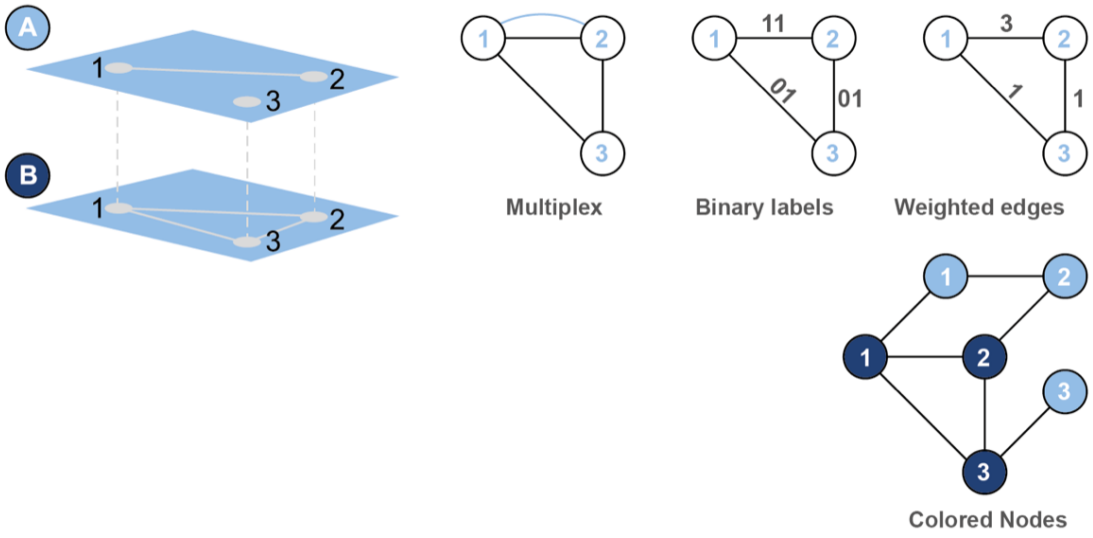


Figure 2.17: Multiplex subgraph encoding: from multiplex graph to multi-edge, to binary labeled, weighted representation and colored network. The process of creating the binary label occurs with the most natural order, first the layer A and then the B. Adapted from [51].

However, the workaround can be done using colored nodes, as nauty suggests in its manual, a solution that consists in transforming the network into a network with colored nodes, which is done by the authors. This being done in this exact way limits the isomorphism tests since the tool will only recognize the set of nodes as possible to be exchanged. Moreover, this work focuses only on a particular two layer network built with data from a database where the final result is not provided.

The same authors, for the random model used a network where each type of edges is modeled separately, fixing the degree of nodes for each type of edges (layer). Therefore, it is only the process used for classical networks applied for each layer individually. In addition, since one of the layers is bipartite, they had to categorize the nodes to ensure that the random networks retained this characteristic. Falling once again into the particularity of this network and forgetting the more general case.

However, this problem of being too specific is quite common. There are other works that use very particular null models because it makes sense in analyzing certain networks. As is the case of [8] in which for the analysis of brain networks by considering networks with two layers, anatomical and functional, they preserve the total degree at each layer, but they also keep fixed the number of connections towards regions in the same brain hemisphere and those towards the other hemisphere.

There are others where the network analyzed is "small and tight" [50] and where the motifs is not the most important analysis and did not show important results having just a few lines describing it. Furthermore, the framework used only allows to search for triadic motifs. This information is given by the authors of the study, since the authors of the framework does not make any reference to the ability to detect motifs but only other network related features as well as their visualization [19].

The methods used also vary and there is research based on mathematical formulas to compute the "number of all possible 3-node multilayer subgraphs" [58]. As stated earlier, there are works that do not use the same definitions. Even the most central primitive of this project, subgraphs, is different in [39]. It is a natural consequence of dealing with an emerging area and proof of this is the three papers cited in this page that were presented this year, during the elaboration of this thesis.

Two thesis were also defended, whose background is also multilayer networks. *"Construction and multilayer motif analysis of temporal fMRI brain networks"* and *"Graphlets in multilayer networks"*, with the same advisor, proves that the area is being studied by others. Although these have a stronger theoretical part than practical they came up with a new algorithm, MESU, that is an improvement of ESU for multilayer concept. The basic idea is to have another extensible list of layers, which combined with nodes, allows us to deal with tuples. However, its efficiency is not demonstrated by running times or the number of subgraphs discovered per second.

Despite scarce practical applications, there are extremely well-developed theoretical concepts.

In 2018 Kivelä et al. introduced a complete formalism in [29] to extend the concept of isomorphism to all general multilayer network types, establishing a reduction capable of transforming these into equivalent classic networks. To the best of our knowledge, no work has already taken advantage of these more mathematical ideas to produce specialized algorithms for counting subgraphs.

The idea behind graph reduction is also to reduce the problem of the multilayer network isomorphism to a problem of graph isomorphism. This is one of the main problems of graph theory requiring special attention in network science and it is imperative to count the occurrences of a subgraph in a larger network. The reduction is useful because allows to make the problem of getting the frequencies of subgraphs computable using software that is already developed.

All the mathematical theories were put on paper by the author and should be read for more details. The author also proposes what should be the way to identify the isomorphism between two multilayer networks. The suggested reduction is a linear function of the size of the multilayer network, which is favorable for the practical part of the problem because the complexity becomes the same as solve the problem in a classic graph.

The techniques used are based on the two situations that must occur in order to detect an isomorphism: their underlying graphs need to be isomorphic (underlying graph results from the first two sets of the quadruplet, $G_M = (V_M, E_M)$) and permutations that occur in each layer are connected to permutations that occur in other layers. In addition, one may want another kind of isomorphism where layers can also be exchanged to get another isomorphic graph.

The first problem is solved by coloring the nodes of the underlying graph according to the layer to which they belong, allowing only exchanges between them. The second problem is solved by adding extra nodes. A *supernode* for each node $v \in V$, connecting the node-layers tuples that the first element is the same that the supernode. That is, when there is a permutation of node labels on a layer, it occurs in the same way in the others.

Thus, the same technique, illustrated in Figure 2.18, can also be applied to allow permutations between layers by adding a supernode for each layer making the color (to identify the layer) redundant information. That is, with the extra nodes it is no longer necessary to color the nodes differently depending on the layer, since this distinction is now made with these new supernodes.

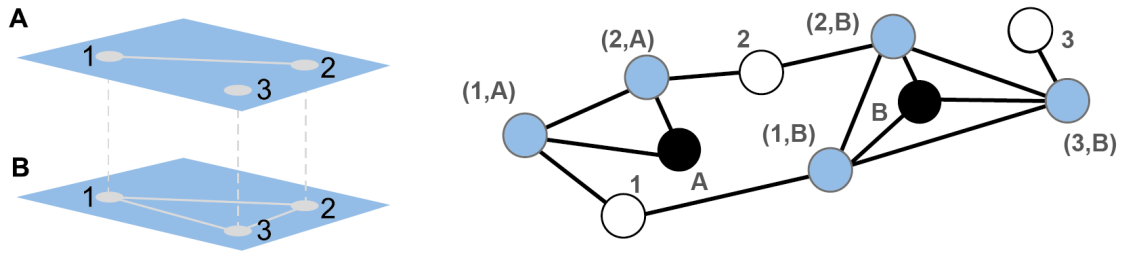


Figure 2.18: A multiplex network with two layers and three nodes and the correspondent node-colored network with supernodes. Adapted from [29].

Chapter 3

Approach

Here we present the practical work we have done divided into three main categories. Since this is sequential work, motif detection does not happen without subgraph counting, we first show the steps taken to compute census in a multiplex network and only after that the method we use to generate random networks. We also briefly show the output of our approach. There are two paths to get the results, but both are explained in detail.

3.1 Subgraph Census

As we saw earlier, it is possible to apply a transformation to a multiplex graph to obtain a graph represented differently, in more than one way. Basically, networks are transformed, somehow reduced, to apply the more classical tools that exist. This transformation is not a simple aggregation because it would lead to information loss. In our case, after doing this to make the census easier, we go back to recover what we lost.

3.1.1 Finding Classic Subgraphs

The search methods we use build a subgraph by adding one node at a time. We are looking for induced subgraphs, which means that if we add a node to a subgraph all the connections in the original multiplex graph, between the nodes already added and the new node, must also be added to this subgraph being constructed. Therefore, to find subgraphs with a certain number of nodes, in the first phase it is indifferent by which of the layers two nodes are connected. Quite simply, once we find a motif in the aggregated network, we go back to the original multiplex network to verify which edges actually exist between the set of nodes found. The need to consider all the edges that exist in the original multiplex network is because the subgraphs detected in the aggregated network do not show what is going on in reality. That is, there is missing information that can be important in the analysis, for example, to know if the edges that form a triangle are all in one layer or in several.

So, to make the census process easier and simpler, it is necessary to create an extra matrix that we call *overlap matrix*. The required size is $|V| \times |V|$ because we want to know if a node is connected to each other in any layer. Therefore, a node no longer has multiple instances and the result is a new graph with the same set of nodes but single layer. For that we needed boolean cells that are set to true only if one node has edges to another node in either layer of the original multiplex network or in the set of layers we want to consider for analysis.

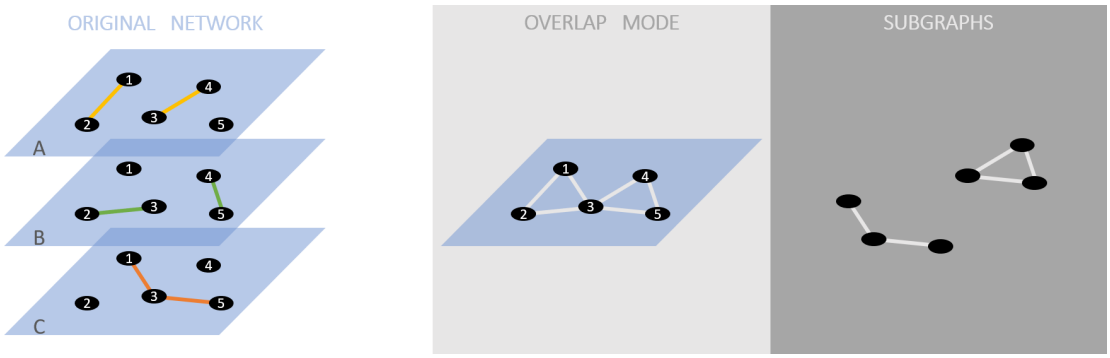


Figure 3.1: A multiplex network with three layers and the network obtained after the aggregation. Also the two types of subgraphs present in the overlap network from which it is not possible to obtain valuable information.

Network construction of Figure 3.1 is not an execution time bottleneck as this process can be done during edge reading and construction of the original network. Doing this apart, what may be needed for finding motifs if the same technique is used, requires traversing all edges to ensure that the construction of the graph is correct. Using adjacency matrices, all values are initialized to zero, or false, and they change to positive if an edge between two nodes exists and is the first edge between these nodes to be processed.

Algorithm 1 *makeOverlap*: Algorithm for create overlap network from graph G

```
1:  procedure OVERLAP(G, M)
2:      RESET(M)
3:      for all edges e of G do
4:          a ← INITIALPOINT(e)
5:          b ← FINALPOINT(e)
6:          if M[a][b] = False then
7:              M[a][b] = True
8:          end if
9:      end for
10: end procedure
```

This algorithm is linear according to the number of edges that exist. If this list does not exist and the information is stored in a matrix, or similar data structure, then the runtime gets worst because we have to iterate over all possible node connections and not just the ones that actually exist. But, once we get this network we are in a position to use a algorithm that already exists to find subgraphs. In our work we used ESU and then improved with ideas from FaSE, both explained earlier and remembered below.

The first algorithm we used is able to return all subgraphs that exist only once. This is a great guarantee because our main interest is to understand what types of subgraphs are in the multiplex networks that we are studying. As it is not a well explored area, execution time is not the main initial factor. That is, we prefer exact results over approximations. The choice to start facing the problem was this algorithm because, despite being one of the oldest, it is conceptually simple and still provides a state-of-the art enumeration of connected subsets of nodes.

The ESU algorithm begins from each node (line 2) where the order is not important. That is the first node of the subgraph being generated (first argument in line 4). It also keeps a list with all the candidate future nodes (line 3 and 15). One of this candidates is chosen to expand the the current subgraph (line 13) and it is added to the list with the nodes already found (line 14). Then, the exclusive neighbors are added to the list of candidates if they have a greater index than the last chosen node (line 15). The exclusive neighbours are the ones that are not shared by the other chosen nodes. The fact that they are exclusive guarantees that each subgraph is enumerated exactly only once, avoiding redundancy.

Algorithm 2 *ESU MP*: Algorithm for computing the frequency of subgraphs of size k in multiplex graph G

```
1:  procedure ESU(G, k)
2:      for all node v of G do
3:          V'_ext ← {u ∈ N(v) : u > v}
4:          EXTEND({v}, V'_ext)
5:      end for
6:  end procedure
7:  procedure EXTEND(V_s, V_ext)
8:      if |V_s| = k then
9:          supra ← MAKESUPRA(G, V_s)
10:         label ← CANONICALLABELING(supra)                    ▷ nauty
11:         INCREMENT(label)
12:     else
13:         for all node v in V_ext do
14:             V'_s ← V_s ∪ {v}
15:             V'_ext ← V_ext ∪ {u ∈ N_exc(v, V_s) : u > V_s([0]}
16:             EXTEND(V'_s, V'_ext)
17:         end for
18:     end if
19: end procedure
```

When the subgraph has already reached the desired size (line 8) then we proceed to the next step. In our strategy we need to call another algorithm (line 9) that we developed and will explain in the next section, to deal with the several layers of the multiplex graphs (we go back to the original network to reconstruct the subgraph found). Then `nauty` is used to determine the subgraph type, as in the original version [55]. However, taking advantage of the ideas developed by another algorithm called FaSE, the calls to calculate isomorphic classes can be drastically reduced.

The main idea of this algorithm is to take advantage of ESU method creating a label every time a node is found. As seen before, the algorithm creates a recursive search tree. Each node of this tree is a distinct state that has two parameters: V_S and V_{ext}. Therefore, if each state is distinct, the conditions are met to create a unique label as well. Therefore, each time a recursive call is made, the identification process can be performed through the current set of nodes. So, if each occurrence is labeled, the joining of labels can lead us to the final state.

Once a label is created at a time it is intended to have a data structure that is efficient in storing it. Moreover, it is intended that this structure take advantage of the hierarchical relationship that exists between the labels created, that is, labels created from a recursive call share the labels created before in that path. A tree complies with the demands and naturally fits to the idea of having an hierarchical identification of common topology. The data structure we use is a g-trie, which can be thought of as "prefix trees for graphs".

The algorithm 3 does not have very significant differences from ESU. There is a need to initialize a g-trie that will store the labels that are being created (line 2). The first insertion occurs from the root (third argument in line 4) and whenever an insertion occurs the last insertion point is returned (line 19). It will be from there that new descendant labels will be inserted recursively. In the end, the frequencies of each subgraph are stored at g-trie leafs and, differently from ESU, the increment of the isomorphic class should reflect this (line 9 and 14).

However, two different leafs may correspond to the same isomorphic subgraph type, and it is still necessary to obtain a canonical label to disambiguate and ensure that the final results reflect the real frequencies of each class. This is the point that brings unequivocal advantages to the algorithm, because the call to an algorithm that detects isomorphisms is not made for each subgraph as in ESU but for a set of subgraphs that correspond to the same leaf saving computational time.

As explained in ESU, our approach requires an extra algorithm to handle the presence of layers. In both algorithms, when a reference to the creation of a supra-adjacency matrix appears, it is relative to the reduction we make to turn the multiplex isomorphism problem in a classical isomorphism problem, which we explain in the next section 3.1.2. Also the creation of labels in FaSE (line 19) will be explained in the same section because we choose to do the same reduction every time a node is added and not just at the end.

The purpose of performing the transformation whenever a label is created is to save execution time. That is, as it is a recursive function whose purpose is to take advantage of the hierarchy, so reducing the subgraph during the creation of layers will make this method not so impactful because reductions can be reused. However, the problem arises in terms of memory because the generated g-trie is much larger since it will save the information about the connections between the set of nodes in all the layers.

Algorithm 3 *FaSE MP*: Algorithm for computing the frequency of subgraphs of size k in multiplex graph G

```
1: procedure FaSE(G, k)
2:     initGTrie(T)
3:     for all node v of G do
4:         extend({v}, {u ∈ N(v) : u > v}, T.root)
5:     end for
6:     for all l in T.leafs() do
7:         supra ← completeSupra(l.Graph)
8:         label ← canonicalLabeling(supra)                    ▷ nauty
9:         IncrementValue(label, l.count)
10:    end for
11: end procedure
12: procedure extend(V_s, V_ext, current)
13:    if |V_s| = k then
14:        current.count++
15:    else
16:        for all node v in V_ext do
17:            V'_ext ← V_ext ∪ {u ∈ N_exc(v, V_s) : u > V_s[0]}
18:            V'_s   ← V_s   ∪ {v}
19:            current' ← current.insert(newLabel(G, V_s, current))
20:            extend(V'_s, V'_ext, current')
21:        end for
22:    end if
23: end procedure
```

3.1.2 Reducing a Multiplex Subgraph

The idea behind our code flow is just to include the layers after finding out that a particular set of nodes forms a subgraph of the desired size. So, after that, we go back to the original network to check in detail the connections that exist between them. The reduction can now be applied to the subgraph and then it allows to calculate its isomorphic class using already developed third party tools.

However, as we have seen that in the FaSE algorithm a g-trie is built on-the-fly and therefore the methods used are necessarily different. Whenever a node is added by the algorithm, the connections it has to others already added are reduced and introduced in a specific label which will then, at the end, have a direct correspondence to the new data structure, which will be a supra-adjacency matrix. Then we just need to complete the matrix adding the supernodes.

Normally the number of layers and the sizes of motifs allowed are not very large so in computational issues this transformation is not problematic but it should not be neglected either because it is performed many times. However, building the supra-adjacency matrix in ESU requires iterating over the possible node set connections for each layer. The new position in the matrix is calculated (line 6 of the next algorithm).

Algorithm 4 *makeSupra*: Algorithm for create supra-adjacency matrix

```
1:  procedure MAKESUPRA(G, V_S, S)
2:      for l = 0 to |L| do
3:          for a = 0 to |V_s| do
4:              for b = 0 to |V_s| do
5:                  if G.hasEdge(V_s[a], V_s[b], L[l]) then
6:                      S[a + l * |V_s|][b + l * |V_s|] = True
7:                  end if
8:              end for
9:          end for
10:     end for
11:     SUPERNODES(S)
12: end procedure
13: procedure SUPERNODES(S)
14:     for l = 0 to |L| do
15:         for v = 0 to |V| do
16:             S[v + l * |V|][v + |L| * |V|] = True
17:             S[v + |L| * |V|][v + l * |V|] = True
18:             if isomorphism.permuteLayers() then
19:                 S[v + l * |V|][l + (|L| + 1) * |V|] = True
20:                 S[l + (|L| + 1) * |V|][v + l * |V|] = True
21:             end if
22:         end for
23:     end for
24: end procedure
```

The last thing to do is add the supernodes (line 11). The "small matrices" produced by the supra-adjacency matrix to represent the extra nodes have a special format. The supernodes representing the other nodes, if they are ordered, will always produce diagonal matrices. For the layers the matrices will have a row or column completely filled if all set of nodes are present in a layer that is assumed by us in multiplex networks. All of these formats can be seen in the Figure 3.2. The last two sections of rows and columns, colored with yellow and green, represent the supernodes. They can have different sizes because the first must be the size of the subgraph to represent each node. The last section, which is placed at the end because it may not exist, must have the size of the layer set. In this particular case the size is the same.

No connection between nodes of different layers is added in this project, but the matrix supports it. In fact, right now there is a lot of space that is wasted in the matrix because there are many positions that will always be filled with the false value (see in the Figure 3.2 the empty squares that exist between the different layers). It would be possible to mitigate this memory problem, through minor adjustments in our source code, but we prefer to leave this extension open for further research by us or others interested in this area. Thus, it is clear that the work to study multilayer networks with all types of edges, including between nodes of different layers, can start from this point and the necessary adaptations are not so many. Unfortunately, the duration of this work did not allow us to explore in that direction.

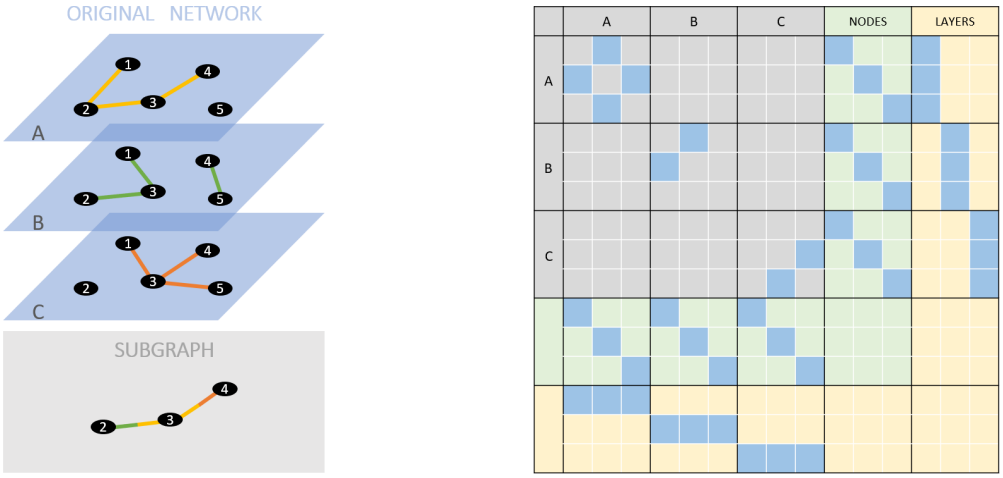


Figure 3.2: A supra-adjacency matrix with supernodes represented by yellow and green section.

The goal of the supernodes is to match the multiplex subgraph with the new classic graph. Thus, a supernode is added for each node index and all instances of a node must be connected to the same supernode. The purpose is to allow the calculation of different isomorphisms. These extra nodes are sufficient to calculate node-isomorphism because they guarantee that whenever two nodes swap in one layer they also swap in the others. However if we also want to calculate layer-isomorphism we need to add a supernode for each layer. All nodes present in a layer must have connections to the supernode that represents the corresponding layer. This allows the layers to permute with each other.

As it is shown in the Figure 3.3 the node-layer isomorphism leads to a smaller number of isomorphic classes. The result is expected since layer permutations allow us to group subgraphs that would not be possible just by exchanging nodes. As we see in the figure, there are five isomorphic classes in the left column. We highlight the two triangles that are of the same type because it is possible to map the nodes of one in the nodes of the other and where there is a connection in each layer. All other subgraphs have only two edges. The difference to the right column is that if these two edges are in different layers then it is possible to consider them isomorphic because the layers can exchange between them.

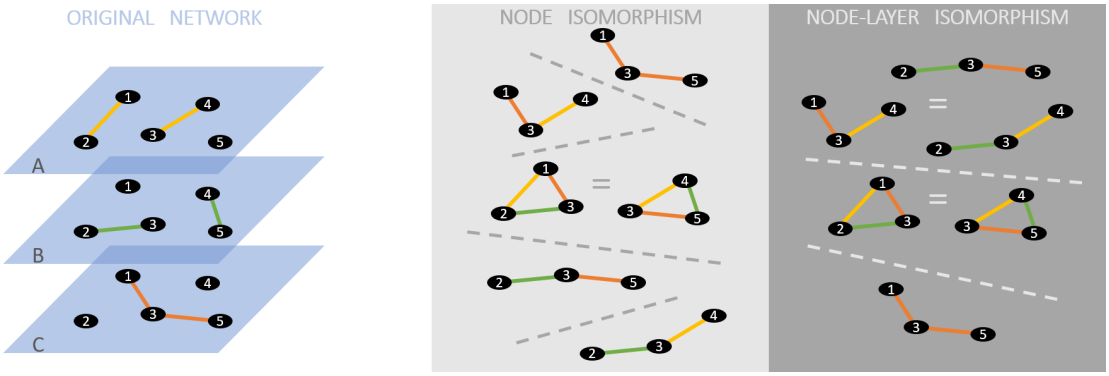


Figure 3.3: Two different types of isomorphism of subgraphs with size 3 from multiplex network.

While in the ESU algorithm the supra-adjacency matrix is created after finding the whole subgraph, in the FaSE algorithm a label is created and inserted whenever a new node is found. We use a g-trie to store these labels because we can easily cross the tree performing a jump or deepening. However, to do this we must have a technique that allows the subgraph to be rebuilt. This is necessary so that in the end the supra-adjacency matrix can be built and only supernodes need to be added by reusing the implemented method.

This process can be divided into several steps. One for each node that is added. In fact, these steps are the same of FaSE because this function is called each time a node is found by the main algorithm. The idea is that in the end, after joining all the labels that were created, in the right order, it will be possible to have all the information that allows to rebuild the subgraph. These labels, from the root of the g-trie to the leaf, must be enough without the need to use the original graph.

Therefore, based on the idea of the algorithm that creates the label in the original FaSE, it also only stores the connections of the node being inserted to the nodes that were previously added. Our big difference is that we multiply this process by the amount of layers that exist in the original network. Thus, we perform a concatenation of each label that results from each layer. We just need to know exactly what each position corresponds to. The colors of the Figure 3.4 helps to easily understand this concept.

Putting it in another way, but that may be more intuitive, this process can be seen as extending each matrix that corresponds to the subgraph in each layer. In step one extends the edge of node one to itself. It has three positions of memory because of the three layers. In the second step happens exactly the same but there is another detail. The extension is done starting with column number two and then concatenated also row two in the second half of the label. In this example they are the same because it is undirected, but it works with directed graphs.

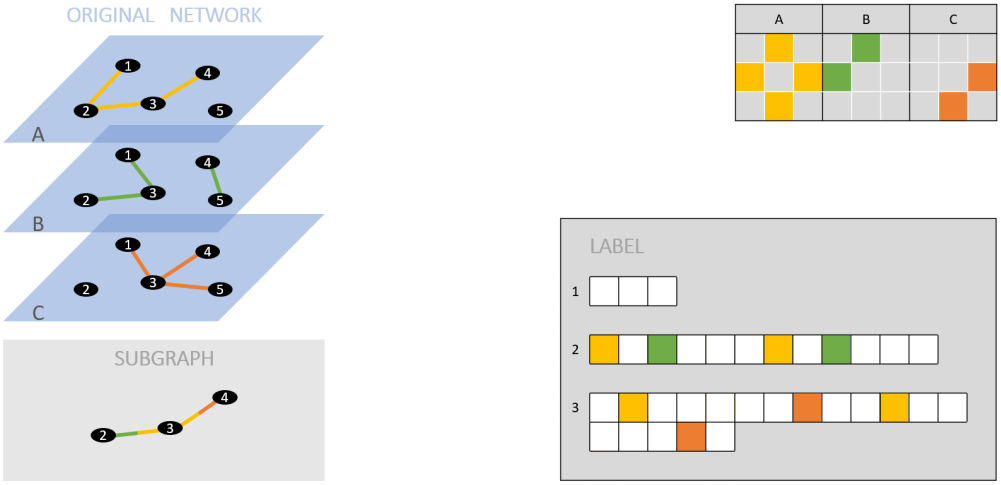


Figure 3.4: The label, divided by steps, built for the subgraph in the left using the algorithm FaSE. The colors in the label indicate a boolean position that has been set to true and it corresponds to the layer.

Algorithm 5 *newLabel*: Algorithm for create new label for FaSE algorithm

```
1:  procedure NEWLABEL(G, V_s, newNode, label)
2:      nodes_size ← |V_s|
3:      label_size ← |V_s| * |L|
4:      pos_label ← 0
5:      for l = 0 to |L| do
6:          for i = 0 to |V_s| do
7:              if G.hasEdge(V_s[i], newNode, l) then
8:                  label[pos_label] = 1
9:              else
10:                 label[pos_label] = 0
11:             end if
12:             if G.hasEdge(newNode, V_s[i], l) then
13:                 label[pos_label + label_size] = 1
14:             else
15:                 label[pos_label + label_size] = 0
16:             end if
17:             pos_label + +
18:         end for
19:     end for
20: end procedure
```

It is important to note that in the algorithm 5 there is little information that is constantly repeated. That is, when we are concatenating a column with one row the last position of each one is the same. Therefore, for each layer, in each step, a matrix position is repeated. It is the position that indicates the connection of a node to itself and that could be ignored in our work. Once again we have maintained this characteristic to keep our code base as general as possible. The problem of repeating the matrix position could be solved with some hard coding but it is easier to understand the way we did.

Our simple idea starts by calculating the size of the label if we only consider that the graph is indirect. That is, the size we calculate will actually be double. However, we use this value as an offset. So when we iterate over the possible edges between the nodes we can check the two way connections and build the label.

We recall that when we reach the end of the enumeration of subgraphs by FaSE algorithm we have a g-trie with labels that were recursively inserted. Thus, by obtaining a complete one, we can reconstruct the corresponding subgraph. All we need is to go through the label, from the first position to the last, and make a conversion that is the opposite of what is done by the above algorithm. Once we know the transformation we have done then we are also able to undo it. As stated earlier, after this we only need to add the supernodes setting the correct positions of the last columns and rows of the supra-adjacency matrix.

3.1.3 Get Isomorphic Classes Frequencies

When we reach this stage, using the ESU or FaSE algorithm, we are in the same situation. We have a supra-adjacency matrix representing a subgraph. In fact, in ESU we have a matrix for each subgraph found and in FaSE we have a matrix for each g-trie leaf created temporarily to store the labels. Therefore, the two existing paths converge at this stage and this step is exactly the same in both. We just need to be careful about how many subgraphs each leaf represents so that we do not get the count wrongly.

For the calculation of isomorphic classes a third party algorithm is used because it guarantees us to solve our problem with the best known time. Nauty "is a set of procedures for determining the automorphism group of a node-colored graph, and for testing graphs for isomorphism" [32]. The supra-adjacency matrix is a colorful graph and it supports these graphs so we just have to understand how the call is made. The user guide provided is not very intuitive and so we present our explanation here.

As input nauty requires to know the connections between the nodes. In fact, a new replica of our matrix is built, but in the format accepted by the algorithm. In addition, as a colored graph, colors must be provided through an array as explained below. We have to be especially careful about the different isomorphism we want to discover. This is because when we are using node-isomorphism to calculate the isomorphic classes the supernodes representing the layers have not been added and therefore there is no distinction between layers. Therefore, it must be stated that each layer has a different color. This is done by setting to true the position that corresponds to the last node of each layer. In the other type of isomorphism it is indicated that all nodes are of the same color except for supernodes. Thus, the array that provides the colors should be like exemplified in Figure 3.5.

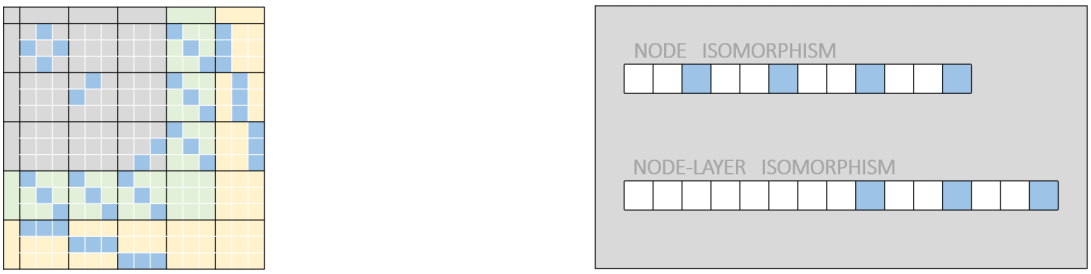


Figure 3.5: The arrays representing the color of the nodes in a way nauty can read it. The positions are boolean and must be true at the last node of each set of nodes of the same color.

Nauty returns a canonical way of representing the subgraph, that is, two subgraphs that have the same isormophic class also have the same label. Therefore, a g-trie is used to store the returned classes because the same classes will correspond to the same leaf and it is possible to compress the information in the same way it is done with subgraphs. The total number of leafs represents the isomorphic classes and the number of subgraphs is obtained by summing the count present in each leaf.

3.2 Network Motifs

Since we are already able to count subgraphs in a multiplex network we can then move on to motif detection. What we have to do now is count subgraphs but in randomly generated networks. For this, it is necessary to obtain a null model capable of absorbing all the necessary characteristics so that it can then be affirmed that a subgraph present in the original network is over-represented or not.

The standard procedure is to use a Markov-Chain method that has a procedure for rewiring edges from the original network as it is described in Figure 3.7. Starting with the original graph, a pair of edges (a, b), (c, d) is repeatedly swapped by (a, d), (b, c). This allows to preserve the degree (in and out) of the nodes if we ensure that the degree of a is the same of the degree of c or the degree of d.

The process described above is what we use for our null model one where we maintain the degree of edges on each node and doing this for each layer separately. That is, when two pairs of different connected nodes are found they can exchange as happened in layer A of Figure 3.6 where the edges $(1, 2)$ and $(3, 4)$ changed to $(1, 3)$ and $(2, 4)$. An identical situation happened in layer B but in the last one it was not possible to exchange edges.

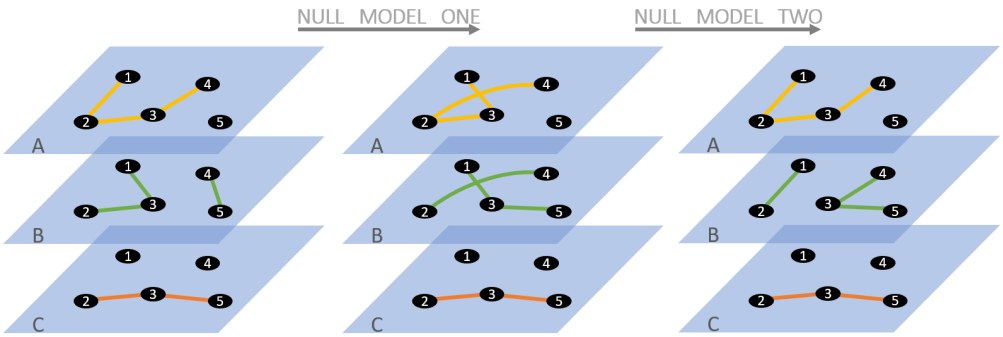


Figure 3.6: Two transformations of a network according to two different null models. The left network is transformed into the middle one according to model one. The right network is obtained from the middle network according to model number two.

The difference for the second model is that all layers are considered when looking for edges to exchange. If we consider the middle network of Figure 3.6 to apply the model only the edge between $(1, 3)$ and the edge $(2, 4)$ have the same set of connections (layer A and B) and therefore it was the only exchange that was made. We can think about this as categorizing edges and just exchanging edges that are of the same type. Therefore, the edges that are only present in one layer will keep this type while the edges that are in more than one will be of a new type that is created using the several layers.

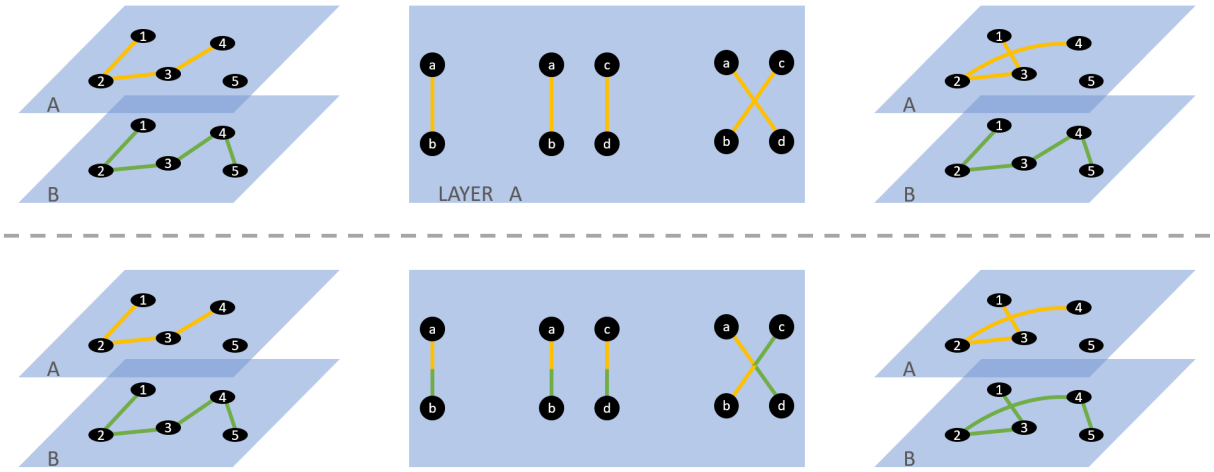


Figure 3.7: Two different edge exchanges of a multiplex network according to two different null models. The first one takes each layer into account separately while the second looks at all the layers at the same time.

3.3 Showing Results

A visualization tool can be useful for identifying and understanding subgraphs present in multiplex networks, especially in this area where subgraph types are not as well known as the classic ones. So, one of our concerns since the beginning was to have available methods that let us to manage the results, to easily search for interesting patterns and show them in a explicit and clear way where the different isomorphic classes are easily distinguished.

First of all, before focus on the visual part of the results, we must ensure that the user has the possibility to choose a range of variables that interfere with the results. We are talking about things like choosing the type of the isomorphism, node or node-layer, the most appropriate null model, which we have two options, the set of layer to consider during the analysis and the most basic of all, but importantly, the size of the subgraphs.

Results are shown in a sequence of html pages and the first one shows only numbers about the results obtained. From the number of subgraphs found, the number of different isomorphic classes identified to the execution time of the algorithm. In fact, an overview is made of the characteristics of the network and the variables that led to the results to ensure that in their interpretation there are no mistakes.

All this information is presented in a box system that facilitates whenever it is decided to introduce or remove any data. That is, it is only necessary to add a box or remove without really worrying about where to add this info. We just have to decide the order in which it is entered, before or after other important statistics. Moreover, the boxes may have more than a certain size that allows to have an ever-aligned layout as we will see in section 4.2.

Then we can start thinking about the other pages. If we want all subgraphs on the same page or how many we want on each. Whether we want to sort in ascending or descending order. Choose to see all the results or just a few and how many. Hide or show subgraphs that exist in random networks but do not exist in the original network. All of this allows to have custom results that may be important for a specific application or to assist in the analysis of the results obtained.

In short, the results consist of a first page of presentation, which have all respective numeric results from the network and the execution itself, followed by other pages with the sequence of subgraphs or motifs that were found by the algorithm. These patterns can be represented in one of two ways. They are drawn in expanded or compact form and they consist of what the name implies. The differences between the two versions can be seen in Figure 3.8 where both sides represent the same subgraph.

On the left side each layer is represented separately and there is no need for use colors on the edges (only present to facilitate the correspondence between the two subgraphs and in reality the edges in this option are white). In the compact version one edge is divided by the number of connections that exist between the two nodes in all layers and each segment is painted with a distinct color. This strategy is fundamental for networks with many layers that might not all be visible on the screen at the same time.

Figure 3.8: Multiplex subgraph with two layers and its correspondent compact version.

As said before, using html pages to show the results made us choose to use a vector drawing tool supported by all modern browsers. This means no extra tools are needed and therefore easy to display the subgraphs. Scalable Vector Graphics allows to draw various shapes that are sufficient for the representation of networks. Circles for representing nodes and lines for edges are the main and fundamental to achieve the goal. However, the way they are used by us makes the graph representation quite clear and explicit.

Although this part of the results presentation may be neglected by many researchers, it is of great importance to our project. Contrary to what would be expected in other works this part began to be thought and developed early because it played a fundamental role in this thesis. It was instrumental in helping us understand multiplex networks. We are sure that this intuitive representation gives, even to the people who are not minimally connected to the area, the ability to understand what is going on.

Chapter 4

Experimental Results

In this chapter we present the results we obtained after applying our approach to two different data sets. One of them contains synthetic networks built through various models. The other is formed by real networks where we try to have a diverse set to ensure better conclusions. We have done several experiences to understand how our approach behaves. The most important are in this chapter, divided into sections.

4.1 Computer Environment

All experimental results were obtained on a two core machine with AMD Opteron(TM) Processor 6274 where CPU max frequency is 2200 MHz. It has 8 GB of RAM. All code was developed in C++ and compiled with gcc 7.4.0. We use the version 2.6 of nauty as external package for computing isomorphism groups of graphs.

4.2 Visual Aspect

In the first page the important numerical results such as network information (number of nodes, edges and layers), number of subgraphs found and number of different isomorphic classes are shown. Furthermore, all the details that originated the numbers. That is, the arguments that were passed to the program before its execution such as subgraph size, isomorphism type and the method used. In addition the runtime on the original network is shown. When the main goal is motif detection then new boxes are added to give average runtime information on randomly generated networks. Therefore, the box format, exemplified in left side of Figure 4.1, allows at anytime to show more information if desired maintaining always an aligned format. The last box, which have an arrow pointing to the right, allows to go to the next page, the first one showing the subgraphs found.

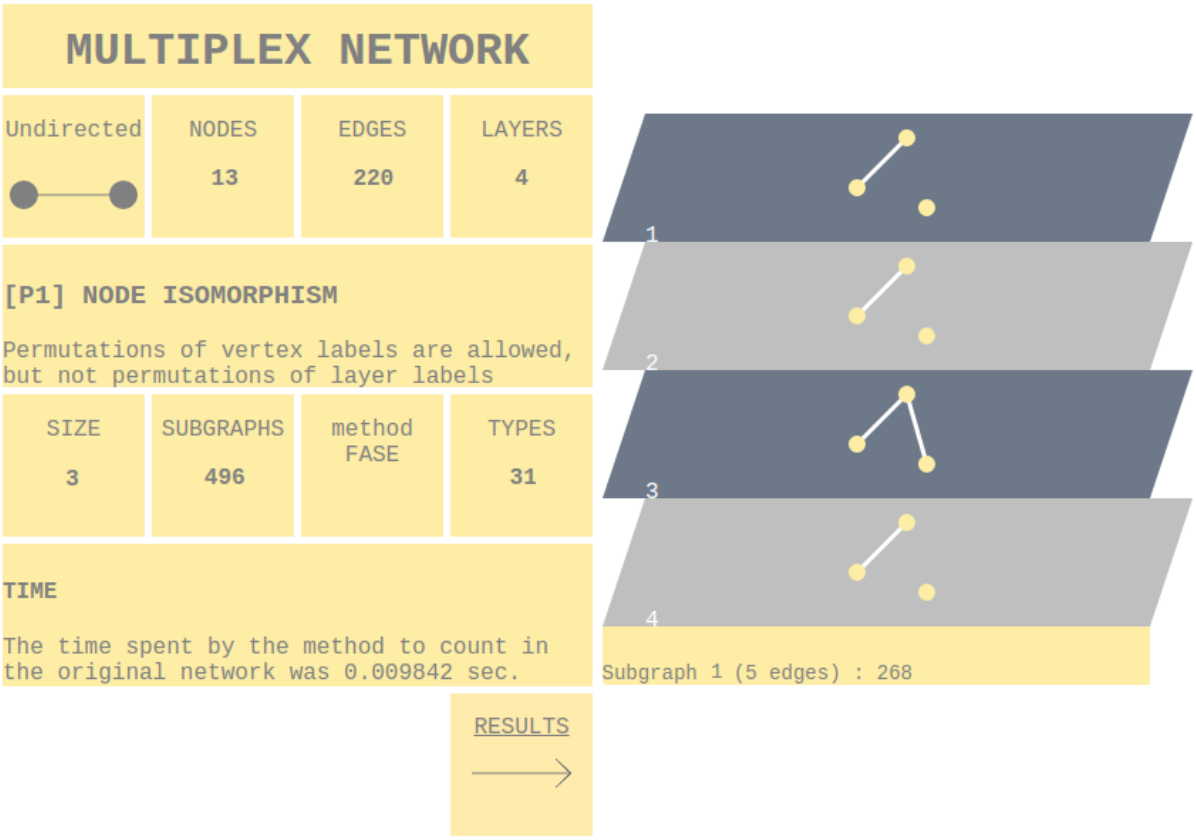


Figure 4.1: Output example of our approach where html is produced by our program. The first page on the left and an example of the subgraphs on the other pages on the right.

The subgraphs found can all be shown on the same page if desired, otherwise they are shown on multiple pages to not overload the browser rendering all vector shapes. Drawing the layers, as in right panel of Figure 4.1, is a clear view of the subgraph because the layers are explicitly drawn separately. However, there is also a compact version where only one layer is represented and colors are used to distinguish the edges. The compact version was developed because it could not be practical to see a subgraph with many layers and it is explained in 3.3.

4.3 Synthetic Networks

One of our biggest difficulties was finding real networks with layers to use to test our approach. Therefore, one of our biggest needs was the generation of networks capable of following the characteristics we intended. Using models that already exist for classic networks we adapted them to the context of layers (more than generating a classic network multiple times and considering each one as a layer). Then we were sure we had networks from where we could obtain experimental results.

4.3.1 Models

Erdõs-Rényi

Erdõs-Rényi (ER) is a model for generating random graphs [21], which includes an edge between two nodes with a certain probability. Therefore, in each model, according to the desired number of nodes, we iterate over all the pairs of nodes and each edge is included in the graph with probability independent from every other edge. All networks from Figure 4.2 were generated with 50 nodes as parameter. However, when the probability is too low, this number may be lower because there may be some who has no connection to any other. This happens in the left network. In the others it is possible to verify that, the greater the probability, the network becomes denser.

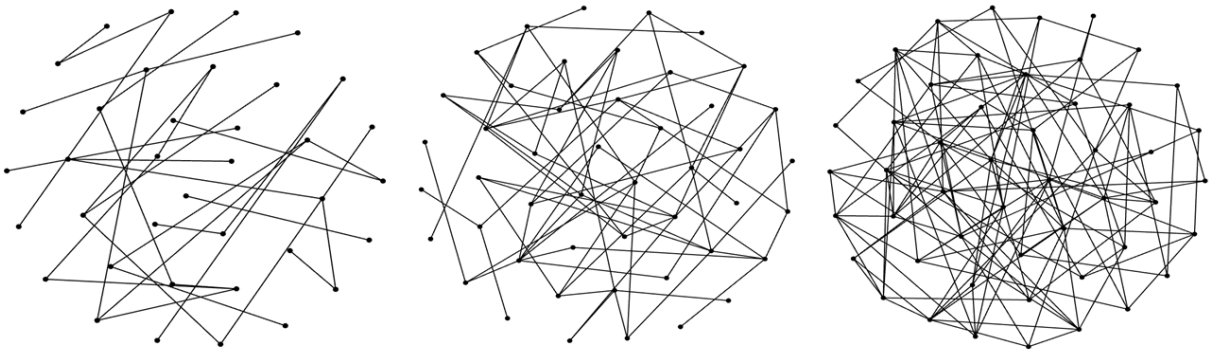


Figure 4.2: Display of three graphs generated with the ER model. The probability increases from left to right.

We consider that in this particular case no adaptation to the layers is required. Since it is a totally random model then the generation of several networks where each one is assumed as a layer is perfectly natural because the multiplex will keep exactly the same randomness as the classic one. In the experiments, this model will be referred as `ER_MP`.

Barabási-Albert

The Barabási-Albert (BA) model is an algorithm for generating scale-free networks [4] whose degree distributions follows the power law where most nodes have few connections and some nodes have a high number of edges, which we call *hubs*. They are generally more resistant to accidental failures but vulnerable to coordinated attacks. The network grows by adding new nodes over time and these new nodes connects to existing nodes in the network with probability proportional to the degree. Therefore, they are more likely to connect to hubs. The power-law degree distributions is a property widely observed in many natural systems including the internet, citation networks, and some social networks. Out of curiosity, the algorithm is named for its inventors Albert-László Barabási and Réka Albert and not only for the first author whose first name is the same as the last name of the second.

The three networks in the picture vary from one to another because they have a different attachment parameter. That is, in the first case, each node that is added only connects to another node (network where the presence of a single hub is easily visible). In the second case, each of the new nodes connects to two others and in the third to three. This characteristic is especially noticeable in nodes that are represented on the outside of the circle.

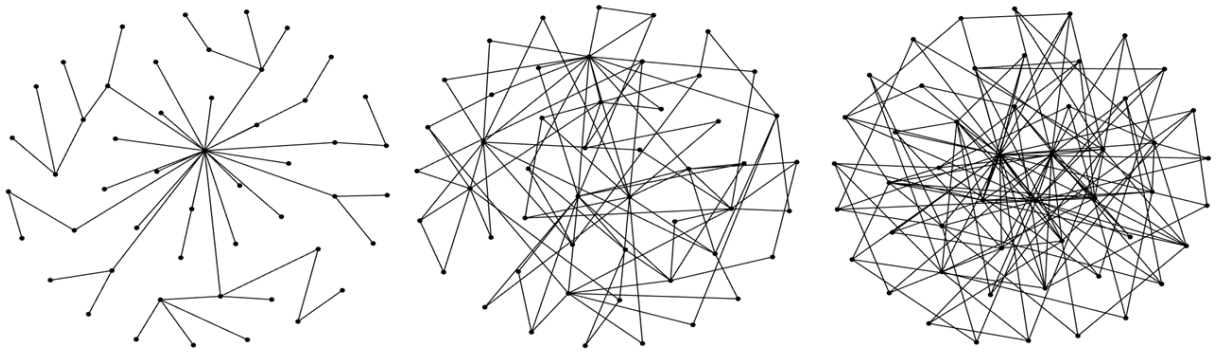


Figure 4.3: Display of three graphs generated with the BA model. Each has 50 nodes and a parameter of attachment between 1 and 3, from left to right.

The networks presented in Figure 4.4 are our attempts to have a model, adapted from BA, which includes the layers. They all have an attachment parameter equal to one and the presence of two layers (orange and blue). The green color represents the edges that are present in both layers. This is the way we found to get our networks visually using the Gephi tool [6].

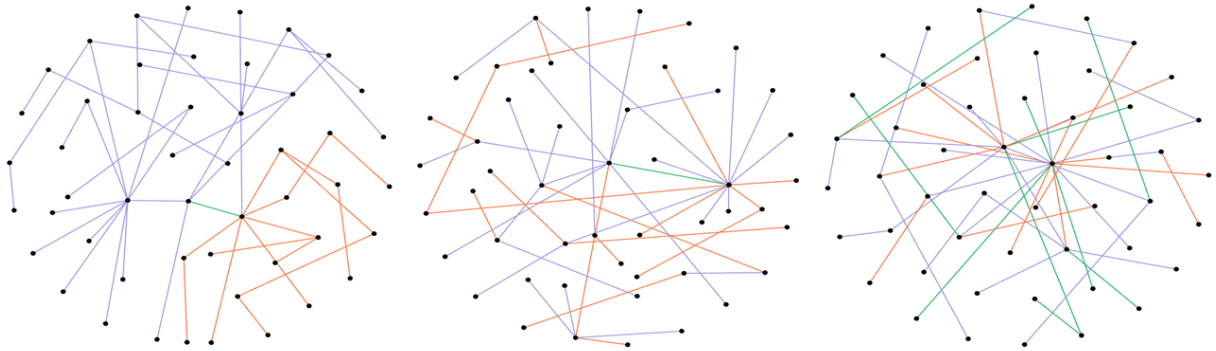


Figure 4.4: Display of three graphs adapted from the BA model. Each has 50 nodes and a parameter of attachment $m = 1$.

In the first case we only allow a new node, which is being added to the network, to connect to the other nodes in a particular layer. That is, considering the initial edge in both layers, colored in green, it is from there that the new nodes connect but always by the same layer. The attachment that is made takes into account the layer and it is like first extending one layer and then another. This does not imply that there is no overlap, but the truth is that the hubs of one layer are unlikely to be the same as in another. We truly believe that this model is very identical to generating the Barabási-Albert model twice and considering each one as a different layer. Therefore, we had the need to change and bring it closer to what we think are the real networks.

The second adaptation is already closer to what we want and closer to our definition of subgraph. That is, from a node, the network can be expanded in each of the different layers. However, with this happening in this way, it is ensured that there is no edge overlap (beyond the initial subgraph that is added to the two layers) and that is why the evolution for the third model emerges.

Considering the need for more overlapping edges we decided to go through the edges generated and with a certain probability add that edge in all the layers. This is the final model that we will be using throughout the rest of this chapter. We will refer to it as `BA_MP`.

Watts–Strogatz

The Watts–Strogatz (WS) model is an algorithm that produces networks considered small-world [54] where most nodes can be reached from every other node by a small number of steps. Short average path lengths and high clustering are two properties that can be found here.

The basic idea is to get an argument, as a parameter, that indicates how many neighbors should have a node. This argument is assumed to be even to ensure that there are as many neighbors on the right as on the left. After creating this chain we guarantee the high clustering property. Then going through all these edges and with some probability removing them and rewire to other nodes, found randomly, we grant the other property, short average path, because we are connecting sets of nodes that are far away.

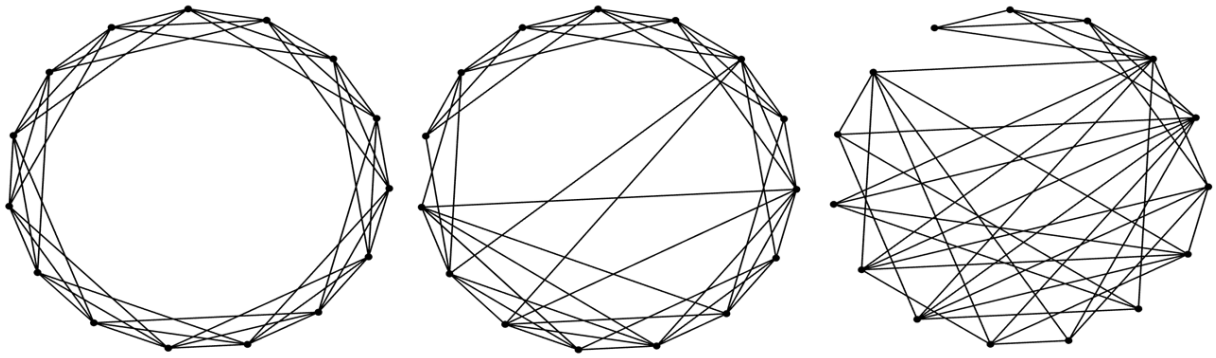


Figure 4.5: Display of three graphs generated with the WS model. Each has 15 nodes and 6 neighbors (half for each side of the circle). The probability of rewiring increases from left to right.

In the left of the Figure 4.5 it is possible to see a network that follows the first feature where all nodes are connected to the same number of neighbors where half are on each side. This causes that peculiar structure. There is no edge removed and connect to another node because the probability is zero. However, in the central network this already happens on some edges (visible through the inside of the circle) while the probability in the last case was close to one and almost all edges were changed.

If we consider the network generated without the rewiring process and if we think of several networks of this type to create the multiplex network we realize that the overlap of edges will be total. Even with the rewiring process, although it may not be total, it will be very high and therefore we consider that it is not an effective way to represent multilayer networks. This is why we thought of an adaptation, presented in the Figure 4.6, where colors represent the same as in previous models. Orange and blue one layer each and green the edges present in both.

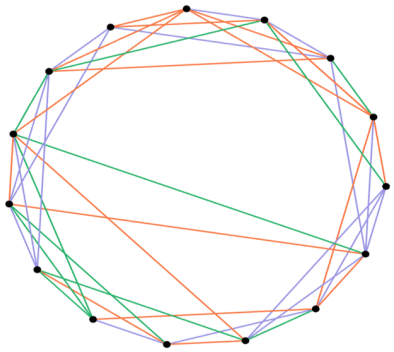


Figure 4.6: Display of a graph adapted from the WS model. It has 15 nodes and each node had 6 neighbours before the rewiring process.

Our main idea is to first create all possible combinations of layers and assign an index to each one. Then in the next phase generate a classic network according to the WS model. After it, iterating over all created edges, randomly choose an index, which matches one of the combinations, and add the edge to the layers present in the combination. Thus we guarantee the same model but in multiplex with a well distributed edge overlap. This is the adaptation of the model that we will be using, and we will refer to it as `WS_MP`.

4.4 Models Fingerprint

In order to ensure that the adaptations of the models explained above are within expectations, we conducted a series of experiments with the ultimate goal of confirming the identity of each one of the different models. The work consisted in the generation of synthetic networks according to the models `ER_MP`, `BA_MP` and `WS_MP`. We used different parameters, varying the number of layers and the number of nodes. Then, we count subgraphs on different networks to see if these patterns could be seen as differentiating elements.

In fact, the metric used in this case was not the frequency but the concentration. This allows comparing values of networks whose size is different since the frequency alone would be incomparable because larger networks would have higher frequencies. This value is obtained by dividing the frequency of a given subgraph by the total number of subgraphs found in the complete network. Other agnostic metrics relative to the size of the network could be used, but this is enough to show what we want.

We choose to show the subgraphs of size four and with the highest concentration in each of the networks, resulting in the set of ten shown in the bottom of the Figure 4.7. The results presented are for two layer networks (orange and green color). Note that these subgraphs would not be detected in classic tools, or at least would not be distinguishable. The first and last three all have the same structure and the rest have another structure. Therefore, this would result in only two detected subgraphs.

The first network of each model consists of 500 nodes. The next double the first, and the third also double the second with 2000 nodes. The concentration graphs are practically the same in networks of the same family, as expected. Also as expected, it is the types of subgraphs that appear where, for example in `BA_MP` networks, subgraphs [4-7] occur because of the presence of hubs. Therefore, we can conclude that the models we generate are sufficiently different to be used as a basis for analysis in our other experiments.

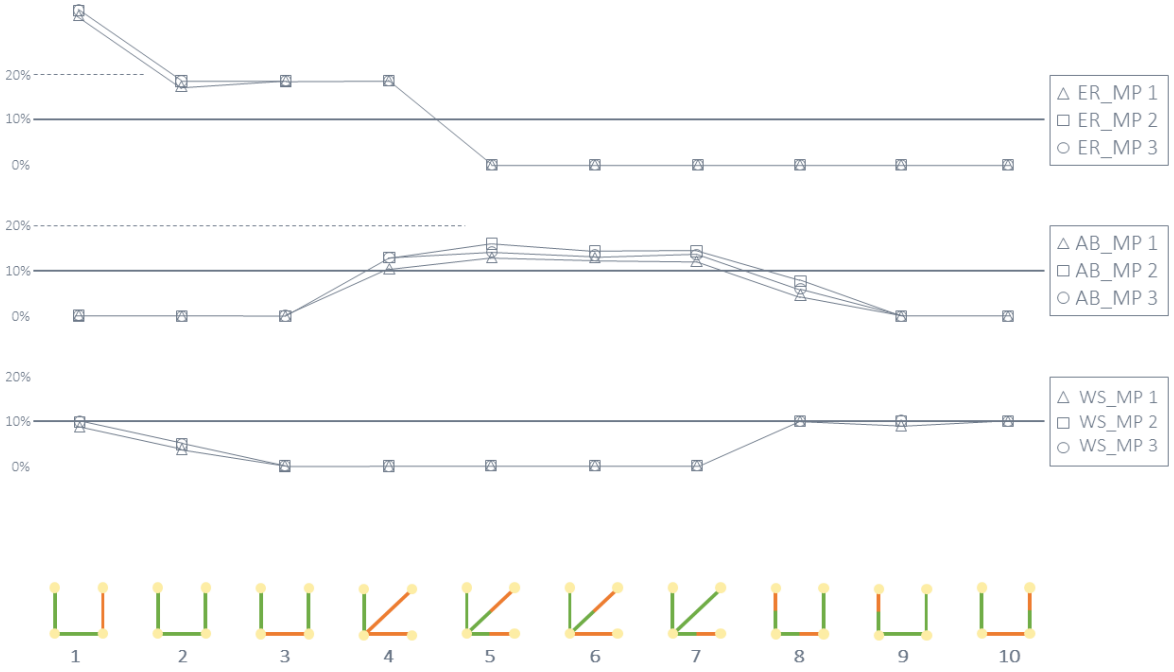


Figure 4.7: Networks fingerprint using subgraph concentration. A set of 10 subgraphs and 3 networks of each model.

4.5 Comparing Algorithms

We now want to show the differences that exist between the two variants of our approach, the ESU method and the FaSE. However, in order to better interpret the results, we consider important to look at some networks numbers such as the occurrences of subgraphs and the different isomorphic classes. These numbers then play a fundamental role in the runtime because the algorithms are enumeration based, which means that they must list all subgraphs and networks with more will take more time.

The main objective of showing the next three tables is to start thinking about what will be expected as the behavior of the algorithms through these numbers. The big difference between the ESU method and the FaSE is the number of calls to a method that returns the isomorphic class. The first one does an isomorphism test for each subgraph found, so the number of calls is equal to the number of occurrences. The second encapsulates the subgraphs before the call and so the number will be smaller.

This reduction of calls may be indicative of the expected gain and that is what we want to prove. Therefore, we show the different amount of calls between methods (in first and second column sections of the tables), which is the main and biggest problem of all this work. The tables are in ascending order of *Reduction* (ESU calls divided by FaSE calls) section. That is, the first table, which refers to the `WS_MP` model, shows a slight decrease of calls where the orders of magnitude are lower when compared with the other models.

The comparison can and should be made with the other two tables because bigger differences are noted in Table 4.2 and even more in Table 4.3. Therefore, the FaSE method is expected to bigger improvements in networks generated through `ER_MP` model than in networks generated using other different models. But, it is expected to be better than ESU in all cases.

We also want to share how difficult it can be to show results in this context as layers add a new dimension to classic works. If we want to show only some numbers it is not possible because they will necessarily be too many.

L	**N**	**Occurrences/ESU Calls**			**FaSE Calls**			**Reduction**		
		3	**4**	**5**	**3**	**4**	**5**	**3**	**4**	**5**
2	0500	6129	30020	167317	45	1011	11019	$1{,}4x10^2$	$3{,}0x10^1$	$1{,}5x10^1$
	1000	12400	61636	349747	45	1248	15171	$2{,}8x10^2$	$4{,}9x10^1$	$2{,}3x10^1$
	2000	24918	125309	722304	45	1559	20417	$5{,}5x10^2$	$8{,}0x10^1$	$3{,}5x10^1$
	4000	49848	250606	1444310	45	1881	27702	$1{,}1x10^3$	$1{,}3x10^2$	$5{,}2x10^1$
	8000	99760	502494	2901765	45	2234	37561	$2{,}2x10^3$	$2{,}2x10^2$	$7{,}7x10^1$
3	0500	6252	31379	182915	412	6010	58424	$1{,}5x10^1$	$5{,}2x10^0$	$3{,}1x10^0$
	1000	12327	61040	356602	436	9068	92148	$2{,}8x10^1$	$6{,}7x10^0$	$3{,}9x10^0$
	2000	24612	121782	734633	441	13147	146085	$5{,}6x10^1$	$9{,}3x10^0$	$5{,}0x10^0$
	4000	50192	254453	1443411	441	18082	228180	$1{,}1x10^2$	$1{,}4x10^1$	$6{,}3x10^0$
	8000	99967	505068	2910618	441	24879	345762	$2{,}3x10^2$	$2{,}0x10^1$	$8{,}4x10^0$
4	0500	6241	31519	182938	1132	14972	135621	$5{,}5x10^0$	$2{,}1x10^0$	$1{,}3x10^0$
	1000	12507	62978	362585	1686	23495	247632	$7{,}4x10^0$	$2{,}7x10^0$	$1{,}5x10^0$
	2000	24950	125441	721591	2453	35778	421010	$1{,}0x10^1$	$3{,}5x10^0$	$1{,}7x10^0$
	4000	49659	248519	1425675	3333	56455	682468	$1{,}5x10^1$	$4{,}4x10^0$	$2{,}1x10^0$
	8000	99421	499575	2881513	3733	91205	1094944	$2{,}7x10^1$	$5{,}5x10^0$	$2{,}6x10^0$

Table 4.1: Subgraph occurrences and isomorphism tests in networks generated with `WS_MP`.

L	N	Occurrences/ESU Calls			FaSE Calls			Reduction		
		3	4	5	3	4	5	3	4	5
2	0500	7964	102004	1537342	36	496	6391	$2{,}2x10^2$	$2{,}1x10^2$	$2{,}4x10^2$
	1000	17812	287336	5674721	37	531	7289	$4{,}8x10^2$	$5{,}4x10^2$	$7{,}8x10^2$
	2000	46081	1320933	48759292	45	697	10205	$1{,}0x10^3$	$1{,}9x10^3$	$4{,}8x10^3$
	4000	98596	3549800	174567364	45	725	11202	$2{,}2x10^3$	$4{,}9x10^3$	$1{,}6x10^4$
	8000	198237	6399619	287005594	45	733	11691	$4{,}4x10^3$	$8{,}7x10^3$	$2{,}5x10^4$
3	0500	9437	166335	3370142	153	3307	63906	$6{,}2x10^1$	$5{,}0x10^1$	$5{,}3x10^1$
	1000	20321	443706	11905870	164	3905	79105	$1{,}2x10^2$	$1{,}1x10^2$	$1{,}5x10^2$
	2000	44388	1223680	44251833	172	4554	94054	$2{,}6x10^2$	$2{,}7x10^2$	$4{,}7x10^2$
	4000	82022	1838539	56986244	171	5293	85378	$4{,}8x10^2$	$3{,}5x10^2$	$6{,}7x10^2$
	8000	208135	8319608	484153974	189	5496	121755	$1{,}1x10^3$	$1{,}5x10^3$	$4{,}0x10^3$
4	0500	8716	135215	2468242	504	16338	365674	$1{,}7x10^1$	$8{,}3x10^0$	$6{,}7x10^0$
	1000	19389	367315	8555848	519	19237	564962	$3{,}7x10^1$	$1{,}9x10^1$	$1{,}5x10^1$
	2000	42397	984640	29256993	531	20677	753286	$8{,}0x10^1$	$4{,}8x10^1$	$3{,}9x10^1$
	4000	94714	2782787	109164361	560	22197	893388	$1{,}7x10^2$	$1{,}3x10^2$	$1{,}2x10^2$
	8000	185377	5320620	211109317	569	22866	947165	$3{,}3x10^2$	$2{,}3x10^2$	$2{,}2x10^2$

Table 4.2: Subgraph occurrences and isomorphism tests in networks generated with `BA_MP`.

L	N	Occurrences/ESU Calls			FaSE Calls			Reduction		
		3	4	5	3	4	5	3	4	5
2	0500	24942	324411	4859704	25	289	3923	$1{,}0x10^3$	$1{,}1x10^3$	$1{,}2x10^3$
	1000	198690	5159970	154442188	32	541	9739	$6{,}2x10^3$	$9{,}5x10^3$	$1{,}6x10^4$
	2000	1544969	79211486	N/A	39	865	N/A	$4{,}0x10^4$	$9{,}2x10^4$	N/A
	4000	12490927	N/A	N/A	42	N/A	N/A	$3{,}0x10^5$	N/A	N/A
	8000	99818031	N/A	N/A	44	N/A	N/A	$2{,}3x10^6$	N/A	N/A
3	0500	50855	934461	19639536	106	2189	46227	$4{,}8x10^2$	$4{,}3x10^2$	$4{,}2x10^2$
	1000	424189	15932264	684763476	165	4886	153176	$2{,}6x10^3$	$3{,}3x10^3$	$4{,}5x10^3$
	2000	3452422	261823578	N/A	188	8853	N/A	$1{,}8x10^4$	$3{,}0x10^4$	N/A
	4000	27836834	N/A	N/A	253	N/A	N/A	$1{,}1x10^5$	N/A	N/A
	8000	221389736	N/A	N/A	318	N/A	N/A	$7{,}0x10^5$	N/A	N/A
4	0500	95481	2376048	67048834	303	9122	277415	$3{,}2x10^2$	$2{,}6x10^2$	$2{,}4x10^2$
	1000	760923	37900182	N/A	509	22453	N/A	$1{,}5x10^3$	$1{,}7x10^3$	N/A
	2000	6086921	606586567	N/A	727	55822	N/A	$8{,}4x10^3$	$1{,}1x10^4$	N/A
	4000	48171241	N/A	N/A	1077	N/A	N/A	$4{,}5x10^4$	N/A	N/A
	8000	387301814	N/A	N/A	1395	N/A	N/A	$2{,}8x10^5$	N/A	N/A

Table 4.3: Subgraph occurrences and isomorphism tests in networks generated with `ER_MP`.

In the last table (4.3) it is not possible to present all the results (`N/A`) because of computational limitations explained below. However, the existing data is sufficient to realize that it is in this model that there is a bigger reduction of calls.

The reduction that occurs in the number of calls to nauty is so significant in the original method that they are able to make it two to three orders of magnitude faster than ESU. However, the previous tables (together with the following) aim to show that in this new context this may not always happen. The presence of layers means that there are a greater number of subgraph possibilities, exponentially, and this makes subgraph encapsulation, before the calls, not able to aggregate such a large number of subgraphs.

However, as we saw above, this can continue to happen in a certain type of networks. The network in which the subgraphs were structurally most similar to each other was `ER_MP` (least calls). This brings us back to section 4.4 and we realize that quite possibly this is connected to subgraph concentrations. That is, as can be seen in the Figure 4.7 the highest concentration of a subgraph in this network is above 35% followed by 3 close to 20%. Inversely, the `WS_MP` network has no concentration above 10%.

The conclusion that can be drawn from this is that since subgraphs are more concentrated, then the FaSE algorithm will aggregate those subgraphs and consequently reduce the number of isomorphism tests. We will now see if this is actually synonymous with a faster algorithm. In short, we want to conclude that the algorithm can continue to be used for multiplex networks, speeding up the process, even if there are differences between networks. We think they are connected to the amount of isomorphism tests and these are less in networks where subgraphs have higher concentrations.

L	**N**	**Runtime FaSE(s)**			**Runtime ESU(s)**			**Speedup**		
		3	**4**	**5**	**3**	**4**	**5**	**3**	**4**	**5**
2	0500	0,003	0,029	0,298	0,044	0,293	2,065	12,83x	10,11x	6,94x
	1000	0,007	0,051	0,490	0,087	0,605	4,273	12,87x	11,96x	8,72x
	2000	0,014	0,094	0,886	0,175	1,222	8,800	12,82x	12,98x	9,93x
	4000	0,028	0,191	1,508	0,367	2,519	17,812	13,04x	13,17x	11,81x
	8000	0,060	0,366	2,806	0,714	5,055	39,416	11,94x	13,82x	14,05x
3	0500	0,010	0,150	1,898	0,075	0,506	3,884	7,40x	3,36x	2,05x
	1000	0,015	0,236	3,090	0,145	0,975	7,325	9,96x	4,13x	2,37x
	2000	0,023	0,361	5,012	0,292	1,924	14,391	12,65x	5,32x	2,87x
	4000	0,041	0,579	8,133	0,604	4,079	30,533	14,61x	7,05x	3,75x
	8000	0,081	0,945	13,151	1,242	8,045	64,009	15,34x	8,52x	4,87x
4	0500	0,029	0,508	6,319	0,116	0,812	6,932	3,96x	1,60x	1,10x
	1000	0,046	0,801	11,315	0,229	1,578	13,312	5,00x	1,97x	1,18x
	2000	0,071	1,242	19,039	0,458	3,092	25,282	6,48x	2,49x	1,33x
	4000	0,109	2,011	30,717	0,917	6,115	48,510	8,40x	3,04x	1,58x
	8000	0,165	3,337	49,952	1,843	12,210	95,691	11,18x	3,66x	1,92x

Table 4.4: Runtimes of ESU and FaSE algorithms in network generated with `WS_MP` model.

L	N	Runtime FaSE(s)			Runtime ESU(s)			Speedup		
		3	4	5	3	4	5	3	4	5
2	0500	0,004	0,052	0,844	0,054	1,012	19,310	12,83x	19,48x	22,87x
	1000	0,010	0,141	3,094	0,120	2,893	74,114	12,28x	20,59x	23,95x
	2000	0,025	0,608	25,904	0,313	13,653	703,894	12,61x	22,46x	27,17x
	4000	0,057	1,805	102,547	0,671	37,269	>30m	11,75x	20,65x	N/A
	8000	0,130	3,525	187,023	1,372	67,486	>30m	10,52x	19,15x	N/A
3	0500	0,008	0,162	3,939	0,109	2,669	68,673	13,49x	16,50x	17,44x
	1000	0,016	0,335	10,175	0,234	7,070	245,930	15,11x	21,11x	24,17x
	2000	0,032	0,820	33,738	0,515	19,659	934,170	16,29x	23,96x	27,69x
	4000	0,074	1,331	46,915	0,969	29,681	>30m	13,15x	22,30x	N/A
	8000	0,162	5,972	418,647	2,482	136,557	>30m	15,28x	22,87x	N/A
4	0500	0,018	0,567	15,697	0,160	3,233	74,491	9,11x	5,70x	4,75x
	1000	0,027	0,831	28,179	0,353	8,723	255,741	13,11x	10,50x	9,08x
	2000	0,045	1,364	54,445	0,770	23,337	876,066	17,25x	17,11x	16,09x
	4000	0,089	2,924	139,519	1,738	66,514	>30m	19,60x	22,75x	N/A
	8000	0,176	5,413	255,926	3,414	129,012	>30m	19,38x	23,83x	N/A

Table 4.5: Runtimes of ESU and FaSE algorithms in network generated with BA_MP model.

L	N	Runtime FaSE(s)			Runtime ESU(s)			Speedup		
		3	4	5	3	4	5	3	4	5
2	0500	0,011	0,150	2,492	0,184	3,161	61,118	17,02x	21,03x	24,52x
	1000	0,101	2,868	91,842	1,583	53,877	1785,615	15,65x	18,79x	19,44x
	2000	0,821	52,324	>30m	12,959	911,435	>30m	15,78x	17,42x	N/A
	4000	7,654	>30m	>30m	111,055	>30m	>30m	14,51x	N/A	N/A
	8000	74,951	>30m	>30m	898,356	>30m	>30m	11,99x	N/A	N/A
3	0500	0,027	0,557	13,745	0,642	15,133	387,981	23,98x	27,19x	28,23x
	1000	0,258	10,280	488,142	5,951	276,065	>30m	23,08x	26,85x	N/A
	2000	2,496	193,393	>30m	52,592	>30m	>30m	21,07x	N/A	N/A
	4000	22,188	>30m	>30m	429,689	>30m	>30m	19,37x	N/A	N/A
	8000	202,385	>30m	>30m	>30m	>30m	>30m	N/A	N/A	N/A
4	0500	0,059	1,793	61,878	2,011	61,230	>30m	33,94x	34,15x	N/A
	1000	0,532	29,734	>30m	18,248	1054,027	>30m	34,27x	35,45x	N/A
	2000	5,094	499,324	>30m	153,938	>30m	>30m	30,22x	N/A	N/A
	4000	42,683	>30m	>30m	1248,073	>30m	>30m	29,24x	N/A	N/A
	8000	378,918	>30m	>30m	>30m	>30m	>30m	N/A	N/A	N/A

Table 4.6: Runtimes of ESU and FaSE algorithms in network generated with ER_MP model.

Table cells containing '>30m' information correspond to processes that were terminated because the execution time exceeded 30 minutes and therefore those containing 'N/A' are those where is not possible to calculate speedup.

Therefore, what can be concluded from the six tables presented above is that there is a connection between the number of isomorphism tests and the execution speed of the algorithm. The fact is, this was the conclusion of the authors of FaSE algorithm. However, with the addition of layers this could not happen since the exponential growth of different subgraphs could lead to their encapsulating, before calling nauty, so small that it became insignificant and without any gains compared to ESU.

What can easily be proved is that there is always a gain from FaSE over ESU, in the Tables 4.4, 4.5 and 4.6, even in this new context . This improvement is because of the difference between the amount of isomorphism tests made by each. On the `WS_MP` network, on average, these tests were reduced by 1,25x10^2, the `BA_MP` network by 1,81x10^3 and the `ER_MP` by 1,64x10^5. This reduction (with the orders of magnitude increasing) is inverse by improvements that were on average 7.47x, 17.28x and 22.63x following the same order.

Another connection we want to make is between isomorphic tests reductions (which happens because subgraphs are first stored in a g-trie and those that are structurally the same correspond to the same leaf and therefore the same test) and the concentration of subgraphs. All subgraphs with relevant concentrations of networks with two layers are shown in the Figure 4.7 (the other subgraphs have values near zero). We realize, as said before, that networks with the highest concentration are the ones where FaSE has the most improvement.

If we go back and look closely at the Tables 4.1, 4.2, and 4.3 we realize in which networks there is a bigger concentration of subgraphs. Not comparing the type of network, but the amount of layers. That is, as can easily be observed, the orders of magnitude of the reductions are higher in networks with two layers (which is expected since the increase of subgraphs is exponential with layers). This is why the presented fingerprint is from the networks with two layers because it is more evident.

Now we detail the parameters we used to generate each of the networks. In `ER_MP` network as probability of connection between two nodes we use 1%. If we increase this value, it would also increase the overlap of edges which would lead to different concentration values. However, with 10%, the most present graphs would be the same but with lower concentration values. In the `BA_MP` model, we used an attachment parameter of 2 (not making the network too small or too dense as `ER_MP`). The subgraphs found are the same because the topology of this network.

In the last case we used a network with 3 neighbors on each side and a rewiring probability of 20%. We experimented with the double neighbors and with different probability values. The subgraphs found are practically the same and with similar concentration values. Therefore, we decided to present the values of three models that we consider sufficiently different to compare. Each with its own topology and different subgraph quantities to understand the behavior of the algorithms.

For the sake of those who are reading this thesis it is not possible to present all the values because the parameters are so many, resulting in so much data, that it is not a good idea to present everything here. But we still have more three tables.

4.6 Runtime Behavior

We saw earlier how one algorithm behaves relative to another. However, we want to evaluate the performance individually and we will do it to FaSE. We want to understand how scalable it can be. We assure that our machine is not a five star but we are very comfortable with it since the results are quite satisfactory and keeping in mind that they could be even more.

Importantly, once again, the algorithm is based on the enumeration of subgraphs. This means that this is a process from which it can not escape. Therefore, graphs that contain more occurrences will necessarily take longer because they must all be considered to calculate the exact frequency. Another important detail is that these occurrences refer to the overlap network.

The first two sections of the Tables 4.7, 4.8 and 4.9 were already contained in the previous tables. However, we put it back here because it is these numbers that allows to calculate what we want to highlight. Our algorithm maintains a constant ratio with very close orders of magnitude. Depending on the model and with the increase of layers there is a slight variation of behaviors.

Models `WS_MP` and `BA_MP` create networks with very similar numbers of subgraph occurrences in the different layers. That is, due to the way it is generated, they maintain an identical structure independent of the layer differing only by how many layers are distributed the edges. This is why there is a slight decrease in the ratio with increasing layers. The occurrences are similar in the overlap network but more runtime is required to reconstruct each subgraph.

L	N	Occurences			Runtime FaSE(s)			Graphs/Sec		
		3	4	5	3	4	5	3	4	5
2	0500	6184	30707	173826	0,003	0,030	0,299	$5{,}5x10^7$	$9{,}6x10^7$	$1{,}7x10^6$
	1000	12354	61315	347744	0,007	0,051	0,489	$5{,}5x10^7$	$8{,}3x10^7$	$1{,}4x10^6$
	2000	24786	123866	708962	0,014	0,093	0,846	$5{,}5x10^7$	$7{,}5x10^7$	$1{,}2x10^6$
	4000	49761	249723	1436261	0,028	0,191	1,508	$5{,}6x10^7$	$7{,}6x10^7$	$1{,}1x10^6$
	8000	99705	501766	2894590	0,060	0,366	2,814	$6{,}0x10^7$	$7{,}3x10^7$	$9{,}7x10^7$
3	0500	6189	30787	175926	0,010	0,150	1,876	$1{,}6x10^6$	$4{,}9x10^6$	$1{,}1x10^5$
	1000	12448	62471	362514	0,015	0,234	3,056	$1{,}2x10^6$	$3{,}7x10^6$	$8{,}4x10^6$
	2000	24819	124110	725798	0,023	0,360	4,974	$9{,}4x10^7$	$2{,}9x10^6$	$6{,}9x10^6$
	4000	49999	252279	1447058	0,045	0,580	8,147	$9{,}0x10^7$	$2{,}3x10^6$	$5{,}6x10^6$
	8000	99920	504139	2911786	0,082	0,940	13,142	$8{,}2x10^7$	$1{,}9x10^6$	$4{,}5x10^6$
4	0500	6263	31740	184078	0,029	0,514	6,413	$4{,}6x10^6$	$1{,}6x10^5$	$3{,}5x10^5$
	1000	12399	61914	353826	0,045	0,797	11,023	$3{,}7x10^6$	$1{,}3x10^5$	$3{,}1x10^5$
	2000	24770	123605	706503	0,071	1,249	18,832	$2{,}9x10^6$	$1{,}0x10^5$	$2{,}7x10^5$
	4000	49923	251524	1453277	0,109	2,006	30,898	$2{,}2x10^6$	$8{,}0x10^6$	$2{,}1x10^5$
	8000	99658	501387	2892922	0,164	3,341	49,944	$1{,}6x10^6$	$6{,}7x10^6$	$1{,}7x10^5$

Table 4.7: Subgraphs found per second in network generated with `WS_MP` model using `FaSE`.

L	N	Occurences			Runtime FaSE(s)			Graphs/Sec		
		3	4	5	3	4	5	3	4	5
2	0500	8953	144951	2749850	0,005	0,069	1,416	$5{,}2x10^7$	$4{,}8x10^7$	$5{,}2x10^7$
	1000	19381	382443	9388452	0,010	0,179	4,883	$5{,}4x10^7$	$4{,}7x10^7$	$5{,}2x10^7$
	2000	44603	1216333	43112113	0,024	0,567	23,088	$5{,}4x10^7$	$4{,}7x10^7$	$5{,}4x10^7$
	4000	97266	3342397	158009576	0,056	1,706	94,200	$5{,}8x10^7$	$5{,}1x10^7$	$6{,}0x10^7$
	8000	210526	8482889	494737974	0,135	4,830	312,850	$6{,}4x10^7$	$5{,}7x10^7$	$6{,}3x10^7$
3	0500	9380	164468	3360745	0,008	0,160	3,958	$8{,}6x10^7$	$9{,}7x10^7$	$1{,}2x10^6$
	1000	20784	462667	12679959	0,016	0,346	10,614	$7{,}6x10^7$	$7{,}5x10^7$	$8{,}4x10^7$
	2000	46170	1330374	50183307	0,033	0,883	37,622	$7{,}1x10^7$	$6{,}6x10^7$	$7{,}5x10^7$
	4000	91880	2642269	103388195	0,075	1,828	82,281	$8{,}2x10^7$	$6{,}9x10^7$	$8{,}0x10^7$
	8000	204779	7684994	412661990	0,161	5,502	354,443	$7{,}8x10^7$	$7{,}2x10^7$	$8{,}6x10^7$
4	0500	8812	135461	2428987	0,018	0,570	15,228	$2{,}0x10^6$	$4{,}2x10^6$	$6{,}3x10^6$
	1000	20828	445876	11472611	0,028	0,898	31,698	$1{,}3x10^6$	$2{,}0x10^6$	$2{,}8x10^6$
	2000	46708	1277417	43599044	0,048	1,603	68,554	$1{,}0x10^6$	$1{,}3x10^6$	$1{,}6x10^6$
	4000	101531	3493515	156790144	0,095	3,487	186,159	$9{,}3x10^7$	$1{,}0x10^6$	$1{,}2x10^6$
	8000	194800	6402993	296271341	0,191	6,202	339,171	$9{,}8x10^7$	$9{,}7x10^7$	$1{,}1x10^6$

Table 4.8: Subgraphs found per second in network generated with `BA_MP` model using `FaSE`.

L	N	Occurrences			Runtime FaSE(s)			Graphs/Sec		
		3	4	5	3	4	5	3	4	5
2	0500	24511	316266	4617683	0,011	0,145	2,389	$4{,}4x10^7$	$4{,}6x10^7$	$5{,}2x10^7$
	1000	197641	5124396	152099919	0,098	2,836	90,139	$5{,}0x10^7$	$5{,}5x10^7$	$5{,}9x10^7$
	2000	1549359	79561466	>30m	0,842	53,062	>30m	$5{,}4x10^7$	$6{,}7x10^7$	N/A
	4000	12515543	>30m	>30m	7,723	>30m	>30m	$6{,}2x10^7$	N/A	N/A
	8000	99897617	>30m	>30m	74,502	>30m	>30m	$7{,}5x10^7$	N/A	N/A
3	0500	51489	949208	20004633	0,027	0,568	13,985	$5{,}3x10^7$	$6{,}0x10^7$	$7{,}0x10^7$
	1000	420464	15721812	672846663	0,257	10,163	480,632	$6{,}1x10^7$	$6{,}5x10^7$	$7{,}1x10^7$
	2000	3440376	260557281	>30m	2,486	192,688	>30m	$7{,}2x10^7$	$7{,}4x10^7$	N/A
	4000	27630410	>30m	>30m	22,243	>30m	>30m	$8{,}1x10^7$	N/A	N/A
	8000	221100753	>30m	>30m	206,086	>30m	>30m	$9{,}3x10^7$	N/A	N/A
4	0500	96115	2404564	68229486	0,059	1,796	62,157	$6{,}1x10^7$	$7{,}5x10^7$	$9{,}1x10^7$
	1000	755209	37449765	>30m	0,537	28,336	>30m	$7{,}1x10^7$	$7{,}6x10^7$	N/A
	2000	6036297	598987400	>30m	4,966	498,308	>30m	$8{,}2x10^7$	$8{,}3x10^7$	N/A
	4000	48189368	>30m	>30m	42,882	>30m	>30m	$8{,}9x10^7$	N/A	N/A
	8000	387764843	>30m	>30m	380,132	>30m	>30m	$9{,}8x10^7$	N/A	N/A

Table 4.9: Subgraphs found per second in network generated with `ER_MP` model using `FaSE`.

The generation of model `ER_MP` is completely random in each layer which makes the occurrences of subgraphs much larger than in other models. That is, with the increase of layers, also increase the subgraphs and therefore the previous effect is not verified.

4.7 Comparison with Related Work

Everything we have seen so far lacks something as important as confirmation of the results. If regarding runtime is more or less obvious how to turn a timer on and off then we need to have a solution that allows to confirm subgraph frequencies. We do not know any available work that does the same as ours, so we had to workaround for find a viable comparison and confirm that we are getting correct values.

The idea used was the same as we used earlier for the network visualization tool. A new color is used for represent two nodes that are connected in more than one layer. In fact this is a solution that becomes unfeasible because a new color is required for each combination of the layer set. If for two layers we only need one extra color for three we would already need seven colors in total.

However, this is enough for our purpose. We present a small example in the Figure 4.8 of our output compared to the output of the FANMOD tool [56]. The two columns are the original output in html format produced by both methods. They found the same subgraphs and have the same frequency values. The classes are defined with node isomorphism since the other tool is not prepared for isomorphism considering layers.

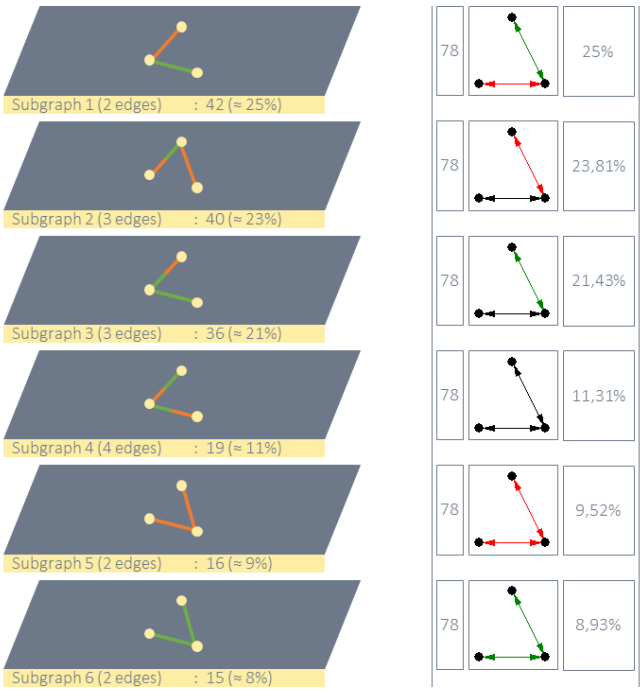


Figure 4.8: Subgraph frequencies compared with another method. Both original outputs are in html format. Results obtained from a `BA_MP` network with 50 nodes and $m = 1$.

Although our algorithm is considerably faster we consider it is not fair to compare our tool with another whose purpose is not to do the same as us. However, we would have liked to compare ourselves to others if it had been possible but we do not know any source code or tool capable of doing the same.

4.8 Subgraphs Injection

Since we have already shown the ability to generate different synthetic networks, how our algorithms behave in terms of runtime, the difference between them and that we get correct values of frequencies we need to prove the ability to detect motifs. For this, we decided to use the subgraph injection technique in a given network and check the output differences. We will also use this section to show that the different types of isomorphism are calculated correctly and the effect of a different null model.

We generated a network with 500 nodes and 2 layers according to the `BA_ML` model. Any of the models presented above could be used but we chose this one due to the type of subgraph we will add and how we can do it. We just need to go to a hub or to the end of a path, and connect two nodes in both layers that are the endpoints, which are not connected between them. Knowing that this addition of a subgraph will always change the original count of other subgraphs as well, we tried to minimize this problem with this strategy present in Figure 4.9.

Figure 4.9: Subgraph injection method. Search for a subgraph like the one on the left. Two edges on the orange layer and none on the green layer. Then, between the two nodes that have no connection yet, add edges in both layers.

We added 200 edges to the network, i.e. 100 triangles of this type. However, the count may not reflect this because due to the connection to other adjacent nodes that may form some extra triangles (in our case just one more). What is certain is that the subgraph on the left of the Figure 4.9 will have a smaller frequency, while the other has more. In this particular injection, count differences are not significant and are reflected in a slight increase of subgraphs that contain an overlapping edge.

Figure 4.10: Subgraph injected in the network, A. The subgraph B is isomorphic to A if we consider the node-layer isomorphism.

Both subgraphs of Figure 4.10 (they are different in node isomorphism but equal in node-layer) have a frequency of 7 in the original network. The highest frequencies are in the order of thousands and there are others in the hundreds. Therefore, we can say that they are subgraphs whose presence is insignificant. The purpose of this experiment is to verify if with the injection they appear as motifs. We injected a quantity of subgraphs in which the frequency remains very low compared with the other frequencies.

It is possible to confirm that all the numbers obtained are as expected as shown in the Table 4.10. The frequency of class A or B in node-layer isomorphism (they are the same) is equal to the sum of the classes if we consider the layers are differentiated as in node isomorphism. Notice that the injection process was made only with the type A, which is reflected in the values.

	Frequencies		
	Node		**Node-Layer**
	A	**B**	**A or B**
before	7	7	14
after	108	7	115

Table 4.10: Subgraphs frequencies before and after the injection process.

Being certain that our injection of subgraphs was well done we want to affirm that our methods are able to return the inserted subgraph as motif of the new network. We recall that the frequency of some subgraphs is in the order of thousands and our subgraph has a very low frequency (the concentration value remains below the 1%) so we need other metric and we used z-score, which is defined as this equation. $z\text{-}score = (f_{original} - \overline{f_{random}})/\sigma(f_{original})$

In the original network the subgraph has a z-score below zero (with both null models) which makes sense given its almost nonexistence. This happens in both types of subgraphs and will continue to happen to B because this one has not been injected. However, after the process the values are different. In the null model one, where edge exchanging occurs layer-by-layer, the subgraphs with the highest frequency also have the highest z-score.

With the same model, the value of the subgraph injected increased to positive values and is the fourth with the highest value. In the first three positions, two are the most represented subgraphs, and only one is a special case. In the null model two, developed by us for multiplex, the highest z-score refers to the subgraph we injected. In the second position, the special case also returned by the other null model.

	Z-Scores		
	Null Model One		
	A	**B**	**A or B**
before	-1,33	-1,22	-1,61
after	10,41	-2,99	5,07 (4th)
	Null Model Two		
	A	**B**	**A or B**
before	-0,30	0,22	-0,07
after	25,55	-0,45	20,39 (1st)

Table 4.11: Subgraphs z-scores, before and after, with two null models.

After the experience we can say that both models meet the expectations and return the subgraph as motif. However, we are more satisfied with the second null model because it was able to return it in the first position without losing other important subgraphs. Therefore, the model we had thought fits this new context.

4.9 Real Networks

Since we have already evaluated our approach in synthetic networks, with real networks we have done a different work that is not focused in the same aspects, and here will focus more on an application view, showing some of the insight that can be gained by using an approach such as ours. We graphically present the most interesting subgraphs that we found. First, we detail the networks we used, some of which were created by ourselves due to the difficulty of finding diverse real networks.

4.9.1 Description

london network Nodes are transport stations in London and edges are routes between stations. Underground, Overground and DLR are the layers considered. This network was found at [17].

porto network Nodes are transport stations in Porto and edges are routes between them using Bus or Metro lines. A layer built by us and another by a student from the same department.

referees network Nodes are football referees from Viana do Castelo, a northern city of Portugal, where this thesis's author was born. Two referees are connected if they refereed a game together last season (18/19). Layers are the formation age levels of the players.

students network Nodes are students from this department since 2014. They are connected to courses if they studied it in one of the five years. It is a bipartite graph and the layer refers to the years.

flights network The network is composed by thirty-seven different layers each one corresponding to a different airline operating in Europe. Each node represents one airport. This network was found at [11].

arxiv network The multiplex consists of layers corresponding to different arXiv categories. To restrict the analysis it is only included papers with "networks" in the title or abstract. This network was found at [18].

4.9.2 Subgraph Census

The Table 4.13 shows how many occurrences there are of each subgraph size. This number does not depend on the method used, ESU or FaSE, and we confirmed that the results are the same in both. Naturally, the larger the size of the subgraph, the more occurrences that exist in a network. As expected, the growth is exponentially. In networks `london`, `porto` and `referees` the average growth rate is 2.9, 3.0 and 11.5 per layer.

Network	Nodes	Edges	Layers	Directed
london	369	882	3	No
porto	1024	2663	2	Yes
referees	131	2220	4	No
students	443	10900	5	No
flights	450	7176	37	No
arxiv	14488	118052	13	No

Table 4.12: Real networks used to test our approach and the features of each.

Network	Subgraph Size	Occurrences	Isomorphic Classes Type One	Isomorphic Classes Type Two
london	3	737	13	6
	4	1680	45	22
	5	4567	150	79
	6	13581	477	285
	7	42102	1442	970
	8	133347	4211	3111
	9	428173	12069	9588
porto	3	2331	39	36
	4	5807	287	276
	5	17891	1683	1658
	6	63215	8538	8482
referees	3	4966	317	71
	4	55231	6519	2348
	5	652632	105291	52971
students	3	559049	349	28
	4	56581790	14799	1983
flights	3	101144	11794	181
arxiv	3	313829	9067	456

Table 4.13: Occurrence of subgraphs and isomorphic classes of real networks.

The last two columns have the number of isomorphic classes that were found in each network. Class one represents node isomorphism where only nodes can be exchanged to verify that two structures are equal. The last column contains the values of isomorphic classes of type two. That is, where layers can also be exchanged.

The results in the Table 4.13 were all calculated in less than 30 minutes and we opted to stop searches of subgraphs that exceeded this time.

As explained before, the number of isomorphic classes of the node-layer isomorphism will be necessarily smaller. In the case of the transportation networks the difference is small and that is because they have their own topology where each layer can be much different from the others with few overlapping points.

It should be noted, again, that our work goes beyond running times. Multiplex networks have not yet been the subject of intensive study and the realization of this thesis allows to clarify and develop new concepts. Furthermore, now we can know the subgraphs that are present in the networks in a simple way that was not possible before. We searched for subgraphs of various sizes and show the ones we consider most interesting.

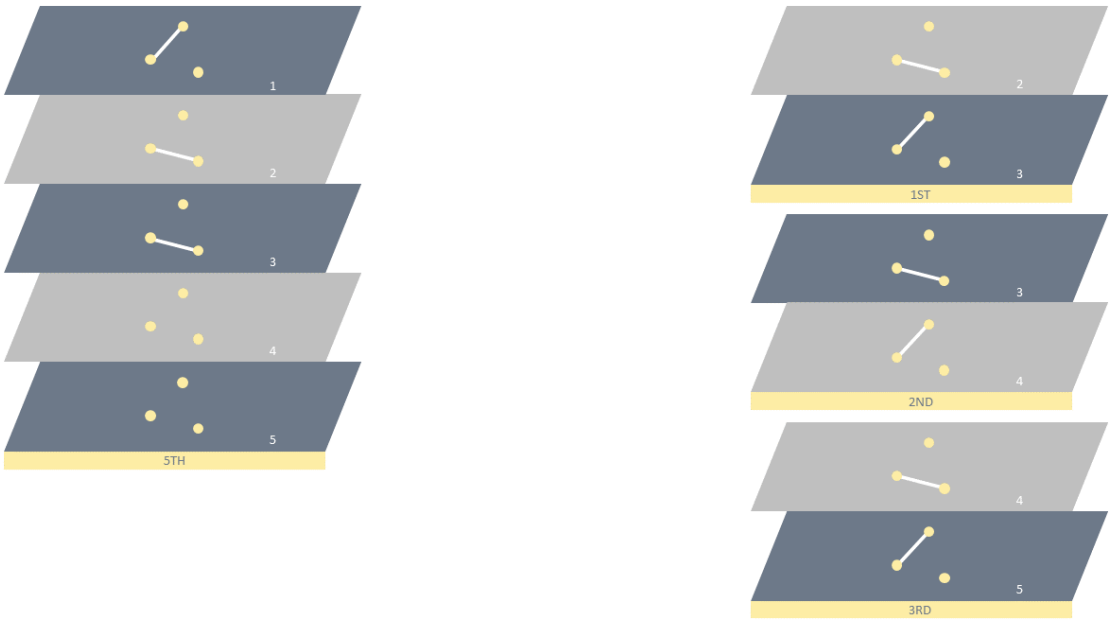


Figure 4.11: Subgraphs found in `students` network. The three most frequent subgraphs on the right side and the most surprising, in the fifth position, on the left. The subgraphs present on right, by representation issues, do not contain empty layers.

The subgraphs shown in Figure 4.11 were found in the students network. The network is composed by students from the computer science department of the Faculty of Sciences of the University of Porto and the courses of the same department, where each student is connected to the courses they attended. Therefore, this network is bipartite. Each layer represents the school year since 2014 until now forming a set of five layers.

Since students are connected to different courses in different years, it is normal that the most represented subgraphs are of the type shown on the right side of the figure. That is, a student with a "normal" academic path will be connected to a node representing the courses of one year and in another layer connected to another node. Therefore, the three most represented subgraphs have this pattern but in different layers. Another expected pattern is for a student to have connections to different courses in the same year because it necessarily happens. This pattern is the fourth with the highest frequency.

However, the surprise comes in fifth position. The pattern discovered by our approach indicates that there are many students who are connected to one course in one year and in the next year again. That is, students often fail courses in this department. In particular, it is more common fail it in the second year than first what makes it even more interesting. The subgraphs in this figure refer to node isomorphism.

We know there are other ways to get this information, such as students failure rates, but we want to prove the functionality of our approach.

Another result that proves the practicality of our work is the subgraph obtained in the referee network. This network represents the set of referees of the Viana do Castelo football association and two referees are connected together if they refereed any game together in the 18/19 season. The layers refer to the ranks that exist in football, under15, under17, under19 and seniors. For context, referees usually have teams and they officiate together for an entire season and are only replaced by someone if they are unavailable on a weekend.

Figure 4.12: Subgraphs found in `referees` network. The most represented subgraphs and a schematic representation of this particular network.

Since referees normally officiate together, the network formed is expected to be similar to the one schematically drawn in the Figure 4.12. In the network it is possible to see the formation of triangles with connections in all layers. Moreover, there are edges between one element of a triangle and two other nodes of another triangle that occur when one referee replace another in a different team. This figure represents the compact version of our approach and not the expanded one. That is, the layers are represented by colors on the edges.

Therefore, if we order the subgraphs with size three obtained by the number of edges, we first get the subgraph with all the connections which is represented on the right side of the Figure 4.12. The frequency corresponds to the number of referee teams in activity. However, the most frequent subgraph are the two represented in the left panel due to the specific topology of this network.

Another network analyzed was the flights network in europe where each node represents an airport. Each layer represents an airline. The results presented in the Figure 4.13 represent isomorphic classes from node-layer isomorphism, that is, it does not matter the airline itself.

The most frequent subgraphs are the lines, not the triangles. In other words, this is due to the existence of hubs where usually two larger airports are connected and then connected to smaller airports around them, which do not have connections to larger airports that are far away. Therefore, this network is similar to `BA_MP`. In addition, the second flight (edge) is more often done by a different airline being the most represented subgraph in the whole network. Subgraphs where edges overlap, that is when there are more than one airline flying between two airports, are less frequent. This leads to the conclusion that there is not much competition but may be due again to hubs where connections to smaller airports are usually only made by domestic airlines. Analyzing the frequencies of the triangles, it appears that the existence of a triangle in which all connections are made by the same airline, A, have half the frequency of B.

Figure 4.13: Subgraphs found in `flights` network. The subgraphs with the highest frequency and the most common triangles.

In the Figure 4.14 are represented subgraphs of two distinct networks but of the same type, transportation. On the left are subgraphs of the London network where node-layer isomorphism was used. On the right are two expanded subgraphs of the network of Porto city where the upper layer shows the Bus and the lower layer the Metro lines. In this particular subgraph the direction of the edges is not represented because in all of them it occurs in both directions and this is the way we use to represent this type of edges. Moreover, this is a directed network that proves that our tool is capable of handling this type of network.

The most frequent subgraphs are the ones that are contained in a single layer. This is expected because most stations usually have only one type of transport. These subgraphs also indicate that there are more lines without diverging than finding a station that is connected to other three. Moreover, stations containing more than one transportation method are more frequent in the middle of a line than at the end. In the city of Porto it is possible to verify that one of the subgraphs most represented, A, with edges in the two layers, is a metro connection joining two stations with bus lines. The other subgraph, B, has the same isomorphic class as the fourth in the city of London.

Figure 4.14: Subgraphs found in transportation networks. On the left the city of `london` and on the right the `porto` city.

4.9.3 Network Motifs

The motifs shown in the Figure 4.15 were captured in the London city transportation network. In this network there is a layer that stands out, with more connections compared to the others. The layer that aggregates all the underground lines is clearly denser than the other two, the layers with connections by overground and by dlr. Another important point is that, being a transport network, the overlap of connections is low. This has a key role when generating random networks and discovering subgraphs there because, as in the original network, there are many connections that are only made in one layer. Random networks were obtained using null model one where the edges of each layer are randomly exchanged.

Figure 4.15: Motifs found in `london` network. On the left the motifs of size five and on right the motif with size three.

Therefore it is not surprising that the two highest z-score motifs belongs only to one layer, with no edges in the others. It is possible to think that this analysis could be done with single layer tools but it is not true. These tools did not tell us if the returned subgraphs would actually be the same when considering multiple layers because there would always be the possibility of edges overlapping. Thus these doubts go away with our method. What is surprising about the first two subgraphs is that they both contain a triangle in their substructure. In fact, the second contains two triangles. The presence of triangles, more than expected, is confirmed because this subgraph is the one with the highest z-score when analyzing subgraphs of size three.

Looking at the results obtained in the referee networks, Figure 4.16, we see that the most over-represented subgraphs are those that are also more frequent because the special characteristics of the network, as explained in the schematic network of Figure 4.12, make that there are too many edges to two nodes that are connected in all layers. Knowing that the isomorphism of this image is node-layer then we see that the two motifs found represent a referee who sporadically replace one member of the team in one or two ranks.

However, knowing that the network contained a considerable number of triangles in which there were edges at all layers, we were not satisfied with the results because we were convicted that this one is a motif. Surprisingly, in many randomly generated networks such subgraph was never found and therefore could not be detected as a motif. That is for this cases that we developed a new null model. To our satisfaction, this subgraph became the one with the highest z-score.

Figure 4.16: Motifs found in `referees` network. On the left the using null model one and on left the motif found with null model two.

Chapter 5

Conclusions

Our initial time was spent in a careful study of the existing research, allowing a good grasp on the existing state-of-the-art. This helped in establishing the central points to produce a work plan capable of leading towards the proposed goal. I remember that our initial target was to get subgraphs in multiplex networks, and this was completely achieved without spending much more time compared to classical network algorithms if we consider that we have layers making the problem difficult.

That is the point we want to make. We get the frequencies of these new subgraphs in these new networks accurately and correctly. However, as we like to do good and better work, we try to improve our first attempt using a new algorithm. We have adapted FaSE expecting significant runtimes reductions. Taking advantage of the subgraphs have substructures that are equivalent, it is able to discard most of the isomorphism tests required to identify each isomorphic class, which is the main bottleneck.

However, the presence of layers makes the similarities between the subgraphs much smaller. Then, the reduction of isomorphic tests is not as significant as in classical networks. Another problem is that these subgraphs have much larger sizes which increases the memory usage. Therefore, the improvements that existed do not allow us to be fully satisfied and lead us to look at what can be done to achieve even more positive results in section 5.1.

Still, we are convinced of the important work we have done. First, we apply mathematical theories, which involves the super nodes, that we are unaware of having been used so far to prove their practical validity. We used novel concepts, like multiplex subgraph, and come up with novel ideas, namely a new null model or the new network models all adapted for multiplex. This allowed us to go around the problem and obtain, visualize and analyze subgraph patterns in networks with new features so unexplored but totally promising. This is proved by the huge number of articles, focused on networks with layers, published during the realization of this thesis.

5.1 Future Work

While we firmly believe we have made a valid contribution and advanced the state-of-the art in this topic, there are many areas in which our approach can be improved. We finish this thesis precisely by giving a list of some possible ideas for future work.

Multilayer

Because of their slightly smaller complexity when compared to fully general multilayer networks with any number of aspects, multiplex were our top priority. During development we had this in mind and therefore with slight changes we believe that it is possible to be able to analyze the most general networks.

Memory

It is necessary to guarantee a representation of the network in an effective way, so that the problems of computation do not begin here. Some results cannot be expanded because of this. The idea is to compare the memory used in different representations and test a new hybrid approach [37].

Datasets

Unfortunately, data sets that meet these characteristics are still scarce, which implied an additional effort during the project in collecting new real data sets. We want to expand our repository, with better networks.

Runtimes

Improve the implementation runtime. Our idea will be to study more and better the valid options to use with multilayer networks. We need to understand which ones can work best with these networks or to develop a new labeling function for FaSE that guarantees a greater overlap of substructures. Moreover, we would like to consider approximate schemes (that allow to trade precision for better execution time) and parallel approaches (that would leverage the power of multiple processors and high performance systems to obtain considerable speedups).

Visualization

Despite the good work done to visualize the subgraphs found we believe that there are significant improvements that can and should be made. Mainly, the creation of a tool that dynamically changes the options that allow modifying the way results, which were calculated before, are displayed.

Publishing

Given that we want our work to be helpful for other researchers, we are thinking on producing at least one scientific publication describing our approach.

Bibliography

[1] Nesreen K Ahmed, Jennifer Neville, Ryan A Rossi, and Nick Duffield. Efficient graphlet counting for large networks. In *Data Mining (ICDM), 2015 IEEE International Conference on*, pages 1–10. IEEE, 2015.

[2] Alberto Aleta, Sandro Meloni, and Yamir Moreno. A multilayer perspective for the analysis of urban transportation systems. *Scientific reports*, 7:44359, 2017.

[3] David Oliveira Aparício, Pedro Manuel Pinto Ribeiro, and Fernando Manuel Augusto da Silva. Parallel subgraph counting for multicore architectures. In *Parallel and Distributed Processing with Applications (ISPA), 2014 IEEE International Symposium on*, pages 34–41. IEEE, 2014.

[4] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[5] Albert-László Barabási et al. *Network science*. Cambridge university press, 2016.

[6] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: an open source software for exploring and manipulating networks. In *Third international AAAI conference on weblogs and social media*, 2009.

[7] Prithwish Basu, Ravi Sundaram, and Matthew Dippel. Multiplex networks: A generative model and algorithmic complexity. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pages 456–463. ACM, 2015.

[8] Federico Battiston, Vincenzo Nicosia, Mario Chavez, and Vito Latora. Multilayer motif analysis of brain networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(4): 047404, 2017.

[9] Stefano Boccaletti, Vito Latora, Yamir Moreno, Martin Chavez, and D-U Hwang. Complex networks: Structure and dynamics. *Physics reports*, 424(4-5):175–308, 2006.

[10] Ed Bullmore and Olaf Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature Reviews Neuroscience*, 10(3):186, 2009.

[11] Alessio Cardillo, Jesús Gómez-Gardenes, Massimiliano Zanin, Miguel Romance, David Papo, Francisco Del Pozo, and Stefano Boccaletti. Emergence of network features from multiplexity. *Scientific reports*, 3:1344, 2013.

[12] Yinghan Chen and Yuguo Chen. An efficient sampling algorithm for network motif detection. *Journal of Computational and Graphical Statistics*, 27(3):503–515, 2018.

[13] Han-Yu Chuang, Eunjung Lee, Yu-Tsueng Liu, Doheon Lee, and Trey Ideker. Network-based classification of breast cancer metastasis. *Molecular systems biology*, 3(1):140, 2007.

[14] Emanuele Cozzo, Guilherme Ferraz de Arruda, Francisco Aparecido Rodrigues, and Yamir Moreno. Multiplex networks: Basic definition and formalism, 2018.

[15] Manlio De Domenico. Multilayer modeling and analysis of human brain networks. *GigaScience*, 6(5):1–8, 2017.

[16] Manlio De Domenico, Albert Solé-Ribalta, Emanuele Cozzo, Mikko Kivelä, Yamir Moreno, Mason A Porter, Sergio Gómez, and Alex Arenas. Mathematical formulation of multilayer networks. *Physical Review X*, 3(4):041022, 2013.

[17] Manlio De Domenico, Albert Solé-Ribalta, Sergio Gómez, and Alex Arenas. Navigability of interconnected networks under random failures. *Proceedings of the National Academy of Sciences*, 111(23):8351–8356, 2014.

[18] Manlio De Domenico, Andrea Lancichinetti, Alex Arenas, and Martin Rosvall. Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems. *Physical Review X*, 5(1):011027, 2015.

[19] Manlio De Domenico, Mason A Porter, and Alex Arenas. Muxviz: a tool for multilayer analysis and visualization of networks. *Journal of Complex Networks*, 3(2):159–176, 2015.

[20] Ahmad Naser eddin and Pedro Ribeiro. Scalable subgraph counting using mapreduce. In *Proceedings of the Symposium on Applied Computing*, SAC '17, pages 1574–1581. ACM, 2017.

[21] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[22] Michael R Garey and David S Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. Freeman, 1979.

[23] Joshua A Grochow and Manolis Kellis. *Network motif discovery using subgraph enumeration and symmetry-breaking*. Springer, 2007.

[24] Roger Guimera, Stefano Mossa, Adrian Turtschi, and LA Nunes Amaral. The worldwide air transportation network: Anomalous centrality, community structure, and cities' global roles. *Proceedings of the National Academy of Sciences*, 102(22):7794–7799, 2005.

[25] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[26] Tomaž Hočevar and Janez Demšar. A combinatorial approach to graphlet counting. *Bioinformatics*, 30(4):559–565, 2014.

[27] Chuntao Jiang, Frans Coenen, and Michele Zito. A survey of frequent subgraph mining algorithms. *The Knowledge Engineering Review*, 28(1):75–105, 2013.

[28] Sahand Khakabimamaghani, Iman Sharafuddin, Norbert Dichter, Ina Koch, and Ali Masoudi-Nejad. Quatexelero: an accelerated exact network motif detection algorithm. *PloS one*, 8 (7):e68073, 2013.

[29] Mikko Kivelä and Mason A Porter. Isomorphisms in multilayer networks. *IEEE Transactions on Network Science and Engineering*, 5(3):198–211, 2018.

[30] Mikko Kivelä, Alex Arenas, Marc Barthelemy, James P Gleeson, Yamir Moreno, and Mason A Porter. Multilayer networks. *Journal of complex networks*, 2(3):203–271, 2014.

[31] Brendan D McKay. Isomorph-free exhaustive generation. *Journal of Algorithms*, 26(2): 306–324, 1998.

[32] Brendan D McKay et al. Practical graph isomorphism. *Congressus Numerantium*, 30:45–87, 1981.

[33] Luis AA Meira, Vinícius R Máximo, Álvaro L Fazenda, and Arlindo F Da Conceiçao. Acc-motif: accelerated network motif detection. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 11(5):853–862, 2014.

[34] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594): 824–827, 2002.

[35] 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*, 303 (5663):1538–1542, 2004.

[36] Pedro Paredes and Pedro Ribeiro. Towards a faster network-centric subgraph census. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 264–271. ACM, 2013.

[37] Pedro Paredes and Pedro Ribeiro. Large scale graph representations for subgraph census. In *International Conference and School on Network Science*, pages 186–194. Springer, 2016.

[38] Nataša Pržulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, 2007.

[39] Yuanfang Ren, Aisharjya Sarkar, Ahmet Ay, Alin Dobra, and Tamer Kahveci. Finding conserved patterns in multilayer networks. In *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, pages 97–102. ACM, 2019.

[40] Pedro Ribeiro. *Efficient and scalable algorithms for network motifs discovery*. PhD thesis, University of Porto, 2011.

[41] Pedro Ribeiro and Fernando Silva. Efficient subgraph frequency estimation with g-tries. In *International Workshop on Algorithms in Bioinformatics*, pages 238–249. Springer, 2010.

[42] Pedro Ribeiro and Fernando Silva. Discovering colored network motifs. In *Complex Networks V*, pages 107–118. Springer, 2014.

[43] Pedro Ribeiro and Fernando Silva. G-tries: a data structure for storing and finding subgraphs. *Data Mining and Knowledge Discovery*, 28(2):337–377, 2014.

[44] Pedro Ribeiro, Fernando Silva, and Luís Lopes. Parallel calculation of subgraph census in biological networks. In *BIOINFORMATICS*, pages 56–65, 2010.

[45] Pedro Ribeiro, Fernando Silva, and Luis Lopes. Efficient parallel subgraph counting using g-tries. In *2010 IEEE International Conference on Cluster Computing*, pages 217–226. IEEE, 2010.

[46] Pedro Ribeiro, Fernando Silva, and Luís Lopes. Parallel discovery of network motifs. *Journal of Parallel and Distributed Computing*, 72(2):144–154, 2012.

[47] Pedro Ribeiro, Pedro Paredes, Miguel E. P. Silva, David Aparicio, and Fernando Silva. A survey on subgraph counting: Concepts, algorithms and applications to network motifs and graphlets, 2019.

[48] Ryan A Rossi, Rong Zhou, and Nesreen K Ahmed. Estimation of graphlet counts in massive networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2018.

[49] Harmanjit Singh and Richa Sharma. Role of adjacency matrix & adjacency list in graph theory. *International Journal of Computers & Technology*, 3(1), 2012.

[50] Sandra E Smith-Aguilar, Filippo Aureli, Laura Busia, Colleen Schaffner, and Gabriel Ramos-Fernández. Using multiplex networks to capture the multidimensional nature of social structure. *Primates*, 60(3):277–295, 2019.

[51] Frank W Takes, Walter A Kosters, Boyd Witte, and Eelke M Heemskerk. Multiplex network motifs as building blocks of corporate networks. *Applied Network Science*, 3(1):39, 2018.

[52] Carlos HC Teixeira, Alexandre J Fonseca, Marco Serafini, Georgos Siganos, Mohammed J Zaki, and Ashraf Aboulnaga. Arabesque: a system for distributed graph mining. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 425–440. ACM, 2015.

[53] İlker Türker and Eyüb Ekmel Sulak. A multilayer network analysis of hashtags in twitter via co-occurrence and semantic links. *International Journal of Modern Physics B*, 32(04): 1850029, 2018.

[54] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440, 1998.

[55] Sebastian Wernicke. A faster algorithm for detecting network motifs. In *International Workshop on Algorithms in Bioinformatics*, pages 165–177. Springer, 2005.

[56] Sebastian Wernicke and Florian Rasche. Fanmod: a tool for fast network motif detection. *Bioinformatics*, 22(9):1152–1153, 2006.

[57] Guangyu Wu, Martin Harrigan, and Pádraig Cunningham. Characterizing wikipedia pages using edit network motif profiles. In *Proceedings of the 3rd international workshop on Search and mining user-generated contents*, pages 45–52. ACM, 2011.

[58] Lu Zhong, Qingpeng Zhang, Dong Yang, Guanrong Chen, and Shi Yu. Analysing motifs in multilayer networks. *arXiv preprint arXiv:1903.01722*, 2019.

U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO