Network Comparison and Node Ranking in Complex Networks

David Oliveira Aparício

Programa Doutoral em Informática das Universidades do Minho, Aveiro e Porto

Departamento de Ciência de Computadores

2018

Orientador

Pedro Ribeiro, Professor Auxiliar, Faculdade de Ciências (UP)

Coorientador

Fernando Silva, Professor Catedrático, Faculdade de Ciências (UP)

Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
2018

Para os meus pais

Acknowledgments

First and foremost, I want to thank my advisors, who guided me through this journey. They were always ready to propose and discuss new ideas. One of the things I am most grateful for is the intellectual liberty they always gave me, which made me a much more autonomous researcher and person. I want to thank Professor Fernando Silva for his guidance and interest in my scientific and personal development. Through his connections I was able to spend time abroad with different research groups, which were always very enriching experiences. I truly feel indebted for all his help and trust. I also want to thank Professor Pedro Ribeiro. Before I met him I just wanted to finish my bachelor's degree and go work in Sweden or something, then he proposed that I applied for a junior research grant, and suddenly I was in academia. I've worked with him for seven years and he's the most ”natural professor” I ever had. He has an enormous passion for teaching, and he is always in a cheerful mood. It is a great privilege to be your student, collaborator, and friend.

I want to thank MAP-i's organization for providing me a good research environment. This allowed me to meet many PhD students and share our ideas, pains, and ambitions. I also want to thank MAP-i for awarding me a four-year FCT Doctoral grant [PD/BD/105801/2014].

Throughout my PhD I had the opportunity to visit UT-Austin and the Newcastle University. I want to thank them for their hospitality and for giving me the opportunity to work with such diverse groups. In particular, I thank Professor Keshav Pingali, Andrew Lenharth and Sreepathi Pai, from UT-Austin, and Professor Marcus Kaiser from Newcastle.

Although I didn't visit the University of Notre Dame, I want to give my very special thanks to Professor Tijana Milenkovic. We were having difficulties entering a new topic (network alignment) and decided to contact Tijana, with only an initial draft of a paper. We only expected some minor comments but received detailed feedback and started a collaboration. I want to thank her for all her time and guidance, and for somehow fitting me in her busy schedule.

I met many colleagues during my time in Porto. I want to thank my old time pals João Patrício and Rafael Nunes, now playing in the big leagues in Amazon and UBS, respectively. I want to thank my office mates for all the interesting discussions at lunch time, namely Miguel Araújo, Pedro Paredes, Miguel Silva, Pedro Ferreira, Christopher Harrison, Sadeeq, Daniel Loureiro, Jorge Silva and Nuno Guimarães.

I have some friends outside the department too. I want to thank Diogo Dietl and Ana Rodrigues for helping me fill my time with things beyond Computer Science, mostly movies, art, literature, and Youtube videos. While you were not directly involved in any part of this thesis, you kept me sane (enough) and your company is invaluable.

Quero agradecer às minhãs irmãs que, apesar de estarem longe, enviam apoio frequentemente. Sou um irmão mais novo muito mimado. Quero agradecer aos meus pais por tudo o que fazem por mim. Obrigado por toda a segurança e conforto que me dão. A minha mãe é a pessoa mais forte que conheço, depois de tudo o que passou mantém a energia para reger a família. O meu pai é a minha maior inspiração. O maior elogio que me podem fazer é dizerem-me que sou parecido com ele, o que, felizmente, ouço com frequência. Obrigado por tudo, sem vocês não teria sido possível concluir esta fase da minha vida.

Finally, just some random thoughts. Recently I was browsing through my high school notebooks at my parents' house and, somewhere between angry teen texts and not-so-great drawings, I found my portfolio for "Área de Projecto", basically a course where students made a year-long group project; mine was to design and build a software catalog to organize movies. The idea was to learn some programming, since me and the other three guys wanted to follow software engineering. Unsurprisingly, I got so bored out of that project that I switched to another one mid-term: build a videogame about AIDS. It didn't go anywhere. At the end of the year, I concluded in my final report that "Este projecto foi muito útil para perceber que, definitivamente, não quero seguir informática". So I decided to apply for Biology.

Ten years later, I want to thank my old self for not studying for the Biology exam, since this Computer Science thing turned out to be great.

Abstract

Networks are widely used to represent biological data, social relations, and many other systems. To make sense of complex networks, network science studies their topology. Numerous topological measures have been proposed, with subgraph-based measures receiving attention due to the rich structural information that they offer.

Here we address two major tasks in network science: network comparison and node ranking. Network comparison consists in analyzing the structural similarities between two networks. Comparing different networks allows for the transfer of knowledge from a well-known system to a less studied one. Node ranking consists in finding important nodes in a network, e.g., influential information spreaders or points of failure (i.e., if these nodes are removed from the network, the system collapses).

We divide network comparison into two tasks: network classification and network alignment. Regarding network classification, we extend subgraph-based measures, namely graphlets, to support both edge direction and temporal information. Regarding network alignment, we propose a new method to align temporal networks. Our results show that we outperform state-of-the-art both in terms of accuracy and running time.

We propose a measure of node centrality based on graphlets and the notion of *dominance*. We show that our measure gives different topological information from traditional node centrality measures and is more well-suited to discover dominant nodes. We verify this on a sports network and on a citations network. We also propose a PageRank-like method that uses, not only network topology, but also features from the citation network, such as author and venue information. Our feature enriched topology method outperforms state-of-the-art when ranking scientific authors.

We believe that our work leads to a better understanding of network structure and puts forward many research threads for the future.

Resumo

As redes são utilizadas para representar ligações biológicas, relações sociais e muitos outros sistemas. Para analisar redes complexas, tipicamente estudamos a sua topologia. Métricas topológicas baseadas em subgrafos são prevalentes, dado o grande volume de informação estrutural que contêm.

Nesta tese focamo-nos em duas tarefas de *Network Science*: comparação de redes e ranking de nós. A comparação de redes consiste em encontrar semelhanças estruturais entre duas redes. Comparar redes permite transferir conhecimento de uma rede para outra. Ranking de nós consiste em encontrar nós importantes numa rede, p.e., propagadores de informação numa rede social ou pontos de falha (i.e., nós que, se forem retirados da rede, o sistema entra em colapso).

Abordamos dois tipos de comparação de redes: classificação e alinhamento. Em termos de classificação de redes, propomos novas métricas baseadas em *graphlets* que têm em conta a direção das ligações e a sequência temporal. Em termos de alinhamento de redes, propomos um novo método para redes temporais. Os nossos resultados indicam que os nossos métodos são mais rápidos e têm maior precisão do que o estado da arte.

Propomos também uma métrica para capturar *dominância* de nós baseada em *graphlets*. Mostramos que a nossa métrica extrai informação da rede mais adequada para encontrar nós dominantes do que métricas de centralidade de nós tradicionais. Comprovamos os resultados em redes de desporto e redes de citações. Propomos ainda um método baseado em *PageRank* que usa, para além da informação topológica, atributos da rede de citações, nomeadamente informação dos autores e das conferências. Este método é mais eficaz a encontrar autores de renome do que outros métodos do estado da arte.

Acreditamos que o nosso trabalho potencia uma melhor análise e compreensão da estrutura de redes complexas, e que abre linhas de investigação para o futuro.

Contents

List of Tables

List of Figures

List of Algorithms

List of Acronyms and Symbols

GDD	**G**raphlet-**D**egree **D**istribution
GDV	**G**raphlet-**D**egree **V**ector
GDA	**G**raphlet-**D**egree **A**greement
GCD	**G**raphlet **C**orrelation **D**istance
DGDV	**D**ynamic **GDV**
SG	**S**tatic **G**raphlet
STG	**S**tatic **T**emporal **G**raphlet
DG	**D**ynamic **G**raphlet
SM	**S**tatic **M**otif
GoT	**G**raphet-**o**rbit **T**ransitions
OTA	**O**rbit-**T**ransition-**A**greement
NA	**N**etwork **A**lignment
GPNA	**G**lobal **P**airwise **NA**
ER	**E**rdős-**R**ényi graphs
SF	**S**cale-**F**ree graphs
FF	**F**orest **F**ire graphs
PPI	**P**rotein-**P**rotein-**I**nteraction networks
DC	**D**egree **C**entrality
BC	**B**etweenness **C**entrality
CC	**C**loseness **C**entrality
SC	**S**ugraph **C**entrality
PR	**P**age**R**ank
GD	**G**raphlet **D**ominance
NDCG	**N**ormalized **D**iscounted **C**umulative **G**ain
MRR	**M**ean **R**eciprocal **R**ank

$\boldsymbol{\mathcal{V}(G), \mathcal{E}(G)}$	vertices and edges of graph G
$\boldsymbol{\mathcal{O}(G)}$	orbits of graph G
$\boldsymbol{\mathcal{S}(G)}$	snapshots of temporal graph G
$\boldsymbol{Fr(H, G)}$	frequency of graph H in G
$\boldsymbol{k}$**-graph**	graph with k vertices
$\boldsymbol{\mathcal{G}_k}$	all non-isomorphic k-graphs
$\boldsymbol{\mathcal{O}_k}$	all orbits of graphs $\mathcal{G}_k$
$\boldsymbol{d\mathcal{G}_k, u\mathcal{G}_k}$	directed $\mathcal{G}_k$ and undirected $\mathcal{G}_k$
$\boldsymbol{d\mathcal{O}_k, u\mathcal{O}_k}$	directed $\mathcal{O}_k$ and undirected $\mathcal{O}_k$

Chapter 1

Introduction

Data Mining collects data in various forms best suited for different purposes [Agg15]. In networks (or graphs), data corresponds to nodes, and the relationships among the data correspond to edges. Network science studies properties of the network (i.e., the system) and properties of the nodes (i.e., the agents).

Networks are used to model all kinds of data. Social networks, for instance, represent people as nodes and their interactions as edges. These interactions can be undirected (such as friendships, i.e., two people are friends) or directed (such as follow-relationships, i.e., one user follows another, but maybe not the other way around). These interactions can be static (such as genealogy, i.e., two women are sisters) or temporal (such as marital status, i.e., two people were married, and now they are divorced). Many studies try to understand social phenomena and dynamics by analyzing social networks [WF94, LBKT08, HW09, MPK11a, CRBS12, RSF$^+$15]. However, the scope of network science is much broader than the study of social networks. For instance, in the field of computational biology different types of cellular networks are modeled as graphs with nodes representing specific biological components such as proteins or genes, and the physical, chemical or functional interactions between them modeled as edges [Prž07a, KMM$^+$10, KKK$^+$12, WLY14]. Networks are also, but not only, used to model telephone or e-mail communication, trading networks, internet routing, paper citations, air/road transportation, power transmission systems, or linguistic networks [COJT$^+$11].

Nowadays we are flooded with information. For instance, with the advent of highthroughput cell biology technologies, such as DNA microarrays [SSDB95], we increased the amount of data pertaining to molecular interactions exponentially [Mar13]. While this recent flood of information has greatly contributed to a more accurate understanding of molecule-level organization, it has also created the need to find ways to filter and model this data so that it is rendered intelligible to the practitioner. The benefits (and problems) of Big Data [KWG13, CZ14] are true for almost any other field, such as chemistry [ÁMDGL+14], finance [HBEJF14], or computer vision [MDF+12].

However, just a few decades ago, data was much scarcer. Network scientists, in particular, did not have access to large network databases which are now available [RA18]. Paul Erdős and Alfréd Rényi proposed a graph model in 1960 that assumed that each possible edge in the network was equally likely [ER60]. At the time, it was not possible to verify if their model was an adequate representation of real-world networks. By the end of the 20th century, with much more data available, Albert-László Barabási and Réka Albert showed that, in real-world networks, nodes with many connections are more likely to gain new connections than nodes with few connections (i.e., the principle of "the rich get richer") [BA99], and Duncan J. Watts and Steven Strogatz showed that, in real-world networks, nodes are never too far from each other (i.e., networks are "small-world", e.g., we all know someone, who knows someone, who knows someone, who knows President Donald Trump) [WS98].

Inspecting the network's topological features can yield valuable information about the network and, thus, of the original system. If a network has topological features that are not expected to occur in neither purely random nor purely regular graphs it is considered to be a complex network, and most real-world networks are found to be complex. As discussed above, singular characteristics commonly associated with complex networks include having a small-world structure [WS98] and/or a degree distribution that follows a power-law (scale-free networks) [BA99]. Brain networks, for instance, have been identified as small-world networks [SCKH04]. Furthermore, the distance of the average path length between nodes in the brain (representing either a brain region in mesoscale connectomes or a neuron in microscale connectomes) has been negatively correlated with a person's IQ [vdHSKP09], suggesting the importance of a small-world organization in networks' efficiency. Other statistics such as the number of connected components and the clustering coefficient are also frequently used to characterize a network. [ARS+07] provided a tool to generate a large set of such statistics.

Another approach to uncover the underlying structure of complex networks is to decompose them into their smaller components or subgraphs (see Figure 1.1). Obtaining

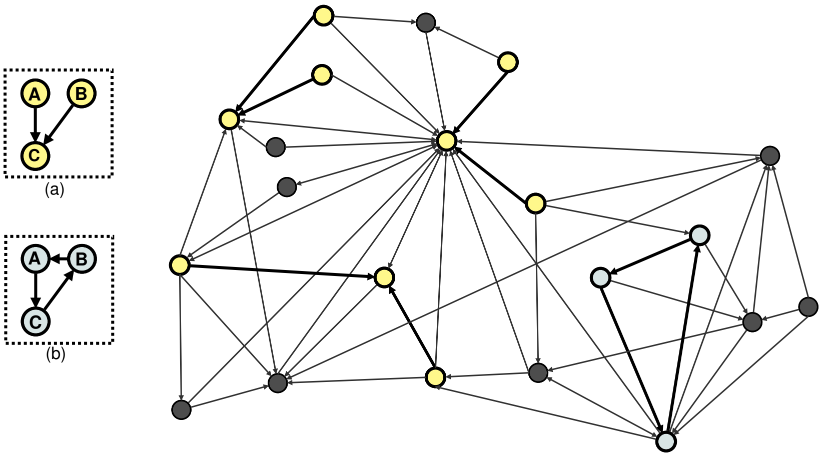


Figure 1.1: A food-web network highlighting two distinct subgraph occurrences. In the context, subgraph (a) means that ”two animals eat the same animal or plant”, while subgraph (b) means that ”*A* eats *C*, *C* eats *B*, and *B* eats *A*, what is going on?”. Subgraph (a) is much more common than subgraph (b). Thus, subgraph (a) might be regarded as a characteristic subgraph of food-web networks, i.e., other food-web networks also have higher frequency of subgraph (a) than subgraph (b). This example showcases the type of information that subgraphs offer.

the frequency of each type of subgraph offers detailed topological information which can be used to summarize and compare networks.

Frequent subgraph mining (FSM) algorithms can identify subgraphs that appear in many networks (of the same type) [JCZ13]. For instance, FSM can uncover biological pathways prevalent in many species or common fragments shared by distinct molecules [BB02].

The aim of FSM is to find subgraphs that appear frequently in an ensemble of networks; however, depending on the researcher's goal, it might be more insightful to discover subgraphs that are over-represented just on a single network. Network motifs are small over-represented subgraphs described by Milo et al. [MSOI$^+$02] as the building blocks of large complex networks. Network motifs have been used to characterize different types of networks, namely transcription networks of microorganisms, linguistic networks, or social networks [MIK$^+$04]. Network motifs have been particularly useful to analyze cellular biological networks, such as as protein-protein interaction networks [AA04], transcriptional regulatory networks [SOMMA02], metabolism networks [SBS13] or cell signaling networks [LCL$^+$13].

In conclusion, subgraphs are a powerful tool to characterize networks and nodes.

1.1 Thesis motivation

This thesis focuses on two distinct but related network problems: network comparison and node ranking. Network comparison consists in measuring topological properties of multiple networks and measuring the networks' topological similarity. Node ranking consists in measuring topological properties of multiple nodes and measuring the nodes' topological superiority.

We use similar topological properties for both tasks, namely graphlet-based measures. Each task is motivated next.

1.1.1 Network comparison

It is often useful to compare networks against each other, particularly because if a given network's properties are known it allows for knowledge transfer based on the similarity or difference between two networks [MNHP10, VKMM17].

One way to perform this comparison is to evaluate the similarity between the subgraphs that each network contains. Network motif fingerprints [MSOI^{+}02] and graphlet-based metrics [Prz07b] are possible choices for this task. Both approaches compute the frequency of a set of small non-isomorphic subgraphs (this task is called subgraph counting, or subgraph census) but, in addition to that, graphlets also evaluate the contribution of each individual node from the network, producing a graphlet degree distribution (GDD) that can be seen as an extension of the node degree concept. Furthermore, enumerating graphlets is computationally less expensive than calculating network motifs since, in the case of graphlets, the subgraphs are only counted in the original network while, in the case of motifs, the subgraphs are also counted in a large set of randomized networks in order to assess motif significance [RSK09].

Both motifs and graphlets can be effectively used to analyze any network, such as social networks [JHK12], chemical networks [BCF^{+}10], or image networks [ZHY^{+}13], since they are general concepts and provide rich topological information [YMP15]. Despite being general, both motifs and graphlets are most often used to analyze biological networks such as protein-protein interaction [KP09], disease genes [SGLP14], age-related genes [YFCM17], or brain networks [KWNP09].

Subgraph-based measures are powerful tools to summarize and compare networks. However, subgraph-based measures, and graphlet-based measures in particular, were mostly limited to undirected and static networks. This is mostly due to the time complexity of subgraph counting, which is necessary to extract subgraph-based measures, and also due to the difficulty in analyzing more complex sets of graphlets (e.g., counting graphlets that consider edge direction and time is more computationally expensive).

Despite the focus on undirected and static networks, we observe that graphlets are a general concept that can be extended to directed and temporal networks. Our work aims precisely at creating new graphlet-based measures that can be used to compare networks efficiently, both in terms of accuracy and running time.

1.1.2 Node ranking

Many applications aim to find important nodes in a network [LCR$^+$16]. One common assumption about *important nodes* is that they are crucial for the quick diffusion of the information in the network. Thus if one wants to spread information quickly (e.g., news in social networks) we only need to pass that information to a few important nodes (e.g., influencers). Similarly, if one removes just a few important nodes from a network (e.g., an essential protein in a biological system), the whole system collapses.

Based on the idea of information diffusion, *important nodes* are commonly associated with *central nodes* [LCR$^+$16]. Many node centrality measures have been proposed, such as degree centrality, betweenness centrality, or PageRank, and they have relatively high correlations between them [OFP$^+$18].

A different way to view *important nodes* is to associate them with *dominant nodes.* As far as we know, this view is not very common in network science. A *dominant node* is not necessarily a central node, since a *dominant node* should *dominate* many others (i.e., have many out-going edges, paths, or subgraphs) but be *dominated* by few (i.e., have few in-going edges, paths, or subgraphs).

The notion of *dominance* is important in social ethology, for instance, where the goal is to order a set of individuals into a dominance hierarchy [Vri98, SdV13]. In fact, dominance hierarchies are present in many species, such as the lobsters [Pet18], and there are network datasets with relations between different species[1].

Dominance networks are not restricted to social ethology, we can build dominance networks from many sources, such as sports data (e.g., good players beat other players), school hierarchy (e.g., professors advise PhD students, who themselves advise MSc

[1] `http://vlado.fmf.uni-lj.si/pub/networks/data/bio/foodweb/foodweb.htm`

students), or politics (e.g., a politician wins debates against other politicians).

We aim to develop new tools for node ranking in dominance networks that are more suitable than current node centrality measures.

1.2 Main contributions

This work consists of the design, implementation and evaluation of network comparison and node ranking methods. Our methods for network comparison target both directed and temporal networks. Most previous work targeted undirected and static networks, which are limited versions for most real world systems, where edge direction and temporal information are crucial. Our methods are consistently faster and more accurate than similar (i.e., graphlet-based measures) state-of-the-art solutions when we test them both on synthetic and real-world networks. We make our tools available for practitioners, which are of great use to computational biologists where graphlets are often used for network classification and network alignment. However, our methods are general and not limited to computational biology. We also propose new methods for node ranking that outperform state-of-the-art node centrality measures. Next, we give a more detailed description of our contributions.

Directed graphlets. We extend the concept of undirected graphlets to take into account the edge direction of the subgraphs. We hypothesize that, in the case of directed graphlets, edge direction captures relevant information than undirected graphlets ignore. We test our hypothesis on (a) a set of synthetic network pertaining to different directed graph models and on (b) real-world data corresponding to directed biological networks. We verify that directed graphlets achieve higher accuracy than undirected graphlets when classifying both types of networks.

Graphlet-tries. Related to directed graphlets, we extend the concept of g-tries to take into account the orbits of the subgraphs stored in the g-trie. Graphlet-tries are an efficient structure to store and enumerate graphlets. In our experiments, we verify that our method consisting of directed graphlets and graphlet-tries, named GT-Scanner, outperforms state-of-the-art algorithms in terms of running time.

Graphlet-orbit Transitions (GoTs). We propose GoTs, which are graphlet-based features of temporal networks. Most graphlet-based features are static or only allow for one new event per graphlet transition. Thus, GoTs captures complex subgraph transitions that other graphlet-based measures do not. Like for directed graphlets, we show that GoTs improve upon state-of-the-art both in terms of accuracy and running time when classifying synthetic and real temporal networks.

GoT-WAVE. We propose GoT-WAVE, a method for temporal network alignment (NA). We combine WAVE [SCTM15], a fast algorithm for static NA, with GoTs, the set of temporal graphlet-based features previously discussed. GoT-WAVE outperforms state-of-the-art temporal NA algorithms for most tests that we perform on synthetic and real data, in terms of accuracy and running time.

Graphlet-based node ranking. We propose graphlet dominance (GD), a measure of node ranking, based on the notion of node dominance, that takes into account the graphlets that the nodes appear at, i.e., different graphlets have different scores, and nodes that appear in high score graphlets have higher score. GD can also be regarded as a node centrality measure. We compare GD with other node centrality measures and apply it to real-world test cases: (a) player dominance in a sports network and (b) author impact in citations networks. While we focus on these two test-cases, our method is applicable to any dominance network, i.e., networks where edges indicate dominance of one node over the other.

OTARIOS. We propose a PageRank-based measure for node ranking, named OTARIOS. OTARIOS is specific to author citations networks and uses features beyond the network. This contrasts with GD, which only uses the topology of the network. OTARIOS takes into consideration multiple factors (e.g., venue prestige, year) concerning the authors' productivity (i.e., their publications) and the authors' impact (i.e., their citations). OTARIOS outperforms state-of-the-art algorithms in terms of how well its ranking matches the ground-truth ranking (i.e., rank more highly authors with more best paper awards) on several real networks, each comprised of citations in conferences on a given topic.

1.3 Thesis organization

This thesis is structured into six major chapters. A brief description of each is provided below.

1. **Introduction** presents the main context of this thesis, introducing networks, subgraph-based measures, network comparison, and node ranking. This chapter also enumerates the main contributions of this work, namely the proposed meth-

ods for network classification, network alignment, and node ranking. Finally, it presents an overview of the thesis' contents and a bibliographic note.

2. **Background** introduces necessary network terminology used throughout this thesis. It also discusses subgraph counting since it is a crucial task to obtain subgraph-based features of a network. The three main problems addressed in this thesis are also formally described, namely network classification, network alignment, and node ranking.

3. **Network classification** motivates the problem and presents related work. The chapter is divided in directed network classification and temporal network classification. On the first, we present GT-Scanner, our method that utilizes graphlet-tries to store and enumerate directed graphlets. On the second, we present GoTs, our temporal graphlet-based features. We test both on sets of synthetic and real-world networks, measuring their classification accuracy and running time against state-of-the-art approaches .

4. **Network alignment** motivates the problem and presents related work. This chapter builds upon the previous one: we use GoTs to create a new temporal NA algorithm, GoT-WAVE. We evaluate the node correctness of GoT-WAVE's produced alignments and how closely these produced alignments match its ideal alignments. We compare our results with those of state-of-the-art approaches.

5. **Node ranking** motivates the problem and presents related work. Here we present our graphlet-based node centrality measure, compare it with state-of-the-art measures, and apply it to two different real-world ranking problems: (a) player dominance in a sports network and (b) author impact in citations networks.

6. **Conclusions and future work** discusses the research done, summarizes contributions, and gives directions for future work.

1.4 Bibliographic note

Parts of the work of this thesis have already been published in international conferences, workshops, and journals. A list of those is given next:

- **Network classification.** An extension of undirected graphlets to directed graphlets. We also extend g-tries [RS14b] to graphlet-tries, incorporating the concept of orbits. We evaluate our approach on both synthetic models and real-world networks and verify that our approach, named GT-Scanner, outperforms state-of-the-art algorithms when classifying directed biological networks, both in terms of accuracy and running time. This work was published in the IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) journal [ARS17].

- **Network classification.** A new network fingerprint for temporal networks, named GoTs, as well as a measure to compare networks based on GoTs, named orbit-transition agreement (OTA). We show that GoTs outperform both static and dynamic subgraph-based measures of network similarity on synthetic and real-world networks. This work was published in the PloS One journal [ARS18].

- **Network alignment.** A new method for temporal network alignment, named GoT-WAVE. We show that GoT-WAVE outperforms existing methods for temporal network alignment on synthetic and real-world networks. This work was published in Oxford Bioinformatics [ARMS19]. An extended version is available in arXiv [ARMS18].

- **Node ranking.** A graphlet-based measure of node centrality, named GD. We apply it to a sports network. This work was published in the 7th International Workshop on Complex Networks (CompleNet 2016) [ARS16].

- **Node ranking.** A PageRank-based measure of node centrality, named OTARIOS. We apply it to a citation network. This work was published in the 7th International Conference on Complex Networks and their Applications (Complex Networks 2018) [SAS18]. An extended invited submission to the Applied Network Science journal is currently being reviewed.

Chapter 2

Background

The main goal of this work is to improve the efficiency of network comparison and node ranking algorithms. This chapter introduces the reader to network science concepts, gives an overview of subgraph counting algorithms, and formally defines the three tasks that we tackle: network classification, network alignment, and node ranking. It is the basis for understanding the remaining chapters.

2.1 Network concepts and terminology

This section introduces relevant concepts and notation from network theory, used throughout this thesis.

Network (or graph). A mathematical representation of a set of objects and their relations. A graph G is an ordered pair $G = (\mathcal{V}, \mathcal{E})$ comprising a set $\mathcal{V}$ of vertices (or nodes) together with a set $\mathcal{E}$ of their edges (or connections). Notation $\mathcal{V}(G)$ and $\mathcal{E}(G)$ is used when it is necessary to clarify that $\mathcal{V}$ and $\mathcal{E}$ are the vertex set and the edge set, respectively, of G.

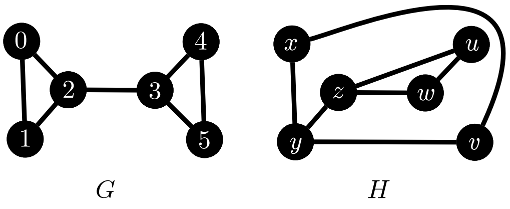
Vertices (or nodes). Set of objects in a graph, represented by $\mathcal{V}$. A vertex v can represent a person in a social network, a protein in a protein-protein-interaction (PPI) network, or an animal in a food web network. We typically use "size of a network" to mean "its number of nodes", represented by $|\mathcal{V}|$. Thus, a k-graph (or graph of size k) has $|\mathcal{V}| = k$.

Edges (or connections). Set of interactions between vertices, represented by $\mathcal{E}$. Edges can be connections in a social network, interactions between proteins in a PPI network, or eating habits in a food web network. Edges are represented by a tuple (u, v), where u and v are vertices. In directed networks, the order of the tuple is relevant, e.g., in a directed food web network (u, v), represents that "u eats v". Node u is thus the "head" of the edge and node v is the "tail". In undirected networks, the order is irrelevant, e.g., in an undirected social network, (u, v) represents that "u is a friend of v and vice-versa". The number of edges in a network is represented by $|\mathcal{E}|$.

Temporal network. Temporal networks used throughout this thesis consist of s consecutive snapshots of a network G. The set of all snapshots of G is referred to as $\mathcal{S}(G)$. An edge (u, v) exists in snapshot $S_i(G) \in \mathcal{S}(G)$ if nodes u and v are connected in the interval $[I_G + \rho \times i, I_G + \rho \times (i+1)[$, where I_G is the starting time of the network (e.g., 1st October of 2012) and ρ is the time-interval (e.g., 2 years). A temporal edge is also referred to as an *event*. Parameters ρ and s depend on the network; for instance, in scientific co-authorship networks one or two years are the more suitable value for ρ, while in conference interaction networks ρ is a few hours or a couple of days. Networks can gain (or lose) new edges (or new nodes) from $S_i(G)$ to $S_{i+1}(G)$.

Neighborhood and exclusive neighborhood. The neighborhood of vertex $v \in \mathcal{V}$, denoted as $N(v)$, is composed by the set of vertices $u \in \mathcal{V}$ such that $(v, u) \in \mathcal{E}$. Each node $u \in N(v)$ is a neighbor of v. The exclusive neighborhood of a vertex u in relation to a set of vertices $\mathcal{V}_s \subset \mathcal{V}$, denoted by $N_{exc}(v, \mathcal{V}_s)$, are the neighbors u of v that are not neighbors of any $w \in \mathcal{V}_s$. More formally, $N_{exc}(v, \mathcal{V}_s) = \{u : u \in N(v), \forall w \in \mathcal{V}_s, u \notin N(w)\}$.

Graph isomorphism. Task of detecting if two graphs are topologically equivalent. More formally, two graphs are isomorphic if there exists a bijection (or mapping) of their vertex sets, $f : \mathcal{V}(G) \to \mathcal{V}(H)$, such that $\forall(u, v) \in \mathcal{E}(G) : (f(u), f(v)) \in \mathcal{E}(H)$. Notation $u \sim u'$ represents that $u \in \mathcal{V}(G)$ was mapped to $u' \in \mathcal{V}(H)$ (i.e., $f(u) = u'$), and $(u, v) \sim (u', v')$ represents that edge $(u, v) \in \mathcal{E}(G)$ was mapped to $(u', v') \in \mathcal{E}(H)$. Figure 2.1 gives an example of two isomorphic graphs.



$0 \sim u$ $\quad$ $3 \sim y$

$1 \sim w$ $\quad$ $4 \sim x$

$2 \sim z$ $\quad$ $5 \sim v$

$f(\mathcal{V}(G) \to \mathcal{V}(H))$

$(0,1) \sim (u,w)$ $\quad$ $(3,4) \sim (y,x)$

$(0,2) \sim (u,z)$ $\quad$ $(3,5) \sim (y,v)$

$(1,2) \sim (w,z)$ $\quad$ $(4,5) \sim (x,v)$

$(2,3) \sim (z,y)$

$f(\mathcal{E}(G) \to \mathcal{E}(H))$

Figure 2.1: Two isomorphic graphs G and H and their bijection.

Subgraph. A subgraph of G is represented as S_G, where $\mathcal{V}(S_G) \subseteq \mathcal{V}(G)$ and $\mathcal{E}(S_G) \subseteq \mathcal{E}(G)$.

Subgraph isomorphism. Task of detecting if graph G contains a subgraph S_G isomorphic to graph H. H is induced by S_G if all edges $(u, v) \in \mathcal{E}(H)$, and only those, are present in its bijection to S_G, i.e., $\forall (u, v) \in \mathcal{E}(H) : (f(u), f(v)) \in \mathcal{E}(S_G)$. Figure 2.2 illustrates induced and non-induced subgraph isomorphism.

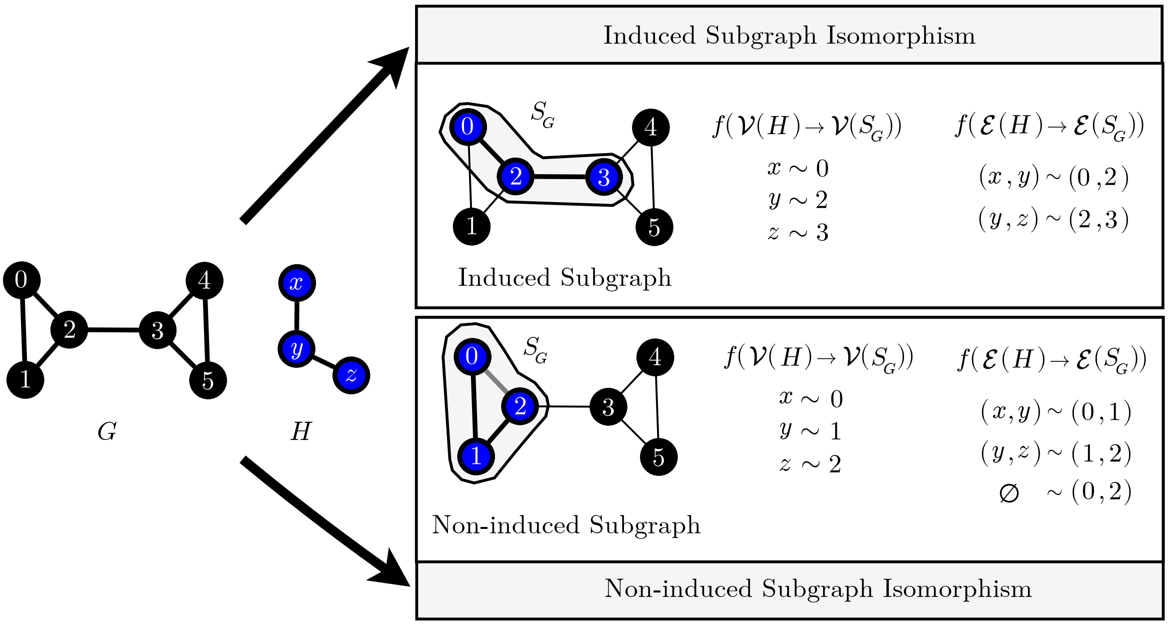

Induced Subgraph Isomorphism
S_G
0
2
3
4
1
5
$f(\mathcal{V}(H) \rightarrow \mathcal{V}(S_G))$
$x \sim 0$
$y \sim 2$
$z \sim 3$
$f(\mathcal{E}(H) \rightarrow \mathcal{E}(S_G))$
$(x, y) \sim (0, 2)$
$(y, z) \sim (2, 3)$
Induced Subgraph
0
1
2
3
4
5
G
x
y
z
H
S_G
$f(\mathcal{V}(H) \rightarrow \mathcal{V}(S_G))$
$x \sim 0$
$y \sim 1$
$z \sim 2$
$f(\mathcal{E}(H) \rightarrow \mathcal{E}(S_G))$
$(x, y) \sim (0, 1)$
$(y, z) \sim (1, 2)$
$\varnothing \sim (0, 2)$
Non-induced Subgraph
Non-induced Subgraph Isomorphism

Figure 2.2: Induced and non-induced subgraph isomorphism. The gray edge $(0, 2) \in \mathcal{E}(S_G)$ is not mapped to any $(u', v') \in \mathcal{E}(H)$ in the non-induced subgraph.

Subgraph counting. Task of counting how many subgraphs S_G of G are isomorphic to graph H. A match of H in G is referred to as an ”occurrence”. The number of occurrences of H in G is its ”frequency”, represented by $Fr(H, G)$. (more details in Section 2.2).

Graphlets. Set of small non-isomorphic graphs (i.e., every graphlet in the set is not isomorphic to any other). We use $\mathcal{G}_k$ to represent all possible k-graphlets. $d\mathcal{G}_k$ and $u\mathcal{G}_k$ denote directed and undirected k-graphlets, respectively. For simplicity, when terms are applicable to both directed and undirected graphlets, the more general notation $\mathcal{G}_k$ is used.

Automorphism and Orbit. An isomorphism of a graph into itself is an automorphism, and the set of all possible automorphisms of G is denoted by $\mathcal{A}(G)$. Figure 2.3 illustrates all four automorphisms of G. Two vertices u and v are equivalent when there exists some automorphism that maps u into v (i.e., they are in the same orbit).

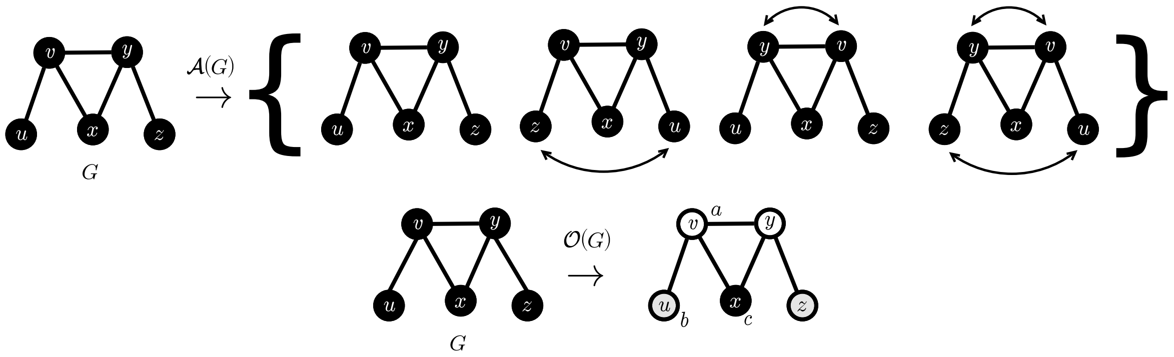


Figure 2.3: On the top figure: set of automorphisms of graph G. Arrows indicate nodes that, when exchanged, the resulting graph is isomorphic to G. On the bottom figure: set of orbits of graph G. Node colors and letters identify orbits.

For instance, u into z are in the same orbit since at least one automorphism maps one into the other. G has three different orbits in total (a, b and c), constituting $\mathcal{O}(G)$. More informally, orbits are the unique positions that a node can occupy in a graph.

Node-degree and Degree distribution. In undirected graphs, the node-degree of u is the number of edges that node u participates in (i.e., the size of its neighborhood). The degree distribution of a network is the number of nodes in the network with degree d, with $d \in [1, +\infty[$. In practice, real networks do not have nodes with infinite connections. Thus, the degree distribution is a vector with size $max(d)$. In directed graphs, the node-outdegree is the number of edges where u is the head, and the node-indegree is the number of edges where u is the tail. The in-degree (out-degree) distribution of a network is the number of nodes in the network with in-degree (out-degree) d, with $d \in \{1, +\infty\}$.

Graphlet-degree vector and Graphlet-degree distribution. Graphlet-degree vector (GDV) is a generalization of the node-degree. Consider graph G and orbit a from Figure 2.4. The node-degree of $v \in \mathcal{V}(G)$ is the number of times v appears in a subgraph of G isomorphic to A in orbit a (which is the only possible orbit in A). GDVs consider not only the node-degree (orbit a) but also orbits from bigger graphlets (Figure 2.4). The GDV of a single node $v \in \mathcal{V}(G)$, represented by $GDV(v)$, is obtained by:

1. Performing subgraph counting of each individual graphlet on network G (e.g., obtaining all occurrences of graphlets A, B, and C on G).

2. For each occurrence that contains v, evaluate in which orbit o node v is and increment $GDV(v)[o]$.

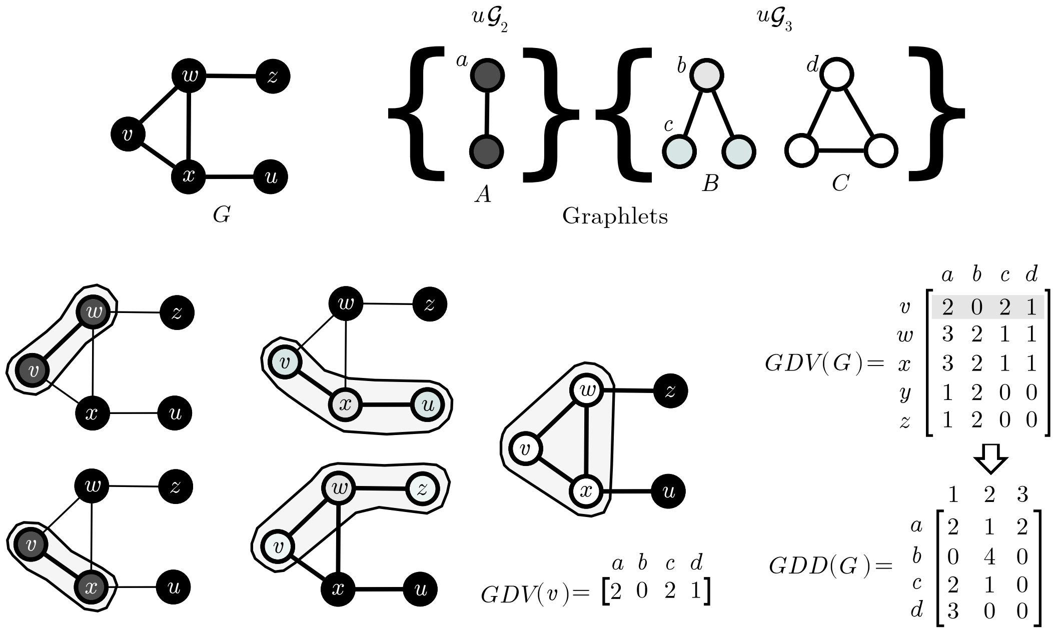


Figure 2.4: Graphlet-degree vectors (GDV) and graphlet-degree distribution (GDD).

For instance, consider orbits a, b, c and d from Figure 2.4. Node v has degree equal to 2 (orbit $a \in A$) and appears in 1 triangle (orbit $d \in C$). Graphlet B has two possible orbits, and v appears at the chain-periphery (orbit $c \in B$) 2 times, and never at the chain-center (orbit $b \in B$). The GDV of all nodes from G is represented by $GDV(G)$. Each row of the $GDV(G)$ matrix can be used as the node's features, and is useful for node comparison. A graphlet-degree distribution (GDD) of orbit o, is the number of nodes in the network that appear d times in orbit o. Notice from Figure 2.4 that $GDD(a)$ is the same as the degree-distribution. The GDD of all orbits is represented by $GDD(G)$. Typically, all possible k-graphlets of a given size k are used to obtain $GDD(G)$, e.g., all undirected graphlets with at most five nodes (Figure 2.5).

Network comparison. General task of comparing two networks according to their topological similarities. Here we subdivide network comparison into two concrete tasks: network classification (Section 2.3) and network alignment (Section 2.4).

Node ranking. General task of obtaining a rank (i.e., score) for each node in a given network. In Section 2.5 we describe this task formally.

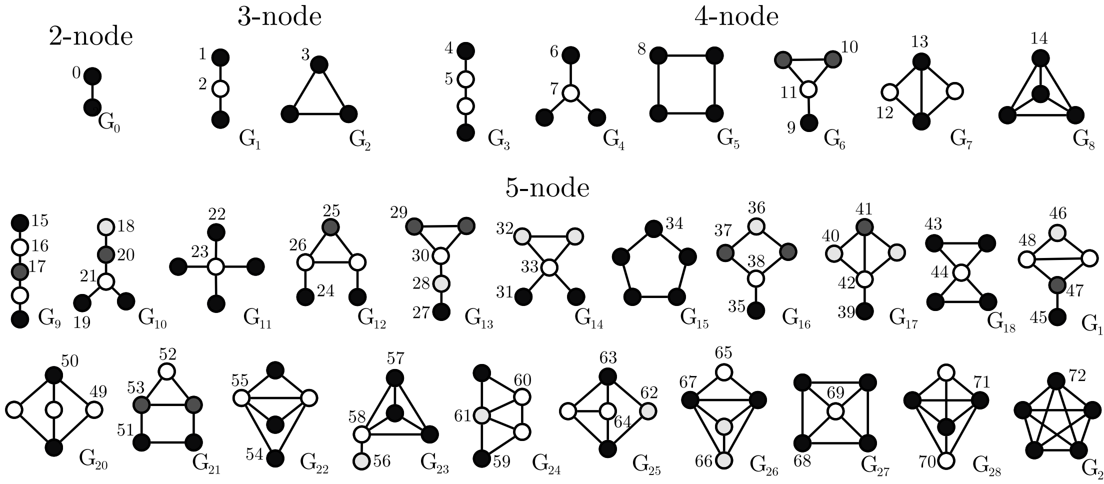


Figure 2.5: All undirected graphlets of up to 5 nodes and respective orbits [adapted from [Prž07a]].

2.2 Subgraph counting

In this section we describe the task of subgraph counting and give an overview of subgraph counting methods. Subgraph counting is required to obtain graphlet-based measures, which we use for both network comparison and node ranking.

Input. A set $\mathcal{G}_k$ of non-isomorphic k-subgraphs and a graph G.

Problem statement. Determine the frequency $Fr_G(G_i)$ of all induced occurrences of the subgraphs $G_i \in \mathcal{G}_k$ in G. Two occurrences are considered different if they have at least one node or edge that they do not share. Other nodes and edges can overlap.

Output. The frequency $Fr_G(G_i)$ of each $G_i \in \mathcal{G}_k$ in graph G.

Subgraph counting, as defined above, is sometimes also referred to as subgraph census [WF94, RS14b], and we use the two terms interchangeably. An important distinction exists between subgraph counting and subgraph enumeration: subgraph enumeration requires the actual subgraph occurrences to be listed while subgraph counting only requires the subgraph frequencies. Since the input is a single graph, subgraph counting differs from Frequent Subgraph Mining (FSM) where the input is, typically, a collection of graphs [JCZ13] (FSM is discussed in Section 2.2.5.2).

In this work we typically count large and general sets of small graphs. However, approaches that target specific subgraphs can also be valuable depending on the context, such as algorithms that count triads [SW05], cliques [FFF15] or stars [GRS11]. Networks used throughout this thesis have no edge colors nor node colors[1], but subgraph counting algorithms targeting them do exist [GS13, GRSJRMK13, RS14a, KS18].

Subgraph counting approaches are typically divided into three categories [RSK09, MNSK12, RS14b]. On one end of the spectrum, network-centric approaches [MSOI+02, WR06a, KAE+09] first extract all k-node subgraph occurrences in G and only after evaluate which graph $G_i \in \mathcal{G}_k$ is isomorphic to each subgraph occurrence found. On the other end of the spectrum, subgraph-centric methods first pick a $G_i \in \mathcal{G}_k$ and then only count subgraph occurrences in G isomorphic to G_i. Therefore, subgraph-centric methods [OSMN09, GK07] are preferable to network-centric algorithms when $\mathcal{G}_k$ consists of just one or a few graphs. Set-centric approaches [PR13, KSD+13, RS14b] are in the middle of the spectrum: they take as input a set of interesting graphs and only count those in G, thus, they count not just one but not necessarily all.

We focus on the network-centric approach because it is intrinsically the most general approach since all subgraph information is collected and, opposing approaches, require hand-picking an (ensemble of) interesting subgraph(s), which might be hard and heavily dependent on our knowledge of the network.

Despite its usefulness, subgraph counting is often limited to small networks and small subgraphs due to its computational complexity. Determining if just one graph is contained in another (i.e., subgraph isomorphism [Ull76]) is a NP-complete problem. Furthermore, millions or billions of subgraph occurrences are found even in relatively small networks, and the number of occurrences increases exponentially with k [Rib11].

Algorithms that perform a full subgraph enumeration (Section 2.2.1) rely on techniques to efficiently traverse the search space, avoid counting the same instance more than once, and enumerate multiple subgraphs at the same time, which is possible due to topological similarities between different subgraphs. Other algorithms avoid doing a full k-graph enumeration and use analytic methods to speed up computation (Section 2.2.2). However, these analytic methods have limitations in their scope, i.e., usually they are applicable only to small undirected subgraphs. Exploiting parallelism has also been put forward by many researchers as a way to reduce computational time (Section 2.2.3). However, subgraph counting induces unbalanced search trees, thus requiring care in how work is divided and shared among parallel workers. Counting only approximate subgraph frequencies is another possibility that greatly reduces computational time (Section 2.2.4), with the additional challenge of managing a trade-off between speed and accuracy.

[1] This type of networks are also commonly referred to as multilayer networks [KAB+14].

Next, we discuss some of these strategies and give an overview of subgraph counting methods. We give a particular focus to g-tries [RS14a] since we use them for subgraph counting in later sections. Finally, we discuss related problems to subgraph counting.

2.2.1 Enumeration approaches

In the seminal work in network motif analysis [MSOI$^+$02], Milo et al. proposed a recursive backtracking algorithm to count subgraphs. The algorithm, named mfinder, begins by picking a single edge from the network. Then, it expands the edge to a 3-node connected subgraph by choosing a neighboring edge, and so forth until obtaining a k-node connected subgraph. mfinder repeats this process until the neighborhood of the initial edge is fully explored. Then, a new initial edge is chosen and the same process is followed. mfinder stops when every edge has been explored as an initial 2-node subgraph. mfinder produces many repeated occurrences since different initial edges reach the same k-node subgraph(s). For instance, in a network with just three fully connected nodes (i.e., a clique) a, b and c, mfinder finds six subgraph occurrences ($\{a, b, c\}$, $\{a, c, b\}$, $\{b, a, c\}$, $\{b, c, a\}$, $\{c, a, b\}$ and $\{c, b, a\}$). However, they are not different occurrences – they all represent the same isomorphic graph. To avoid counting the same subgraph occurrence multiple times, mfinder uses a graph isomorphism tool (such as nauty [MP14]) to obtain a canonical representation of the subgraph occurrences (e.g., all six occurrences mentioned have canonical class $\{a, b, c\}$). In practice, mfinder stores the first occurrence of the canonical class in an hashtable and, when a repeated occurrence is found, it does not increase its frequency. Thus, while the subgraph frequencies are correct, mfinder performs redundant computation.

ESU [Wer06] greatly improved upon mfinder by never finding the same subgraph occurrence twice, thus saving time and also space since it does not need mfinder's hashtable. ESU applies a recursive method to each vertex v of the input graph G: it uses two sets $\mathcal{V}_S$ (the subgraph occurrence) and $\mathcal{V}_E$ (the possible extensions), which initially are set as $\mathcal{V}_S = \{v\}$ and $\mathcal{V}_E = N(v)$. Then, for each vertex u in $\mathcal{V}_E$, it removes it from $\mathcal{V}_E$ and makes $\mathcal{V}_S = \mathcal{V}_S \cup \{u\}$, effectively adding it to the subgraph being enumerated and $\mathcal{V}_E = \mathcal{V}_E \cup \{u \in N_{exc}(u, \mathcal{V}_S) : u > v\}$ (where v is the original vertex to be added to $\mathcal{V}_S$). The N_{exc} here makes sure that ESU only grows the list of possibilities with vertices not already in $\mathcal{V}_S$ and the condition $u > v$ is used to break symmetries, consequently preventing any subgraph from being found twice. This process is repeated until $\mathcal{V}_S$ has k elements, which means a k-subgraph was found. Like mfinder, at the end of the process ESU performs isomorphism tests (using nauty) to compute the canonical class of each subgraph occurrence, which is a considerable bottleneck.

2.2.1.1 G-Tries

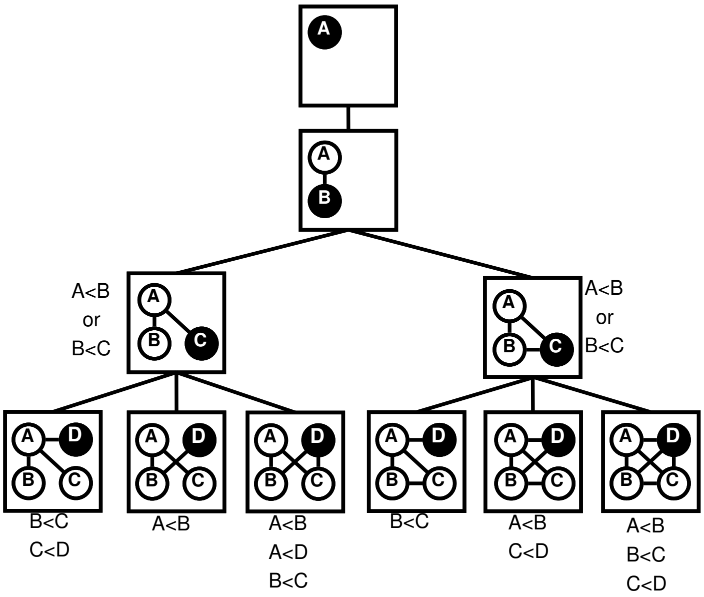


Figure 2.6: Example of a g-trie containing all 4-node undirected subgraphs.

A g-trie (Figure 2.6) is a data-structure to store and enumerate subgraphs [RS10b, RS14c]. Its efficiency is mostly due to two main algorithmic ideas.

First, the search space is heavily constrained by identifying common subtopologies between the set of subgraphs $\mathcal{G}_k$ before enumerating them. Figure 2.7 illustrates this base concept by showing three small graphs that share a common subtopology. In practice this means that instead of enumerating each subgraph, G_{S1}, G_{S2} and G_{S3}, individually, a g-trie starts by looking for occurrences of the smaller common subgraph G_{S0} and then performs the necessary expansions for each larger subgraph. This also means that g-tries already know the isomorphic class of the subgraph occurrence during enumeration, removing the need to use nauty (or similar tools) to assess the canonical class of the subgraph occurrence after it has been found.

Second, symmetry breaking conditions are automatically generated to eliminate automorphisms, thus avoiding redundancies and guaranteeing that each occurrence is found only once. Consider Figure 2.6: a 4-node clique $\{A = 1, B = 2, C = 3, D = 4\}$ is valid but a 4-node clique $\{A = 2, B = 1, C = 3, D = 4\}$ is not valid because $(A < B) \rightarrow (2 < 1)$, which is false. Thus, the second example is not found. Now

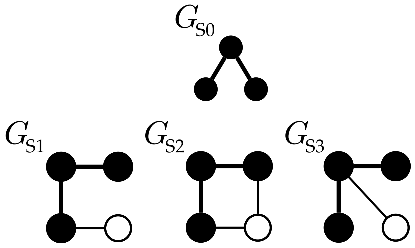


Figure 2.7: Common topology of three graphs. G-Tries use common structures between the graphs of $\mathcal{G}_s$ to heavily constrain the search space.

consider $\{A = 2, B = 3, C = 1, D = 4\}$: the g-trie does not even reach a 4-node clique because its parent node has conditions $A < B \vee B < C$, which is false since $2 < 3 \vee 3 < 1$ is false. Symmetry breaking at intermediate stages greatly reduce computation time. G-Tries were shown to be one or two orders of magnitude faster than ESU [Rib11].

A g-trie receives as input the list of graphs that the user wants to enumerate, which can be all undirected graphlets with up to five nodes, a set of directed graphs, specific interesting patterns (such as cliques or stars), or any other desired graphs. However, due to the nature of the g-trie, which relies on common subtopologies between graphs to construct a compact search tree, g-tries are better suited for tasks where one wants to count the occurrences of many small graphs inside a large network. Since we want to enumerate all k-subgraphs, g-tries are an appropriate choice. G-Tries can be fully constructed before enumeration and stored in files, one for each set of subgraphs to enumerate (e.g., build a g-trie containing $u\mathcal{G}_5$ and use it to compute the frequencies of $u\mathcal{G}_5$ on several networks.)

Other tree-based approaches have been proposed. FASE builds a graph-tree structure on-the-fly, thus avoiding pre-building a full g-trie [PR13]. Quatexelero builds a different type of graph-tree (i.e., a quad-tree) which is more specialized in directed graphs [KSD$^+$13]. Both Quatexelero and FaSE have potential memory issues, since there may be several tree-nodes representing the same graph, which is not a problem for g-tries since it only stores one copy of each possible graph. Thus, g-tries are still the data-structure that results in the fastest algorithm for general subgraph counting, i.e., subgraph counting of any graph set, without targeting a specific set.

2.2.2 Analytic approaches

The previous section focused on general approaches. In some cases, when one chooses a set of subgraphs of interest (e.g., 4-node or 5-node undirected subgraphs are a common choice [Prž07a, MP08, JHK12, MDP14]), it might be faster to use a specialized tool.

Some approaches [HD14, OB16] relate the frequency of each subgraph with the frequencies of smaller subgraphs, which are thus less computationally expensive to count. They construct a matrix of linear equations between subgraphs frequencies that can be solved using traditional linear algebra methods. Other approaches [MS12, MMFDC14, ANR$^+$17, PSV17] decompose subgraphs into several smaller patterns of graph properties, like common neighbors, or triangles that touch two vertices.

ORCA [HD14] is an example of the first kind of approaches. ORCA begins by building a system of linear equations relating the orbit frequencies of 5-node undirected orbits with 4-node undirected orbits. Then, only the 4-node undirected orbits are enumerated, and the linear equations are solved to obtain the frequencies of the 5-node undirected orbits. In practice, due to the way ORCA's equations are built, the system also requires the enumeration of a single 5-node undirected orbit. Usually, the orbit pertaining to the clique is chosen, since there are efficient algorithms to count this orbit [JS17, ERS18] and, for sparse enough networks, it is usually the one with fewest occurrences, making it less expensive to count. ORCA was shown to be one order of magnitude faster than enumeration based approaches [HD14]. Addressing size limitation, [HD17] suggested a way of producing efficient equations for arbitrary sized undirected subgraphs. However, as far as we know, an extension for directed graphlets has not been proposed yet.

ESCAPE [PSV17] is an example of the second kind of approaches. ESCAPE is based on a divide and conquer approach that identifies substructures of each counting subgraph to partition them into smaller patterns (e.g., triangles, paths). It is a very general method, but with the correct choices for decomposition, it is possible to describe a set of formulas to compute the frequency of each subgraph. The original paper only describes the resulting formulas up to 5-node undirected subgraphs. For most test cases, ESCAPE is one order of magnitude faster than state-of-the-art, namely ORCA. As far as we know, ESCAPE is the most efficient algorithm to count undirected subgraphs and orbits up to size 5.

Another intrinsic limitation of these approaches is that, since they do not enumerate all subgraph occurrences, they can not be used when one needs not only subgraph counts but also subgraph listings. For instance, one might want to analyze interesting subgraph occurrences, e.g., communities, blacklisted patterns, nodes with unusual GDV, etc.

2.2.3 Parallel approaches

The availability of parallel environments, such as multicores, hybrid clusters, and GPUs gave rise to strategies that leverage on these resources. One key aspect necessary to achieve a scalable parallel computation is finding a balanced work division (i.e., splitting work-units *evenly* between workers, e.g., parallel processors, threads). A naive possibility for subgraph counting is to assign $\frac{|\mathcal{V}|}{|P|}$ nodes from network G to each worker $p \in P$. This egalitarian division is a poor choice since two nodes induce very different search spaces; for instance, *hub*-like nodes induce many more subgraph occurrences than nearly-isolated nodes.

Instead of performing an egalitarian division, [WTZ^{+}05] discriminate nodes by their degree and distribute them among workers, the idea being that each worker gets roughly the same amount of *hard* and *easy* work-units. Despite achieving a more balanced division than the naive version, there is no guarantee that the node-degree is sufficient to determine the actual complexity of the work-unit. Distributing work immediately (without runtime adjustments) is called a static division. Wang et al. did not assess scalability in [WTZ^{+}05], but they showed that their parallel algorithm was faster than mfinder [MSOI^{+}02] in an E. Coli transcription regulation network.

MPRF [LJC^{+}09] is implemented following a MapReduce model [DG08]. In MPRF, mappers extend size k occurrences to size $k+1$ and reducers remove repeated occurrences. At each level, MPRF divides work-units evenly among workers. We still consider this to be a static division since no adjustments are made in runtime. Overhead caused by reading and writing to files reduces MRPF's efficiency, but the authors report speedups of $\approx$ 7x on a 48-node cluster, when compared to the execution on a single-processor.

In both cases previously discussed, a worker has to finish a work-unit before proceeding a new one. Therefore, it is possible that a worker gets stuck processing a very computationally heavy work-unit while other workers are idle.

[RSL10] was the first to implement work sharing during parallel subgraph counting. Workers have a splitting threshold that dictates how likely it is to, instead of fully processing a work-unit, putting part of it in a global work queue. A work-unit is divided using diagonal work splitting which gathers unprocessed nodes at level k (i.e., nodes that are reached by expanding the current work-unit) and recursively goes up in the search tree, also gathering unprocessed nodes of level $k - i$, $i < k$, until reaching level 1. This process results in a set of finer-grained work-units that induces a more balanced search space than static and first-fit divisions. [RSL10] uses ESU as their core enumeration algorithm and propose a master-worker architecture where a master-node manages a work-queue and distributes its work-units among slave workers. This strategy was the first to achieve near-linear speedups (≈128x on a 128-node cluster) on a set of heterogeneous networks. A subsequent version [RS10a] used g-tries as their base algorithm and implemented a work stealing architecture. The approach by [RS10a] was more efficient since it uses a faster enumeration algorithm (g-tries vs ESU) and has all workers perform subgraph enumeration (without wasting a node in work queue management). A similar implementation (based on W-W sharing and diagonal splitting) of g-tries were also developed for shared memory (SM) environments, which achieved near-linear speedups in a 64-core machine [ARdS14]. The main advantages of SM implementations is that work sharing is faster (since no message passing is necessary) and SM architectures (such as multicores) are a commodity while distributed memory (DM) architectures (such as a cluster) are not.

Other approaches use graphics processing units (GPUs). GPUs are processors specialized in image generation, but numerous general purpose tasks have been adapted to them [FLL$^+$08, HOO11, MGG12]. GPUs are appealing due to their large number of cores, reaching hundreds or thousands of parallel threads whereas commodity multicores typically have no more than a dozen. However, algorithms that rely on graph traversal are not best suited for the GPU framework due to branching code, non-coalesced memory accesses and coarse work-unit granularity [MGG12].

[LXXL17] put forward a GPU algorithm mostly targeted at network motif discovery (thus, counting subgraphs in many networks in parallel) but also with some emphasis on efficient subgraph enumeration. [LXXL17] avoids duplicate in a similar fashion to ESU [Wer06] and auxiliary arrays are used to mitigate uncoalesced memory accesses. A BFS-style traversal is used (extending each subgraph 1 node at a time) to better balance work-units among threads. [LXXL17] their GPU-algorithm running on a 2496-core GPU (Tesla K20) against parallel CPU algorithms and report a speedup of ≈10x to a 6-core execution of the fastest CPU algorithm [RS10a].

[RZ16] proposed a subgraph counting algorithm that combines multiple GPUs and CPUs. Their method dynamically distributes work between CPUs and GPUs, where few (but complex) work-units are given to the CPU whereas many (but simple) work-units are given to the GPUs. When compared to a sequential version, their hybrid CPU-GPU version achieves speedups of $\approx$ 20x to $\approx$ 200x, depending largely on the network. However, their approach is limited to 4-node subgraphs, while g-trie based parallel implementations [RS10a, ARdS14] are general approaches.

2.2.4 Sampling approaches

In some cases, extracting exact subgraph counts is too computationally expensive and/or unnecessary, and just an approximation of the frequencies is enough [KIMA04, RS10a]. For instance, one might want to estimate the magnitude of certain subgraphs (e.g., the network contains billions of 3-node cliques, or just millions).

Some approaches estimate subgraph concentrations [BRAH12, WLR$^+$14] while others estimate subgraph frequencies [Wer05, RS10a, WZZ$^+$18]. Estimating subgraph concentrations is computationally easier since it only requires an estimation of the different proportions of each subgraph in the network (e.g., 30% 3-node undirected cliques and 70% 3-node undirected chains), while estimating subgraph frequencies requires an estimation of the magnitude (e.g., consider the same concentrations as before: it is different to estimate $3 \cdot 10^3$ cliques and $7 \cdot 10^3$ chains than estimating $3 \cdot 10^6$ cliques and $7 \cdot 10^6$ chains).

Regarding subgraph concentration estimation, ESA (Edge SAmpling) [KIMA04] is a random walk method that picks a random seed edge from the network and grows that 2-node subgraph up to a k-subgraph. The process is repeated with different seed edges until the desired sample size is computed. It is a subgraph concentration estimation because, at the end of computation, ESA does not know what area of the network it has actually explored (the same subgraphs might even be found multiple times), it only knows how frequently each subgraph appeared. Furthermore, ESA is a biased estimator since edges in denser areas of the network are more likely to be sampled in a k-subgraph (Figure 2.8). Instead of sampling edges, GUISE [BRAH12] samples a seed graphlet and then counts (a portion of) its neighboring graphlets. GUISE obtains neighboring k-node graphlets from the original k-node graphlet by removing a node from the seed one and, if the other $k-1$ nodes still form a connected graph, a node from their neighborhood is picked (i.e., the new graphlet is also a connected k-graphlet). This process is repeated for different seed graphlets until a predefined number of samples is taken from the graph. The idea is that, instead of counting all subgraph occurrences from the network, GUISE only looks at a few local subgraph occurrences and assumes that the global subgraph concentration is similar to that of its sample. They address the bias problem by introducing an acceptance probability that rejects denser neighboring graphlets with higher probability than sparser ones. This strategy has the drawback of producing computation that is then ignored since some subgraphs are found but not counted, making the algorithm slower. [WLR+14] proposes a similar strategy to GUISE where no samples are disregarded.

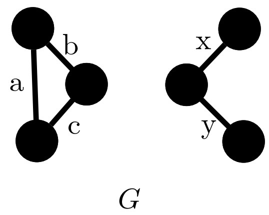


Figure 2.8: Example of a biased subgraph counting estimator. In this toy example, each edge has 20% probability of being picked as a seed edge by ESA. However, subgraph $\{a, b, c\}$ has 60% probability of being sampled by ESA while $\{x, y\}$ has 40%. An unbiased estimator should give them equal probability of being sampled, since their exact frequency is the same, otherwise it is overestimating denser subgraphs.

Regarding subgraph frequency estimation, [Wer05] proposed an approximate version of ESU (described in Section 2.2.1). Recall that ESU maintains two sets $\mathcal{V}_S$ (the set of vertices in the subgraph) and $\mathcal{V}_E$ (the set of candidate vertices to extend the subgraph). A vertex is added from $\mathcal{V}_E$ to $\mathcal{V}_S$ with probability $p(|\mathcal{V}_S|)$, where $|\mathcal{V}_S|$ is the size of the subgraph already found. In the exact solution $p(|\mathcal{V}_S|)$ is always 100%, while in the sampling solution $p(|\mathcal{V}_S|)$ depends on $|\mathcal{V}_S|$. Typically, $p(|\mathcal{V}_S|)$ is lower for larger $|\mathcal{V}_S|$, reducing computational time since larger graphlets are harder to count [Wer05, RS10a]. [Wer05], and similar strategies, give an estimation not only of subgraph concentration but also of subgraph frequency, because they know (approximately) how much of the network was explored. [RS10a] proposed a similar but more efficient strategy since it uses g-tries to store and enumerate the subgraphs. Path sampling approaches [JSP15, SPK13, WZZ+18] follow a different strategy. Paths are subgraphs composed of 2 exterior nodes and $k - 2$ interior nodes arranged in a single line. Examples of these are the subgraphs G_1, G_3 and G_9 from Figure 2.5. Instead of counting all subgraphs, these algorithms relate the number of non-induced occurrences between k-node subgraphs. For example, when $k = 4$, there are 4 non-induced occurrences of G_3 in G_5 or 12 non-induced occurrences of G_3 in G_8. Table 2.1 shows this full relationship when $k = 4$. Then, path sampling approaches enumerate only the path subgraphs and estimate the others using their non-induced occurrences relations. To the best of our knowledge, MOSS-5 [WZZ+18] is the algorithm that achieves the best trade-off between accuracy and time to estimate the frequency of 5-node subgraphs, as it is able to reach errors of $\approx 10^{-2}$ with a few samples, even for

	g_3	g_4	g_5	g_6	g_7	g_8
g_3	0	1	2	4	6	12
g_4	1	0	1	0	2	4
g_5	0	0	0	1	1	3
g_6	0	0	1	0	4	12
g_7	0	0	0	0	1	6
g_8	0	0	0	0	0	1

Table 2.1: Number of non-induced occurrences of each undirected graph of size 4 in each other. Position (i, j) in the table indicates the number of times that graph i occurs non-induced in graph j.

big networks. However the ideas behind MOSS-5 are not easily extendable to directed subgraphs and larger sized undirected subgraphs due to the ever increasing number of dependencies between the number of non-induced occurrences, making it harder to use the information contained in a table similar to Table 2.1 for these cases.

2.2.5 Related problems

2.2.5.1 Network motifs

Finding recurrent patterns in a large network can give us insights into how the real system works, and network motifs (or simply motifs) [MSOI$^+$02] are an example of recurrent structural patterns. Motif discovery counts a set of small non-isomorphic subgraphs $\mathcal{G}$ (typically all k-node subgraphs) in a large network G. This is exactly the problem of subgraph counting. However, motif analysis also assesses which subgraphs $H \in \mathcal{G}$ are over-represented in G, i.e., which subgraphs are network motifs of G.

Motif analysis thus consists of two steps: (i) subgraph counting is performed on the original graph G, and (ii) motif significance is assessed on a null model [MSOI$^+$02]. Numerous null models can be used, such as the one by [MSOI$^+$02] which generates a set $\mathcal{R}(G)$ of randomized networks that keep G's degree sequence. Subgraph counting is then performed on each $R \in \mathcal{R}(G)$. Typically, $|\mathcal{R}(G)| \approx 100$ to guarantee statistical significance, and increasing the number of randomized networks greatly increase necessary computational time The average frequency of a subgraph $H \in \mathcal{G}$ on the randomized networks is represented by $<Fr(H, \mathcal{R}(G))>$, and H is considered a network motif if it appears with a significantly higher frequency in G than in $\mathcal{R}(G)$. Motif scores, represented by $\delta_{(H,G)}$, are computed for each subgraph H (Eq 2.1). As

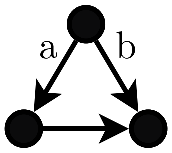


Figure 2.9: Example of a feed-forward-loop (FFL) motif. FFLs speed up the response time of the target gene expression following stimulus steps in one direction (a) but not in the other (b) [MA03]. Network motif analysis identified this subgraph as a motif in biological networks, namely gene regulation networks [MSOI+02].

was proposed in [MIK+04], motif scores are normalized, represented by $\Delta_{(H,G)}$ (Eq 2.2).

$$\delta_{(H,G)} = \frac{Fr(H,G)\ - <Fr(H,\mathcal{R}(G))>}{Fr(H,G)\ + <Fr(H,\mathcal{R}(G))>} \tag{2.1}$$

$$\Delta_{(H,G)} = \frac{\delta_{(H,G)}}{\sqrt{\sum(\delta_{(H,G)})^2}} \tag{2.2}$$

Graph H is considered a motif of G if both $Fr(H,G)$ and $\Delta_{(H,G)}$ are above certain thresholds [Rib11], guaranteeing minimum frequency and over-representation. The motif-fingerprint of G is a vector containing all $\Delta_{(H,G)}$.

On the application side, network motif analysis has identified the feed-forward-loop (FFL) (Figure 2.9) as an intrinsic part of the gene regulation of *C. elegans* and *S. cerevisiae* [MSOI+02], suggesting a fundamental similarity in the design of these biological networks. Network motifs have also been used as a network fingerprint to compare networks of different families [MIK+04], distinguishing social, linguistic, and biological networks. Similar studies have been carried out to classify metabolic networks [ZQ05], co-authorship networks [CRBS12] or articles [WHC12].

2.2.5.2 Frequent Subgraph Mining

Frequent Subgraph Mining (FSM) has similarities with network motif analysis, but it also has differences. Like network motif analysis, FSM's aim is to discover recurrent subgraph patterns but, unlike network motif analysis, FSM's input is an ensemble of networks instead of a single network[2]. Furthermore, FSM only checks for the presence

[2] Some studies separate single-graph FSM from ensemble FSM [JCZ13]. Single-graph FSM consists of subgraph counting on the network and checking which subgraphs have frequency above a support threshold. This can be seen as a simplification of network motif analysis since no null model is used. We use FSM to refer to ensemble FSM, which is the standard definition in the literature.

or absence of a subgraph in each network of the ensemble, instead of computing how many times the subgraph appears in each[3]. Then, a subgraph is considered frequent if it is found in many networks of the ensemble (i.e., above a certain support threshold).

This major algorithmic difference, i.e., only checking for presence instead of counts, allows FSM algorithms to use pruning strategies that, in other frequent subgraph analysis tasks, one can not use. By the downward closure property (DCP), if a graph is frequent in an ensemble of networks (i.e., it is present in many networks of the ensemble), then the subgraphs of that graph are also certainly frequent and checking if they are frequent is not needed, thus saving computational time. However, the DCP is not valid when checking if a graph is frequent in a single network (i.e., it appears many times in the network), as exemplified in Figure 2.10. Thus, subgraph counting algorithms [MSOI+02, Wer06, RS10a] are not very efficient in a FSM setting, and FSM algorithms [YH02, BB02, HWP03, NK05, JYW10] are not usable in a subgraph counting setting.

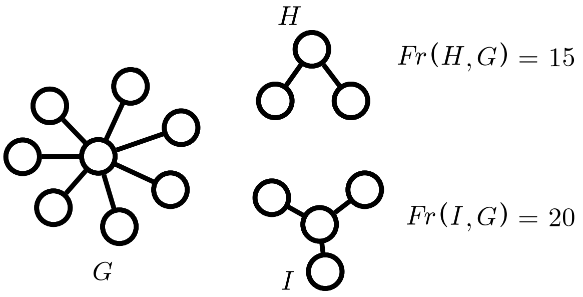


Figure 2.10: Example of why the DCP is not valid in subgraph counting. While H is a subgraph of I, I has higher frequency in G than I. Thus, it is not guaranteed that the subgraphs of a frequent graph are also frequent.

Like graphlets and network motifs, FSM is a general concept and it has been applied to biological, e-mail, chemistry, and linguist networks [JCZ13].

In this thesis, we are mainly concerned with actual subgraph counts and not just subgraph presence. One can compare networks by dividing them according to the subgraphs that appear in them, but we feel that subgraph counting is more general and offers more detailed topological information of the network.

[3] Some approaches perform subgraph counting on each network of the ensemble, but this is not the most typical case of FSM [JCZ13].

2.2.6 Subgraph counting on temporal networks

In many cases, it might be useful to analyze not only the structure of the system at a given time but also how said structure is evolving. Temporal network analysis is an ever-growing field with applications in various domains [HS12, NTM$^+$13].

There are several approaches that incorporate the temporal evolution of subgraphs to study and characterize networks. Given the computationally demanding nature of the involved computations, very small or very specific classes of subgraphs are typically used. One example of this are triangles, which are meaningful for many applications since they are the simplest communities. Buriol et al. [BFL$^+$06] and Pavan et al. [PTTW13] put forward a method to extract approximate and exact counts of all triangles in graph streaming environments. Finocchi et al. [FFF14] proposed an algorithm to count cliques for sizes slightly larger than 3. Instead of triangles, Aliakbarpour et al. [ABG$^+$16] focused on star shaped graphs. In this thesis, we are more interested in approaches that count any type of subgraphs, and not just specific subgraphs.

Counting subgraphs in temporal networks requires a definition of what temporal subgraphs are. In our case, we use temporal networks as a sequence of network snapshots (as defined at the start of Section 2.1). Thus, a direct translation of subgraph counting from static networks to temporal networks is to simply count subgraphs in each network snapshot and consider a time-series of the results. Then, the time-series of the subgraph counts (e.g., motifs or graphlets) of different networks are compared [BBY09, FM14]. This approach, however, does not offer a real inter-snapshot relation since the subgraphs are static.

Another option is to, instead of treating each snapshot individually, consider structural changes between snapshots. This is the approach that we follow in Chapter 3.2. In short, instead of computing static subgraphs in each snapshot, we compute each subgraph individually and analyze how it evolves (i.e., how its structure changes or stabilizes) in subsequent snapshots (more details on the appropriate chapter).

Martin et al. [MDCR$^+$16] proposed a metric to evaluate network similarity based on how their triplets are evolving over time. Their metric is based on the loss or gain of edges from one state to the next. They differentiate networks by increase or decrease of total edges between states (i.e. different pair-wise transitions are not differentiated as long as they affect the same number of edges). The approach by Doroud et al. [DBWF11] is more similar to our own since they enumerate all transitions between 3-node directed subgraphs in network snapshots. That information is used in order to estimate the probability of a given transition in a social network and predict network changes. Kim et al. [KKK$^+$12] also count all 3-node directed subgraphs to assess which motifs are present in different states of developing gene networks in different regions. These approaches are however limited to 3-node subgraphs and do not consider the roles of the individual nodes, that is, the orbits.

Other approaches do not represent temporal networks as snapshots but instead as a sequence of events. Using this definition, Zhao et al. [ZTH$^+$10] propose communication motifs, an extension of network motifs. Each edge has an "alive" period of Δt time-units and, likewise, a subgraph is "alive" when all of its edges are "alive". If a subgraph is persistently "alive", it is considered a communication motif.

Temporal motifs [KKK$^+$11] extend communication motifs to account for the order of the events. When the order of the events is ignored, the two temporal subgraphs shown in Figure 2.11 are the same, but temporal motifs recognize the two as different. Like static network motifs, temporal motifs require null models to evaluate over-representation[4]. Developing suitable null models for temporal networks is even harder than for static networks. For instance, if the events are sparse enough, and the null model simply randomly reshuffles edges, any larger temporal subgraphs will be considered over-represented by temporal motifs, thus not much information is gained from applying the null model [HS12]. Significant care is needed to guarantee that meaningful properties of the original temporal network are kept in the randomized networks, and remains an important research topic.

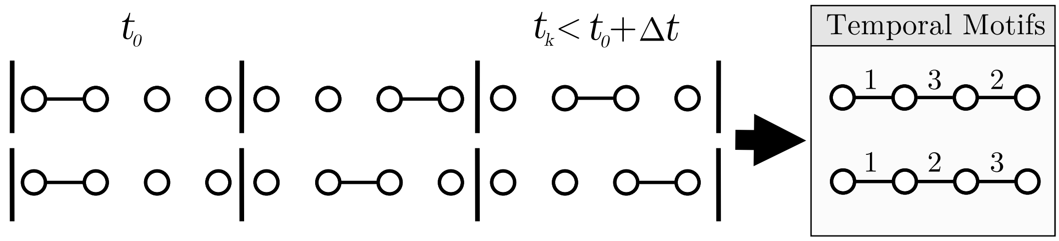


Figure 2.11: Temporal motifs, like communication motifs, have a threshold Δt of "aliveness" for each event (i.e., edges from t_0 are only "alive" until $t_0 + \Delta t$). A set of touching events (i.e., events that share vertices) are considered a subgraph if all of its edges are "alive". Temporal motifs are richer than communication motifs since they account for the order of the events (e.g., the two temporal motifs shown represent the same communication motif).

[4] [PBL17] propose fast algorithms for temporal motif discovery and they do not assess motif over-representation on a null model. However, this definition deviates from the original one by Milo et al. [MSOI$^+$02].

Dynamic graphlets [HCM15], like static graphlets, do not use a reference null model. Like temporal motifs, dynamic graphlets consider the order of the events, but dynamic graphlets also consider the temporal orbits that the nodes occupy in the subgraph. Dynamic GDVs (DGDVs) describe the node's neighborhood in a temporal network (Figure 2.12). Comparing nodes' DGDVs yields a measure of similarity between the nodes' evolving neighborhoods.

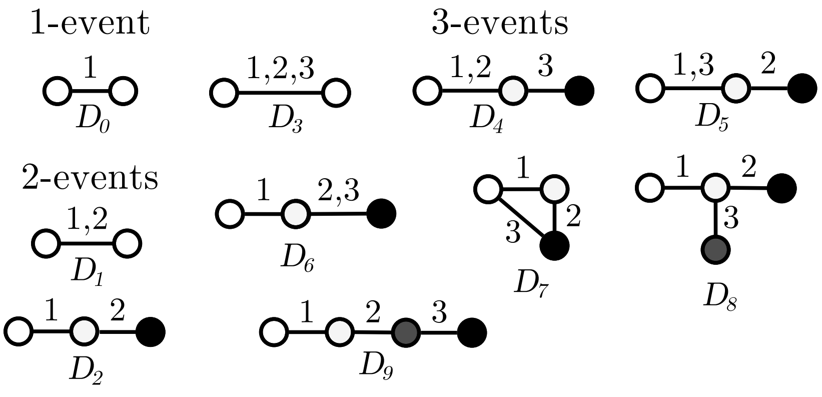


Figure 2.12: Dynamic graphlets with up to 3-events. Edge numbers represent the order of events and node colors represent temporal orbits [adapted from [HCM15]].

2.3 Network classification

Input. A network set $\mathcal{N}$, where each network $G \in \mathcal{N}$ is labeled with one class $c_n \in \mathcal{C}$, represented by $c(G)$. Different sets of features, each represented by $\mathcal{F}_i$. The dimension of $\mathcal{F}_i$ (i.e., its number of features) is represented by $|\mathcal{F}_i|$. The superset of all $\mathcal{F}_i$ is represented as $\hat{\mathcal{F}}$. Features can be any node properties, such as the clustering coefficient, average path length, or GDVs.

Problem statement. For each $\mathcal{F}_i \in \hat{\mathcal{F}}$ and for each $G \in \mathcal{N}$, compute feature vector $f_i(G)$. Then, for each pair of networks $G_a, G_b \in \mathcal{N}$, compute the similarity ϕ_i of their respective f_i, represented by:

$$\phi_i = \phi(f_i(G_a), f_i(G_b)) \tag{2.3}$$

We assume that $\phi_i \in [0..1]$, thus either (a) the similarity measure itself should produce values between 0 and 1 or (b) min-max normalization needs to be performed. For most tests we use the cosine similarity (cos), defined as:

$$\phi(f_i(G_a), f_i(G_b)) = cos(f_i(G_a), f_i(G_b)) = \frac{\sum_{d=1}^{|\mathcal{F}_i|} f_i(G_a)[d] \times f_i(G_b)[d]}{\sqrt{\sum_{d=1}^{|\mathcal{F}_i|} (f_i(G_a)[d])^2} \times \sqrt{\sum_{d=1}^{|\mathcal{F}_i|} (f_i(G_b)[d])^2}} \tag{2.4}$$

We compute a similarity matrix SM_i between all $G_a, G_b \in \mathcal{N}$ for each $\mathcal{F}_i$, resulting in $|\hat{\mathcal{F}}|$ similarity matrices, each with dimension $|\mathcal{N}| \times |\mathcal{N}|$. Each network G is represented by its class $c(G)$. Thus, a good $\mathcal{F}_i$ should lead to high similarity ϕ_i between networks of the same class and low similarity ϕ_i between networks of different classes.

Output: The feature vector $\mathcal{F}_i \in \hat{\mathcal{F}}$ that classifies networks more accurately.

As an example, suppose that we have two classes c_1 and c_2, each comprised of two networks as input ($\{C_{1,1}, C_{1,2}\}$ and $\{C_{2,1}, C_{2,2}\}$, respectively). Now suppose that one of the feature vectors $\mathcal{F}_i$ produces a *perfect* similarity matrix (i.e., networks of the same class are identical and networks of different classes are opposite) and another feature vector $\mathcal{F}_j$ produces a *regular* similarity matrix (i.e., networks are indistinguishable) (Table 2.2). The cell coloring indicates that the networks are of the same class c_n and thus should have high similarity, i.e., $\phi_i \approx 1$, while white cells are networks of different classes and thus should have low similarity, i.e., $\phi_i \approx 0$.

	$C_{1,1}$	$C_{1,2}$	$C_{2,1}$	$C_{2,2}$
$C_{1,1}$	**1**	**1**	0	0
$C_{1,2}$	**1**	**1**	0	0
$C_{2,1}$	0	0	**1**	**1**
$C_{2,2}$	0	0	**1**	**1**

(a) Perfect.

	$C_{1,1}$	$C_{1,2}$	$C_{2,1}$	$C_{2,2}$
$C_{1,1}$	**1**	**0.5**	0.5	0.5
$C_{1,2}$	**0.5**	**1**	0.5	0.5
$C_{2,1}$	0.5	0.5	**1**	**0.5**
$C_{2,2}$	0.5	0.5	**0.5**	**1**

(b) Regular.

Table 2.2: Similarity matrices between two instances of two classes c_1 and c_2: (a) perfectly separates the two classes and (b) can not distinguish between them.

In this toy example, the feature vector that produced matrix (a) is *better* than the one that produced matrix (b). We describe next how we evaluate classification in a multi-class problem since, in general, our datasets comprise more than two classes.

Evaluation: If two networks have the same class ($c(G_a) = c(G_b)$), a positive value is added to the *true* set ($True(G_a, G_b)^+$). Otherwise, a negative value is added

($True(G_a, G_b)^-$). The *true* set is thus comprised of $|\mathcal{N}| \times |\mathcal{N}|$ labels, one for each pair of networks.

If the similarity score ϕ_i (Equation 2.3) between two networks is *high*, the prediction is that they belong to the same class, thus a positive value is added to the *predicted* set ($Pred(G_a, G_b)^+$). Otherwise, the prediction is that they belong to different classes, thus a negative value is added ($Pred(G_a, G_b)^-$). The *predicted* set is thus also comprised of $|\mathcal{N}| \times |\mathcal{N}|$ labels, one for each pair of networks.

We use similar multi-class evaluation measures as Godbole, S. and Sarawagi, S. [GS04].

$$\text{Precision} = \frac{|Pred(G_a, G_b)^+ \cap True(G_a, G_b)^+|}{|Pred(G_a, G_b)^+|}, \forall G_a, G_b \in \mathcal{N} \tag{2.5}$$

$$\text{Recall/True Positive Rate (TPR)} = \frac{|Pred(G_a, G_b)^+ \cap True(G_a, G_b)^+|}{|True(G_a, G_b)^+|}, \forall G_a, G_b \in \mathcal{N} \tag{2.6}$$

$$\text{False Positive Rate (FPR)} = \frac{|Pred(G_a, G_b)^+ \cap True(G_a, G_b)^-|}{|True(G_a, G_b)^-|}, \forall G_a, G_b \in \mathcal{N} \tag{2.7}$$

Since it is subjective to say that similarity is *high*, we compute Precision-Recall and ROC curves. A variable ϵ dictates if two networks' similarity is high, i.e., if they are predicted to be of the same class. To obtain the curves, ϵ is incremented in small steps s, such that $\epsilon \in [0..1]$ and, at each step, the predictions are made:

$$Pred(G_a, G_b, \phi_i, \epsilon) = \begin{cases} Pred(G_a, G_b)^+, & \text{if } \phi_i \geq \epsilon \\ Pred(G_a, G_b)^-, & \text{otherwise} \end{cases} \tag{2.8}$$

In the extreme cases: (a) if $\epsilon = 1$, the two networks have to be exactly alike to be considered of the same class and (b) if $\epsilon = 0$, the networks are always considered to be of the same class. In our experiments, we increase ϵ in increments of 0.001, thus taking 1000 steps to traverse the full range between 0 and 1.

Precision, Recall/TPR and FPR are calculated for each level (Table 2.3). Then, we calculate area under (i) the Precision-Recall curve (AUPR) and (ii) the ROC curve (AUROC). The best feature vector $\mathcal{F}_i$ is the one that produces similarities ϕ_i with highest AUPR and AUROC.

ϵ	Precision	Recall
0	$Precision(\epsilon)$	$Recall(\epsilon)$
0.001	$Precision(\epsilon)$	$Recall(\epsilon)$
0.002	$Precision(\epsilon)$	$Recall(\epsilon)$
...	...	...
1	$Precision(\epsilon)$	$Recall(\epsilon)$

(a) Precision × Recall.

ϵ	TPR	FPR
0	$TPR(\epsilon)$	$FPR(\epsilon)$
0.001	$TPR(\epsilon)$	$FPR(\epsilon)$
0.002	$TPR(\epsilon)$	$FPR(\epsilon)$
...	...	...
1	$TPR(\epsilon)$	$FPR(\epsilon)$

(b) TPR × FPR.

Table 2.3: Evaluation measures computed for different ϵ, used to build (a) Precision-Recall curves and (b) ROC curves. Note that parameter ϵ affects the *predicted* set but not the *true* set from Equations 2.5, 2.6, and 2.7.)

2.4 Network alignment

Input. Two networks G and H, where $|\mathcal{V}(H)| \leq |\mathcal{V}(G)|$.

Problem statement. Produce a mapping $f : \mathcal{V}(G) \rightarrow \mathcal{V}(H)$ that maximizes *structural conservation*. Graph isomorphism is a specific case of network alignment (NA) where $|\mathcal{V}(H)| = |\mathcal{V}(G)|$, and G and H need to be *perfectly conserved* after alignment, i.e., all their edges match must match, otherwise G and H are not isomorphic (see Section 2.1).

Consider two non-isomorphic graphs with four nodes (Figure 2.13); NA aims to find the best possible mapping out of the 4! possible ones.

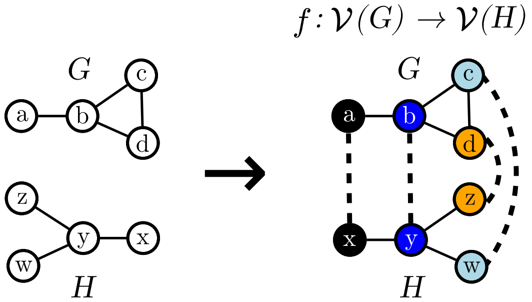


Figure 2.13: Network alignment of two graphs with four nodes.

Output. A value between 0 and 1 of how similar the aligned networks are.

Evaluation. Structural conservation can be measured in different ways. Typically, *edge conservation* is used in combination with *node conservation*.

Regarding edge conservation, recall that graph isomorphism aims to obtain $f : \mathcal{V}(G) \rightarrow \mathcal{V}(H)$, such that $\forall (u, v) \in \mathcal{E}(G) : (f(u), f(v)) \in \mathcal{E}(H)$, i.e., all edges are conserved. In

NA, not all edges are necessarily conserved. Consider E_c as the number of conserved edges, i.e., $E_c = |\{(u, v) \in \mathcal{E}(G) : (f(u), f(v)) \in \mathcal{E}(H)\}|$, and E_{nc} as the number of non-conserved edges, i.e., $E_{nc} = |\{(u, v) \in \mathcal{E}(G) : (f(u), f(v)) \notin \mathcal{E}(H)\}|$. Simple measures, such as S^3 [SM14], compute the ratio of conserved and non-conserved edges (Figure 2.14).

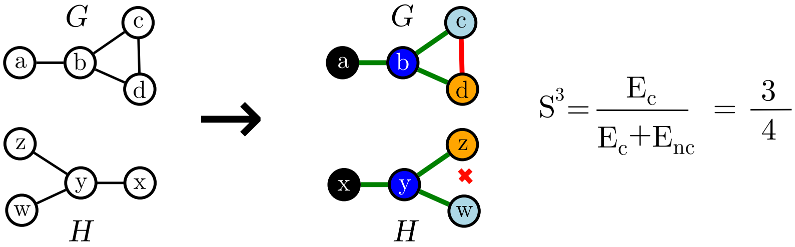


Figure 2.14: Edge conservation between two aligned networks. Green edges are conserved and red edges are non-conserved.

Regarding node conservation, NA aims to preserve the topological features of aligned nodes. Each node $u \in \mathcal{V}(G)$ and $v \in \mathcal{V}(H)$ is represented by a set of node features $\mathcal{F}$. For instance, $\mathcal{F}$ can contain the node degree, the clustering coefficient, and the betweenness centrality. Thus, each node x is represented by feature vector $fv(x) = \{\text{degree}(x), \text{clust-coef}(x), \text{betweenness}(x)\}$. When aligning two nodes $u \in \mathcal{V}(G)$ and $v \in \mathcal{V}(H)$, node conservation measures how similar $fv(u)$ and $fv(v)$ are, computing their Euclidean distance or cosine distance, for instance.

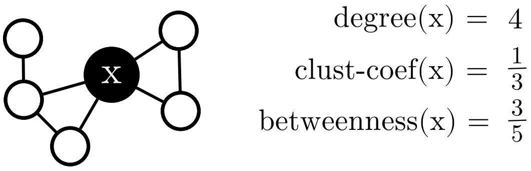


Figure 2.15: Topological properties of a node x.

Choosing a set of features to evaluate node conservation can be challenging, since numerous measures exist. A possibility is to choose a large set of mostly unrelated features, extract them, and do feature selection to analyze desired properties [CPS$^+$16]. Another option is to extract graphlets up to a certain size and use them as features [MNHP10, SM14, SCTM15].

NA maximizes an objective function $c(G, H)$ which measures both *edge conservation* (ec) and *node conservation* (nc),

$$c(G, H) = \alpha \cdot ec(G, H) + (1 - \alpha) \cdot nc(G, H) \tag{2.9}$$

α is a parameter between 0 and 1, controlling the relative importance of node conservation against edge conservation, e.g., if $\alpha = 0$, only node conservation is considered; if $\alpha = \frac{1}{2}$, edge and node conservation are considered equally; etc. High $c(G, H)$ means that the aligned networks are similar.

2.5 Node ranking

Input. A network $G = \{\mathcal{V}, \mathcal{E}\}$. A ground-truth score $Gs(v)$ for each node $v \in \mathcal{V}$.

Problem statement. Calculate a produced score $Ps(v)$ for each node $v \in \mathcal{V}$ without knowing the $Gs(v)$.

Output. A produced ranking Pr, which is a vector of the positions of each node in the ranking, ordered from highest to lowest $Ps(v)$.

Evaluation. We compare the produced ranking Pr with the ground-truth score Gs using normalized discounted cumulative gain (NDCG) and the mean reciprocal rank (MRR). Both NDCG and MRR are computed in relation to the Top-n nodes, represented by NDCG@n and MRR@n, respectively (e.g., NDCG@5 is computed only for the Top-5 ranked nodes of the produced ranking).

Consider a toy example with five nodes v with a certain ground-truth $Gs(v)$ (Table 2.4). The ideal ranking is represented as $Ir(v)$ and a produced ranking is represented as $Pr(v)$.

v	$Gs(v)$	$Ir(v)$	$Pr(v)$
1	6	#2	#5
2	5	#3	#4
3	5	#4	#3
4	8	#1	#2
5	2	#5	#1

Table 2.4: Example of node ground-truth and ranking.

NDCG is computed in two steps [JK02]. In the first step, the DCG is calculated as shown in Equation 2.10. For each position $p \in \{1..n\}$, we find the node v ranked in position p in the produced ranking Pr. Then, we use the ground-truth score $Gs(v)$ as an indicator of a good or bad rank (i.e., a Pr that puts nodes with high Gs in the top-n positions has higher DCG). This value is divided by $log(p + 1)$ because, otherwise, placing a node in any of the top-n positions would be valued equally.

$$\text{DCG@}n = \sum_{p=1}^{n} \frac{Gs(v)}{log(p+1)}, Pr(v) = p \tag{2.10}$$

In the second step, NDCG is obtained by normalizing the value of the produced ranking by that of the ideal ranking (Equation 2.11), thus NDCG@$n \in [0, 1]$.

$$\text{NDCG@}n = \frac{\sum\limits_{p=1}^{n} \frac{Gs(v)}{log(p+1)}, Pr(v) = p}{\sum\limits_{p=1}^{n} \frac{Gs(u)}{log(p+1)}, Ir(u) = p} \tag{2.11}$$

MRR@n is obtained by computing the average ground-truth rank of the top-n ranked nodes by the produced ranking (Equation 2.12).

$$\text{MRR@}n = \frac{\sum\limits_{p=1}^{n} Ir(v), Pr(v) = p}{n} \tag{2.12}$$

Typically, MRR is not normalized; thus MRR@$n \in [\frac{\sum_{p=1}^{n} p}{n}, \frac{\sum_{p=1}^{n} \mathcal{V}(G)-p+1}{n}]$.

Considering the toy example from Table 2.4:

$$\text{NDCG@}5 = \frac{\frac{2}{log(2)} + \frac{8}{log(3)} + \frac{5}{log(4)} + \frac{5}{log(5)} + \frac{6}{log(6)}}{\frac{8}{log(2)} + \frac{6}{log(3)} + \frac{5}{log(4)} + \frac{5}{log(5)} + \frac{2}{log(6)}} \approx \frac{14.02}{17.21} \approx 0.81$$

$$\text{MRR@}2 = \frac{5+1}{2} = 3$$

$$\text{MRR@}4 = \frac{5+1+4+3}{4} = 3.25$$

Rankings that have higher NDCG@n and lower MRR@n are better.

Chapter 3

Network classification

It is believed that network structure (or topology) and network function are intrinsically related [vdHSKP09]. For instance, social networks have similar network motifs, and linguistic networks also have similar network motifs, but social and linguistic networks have different network motifs between them [vdHSKP09]. Network motifs are just one possibility to compare network structure.

Here we assume that the hypothesis "*topological similarity* indicates *functional similarity*" is true. Thus, a method that measures topological similarity should yield high values for functionally similar networks (i.e., networks of the same *class*) and low values for functionally dissimilar networks (i.e., networks of different classes). In the previous example, network motifs were a *good* method because different networks (i.e., social versus linguistic networks) had clear topological differences.

In this chapter we propose new methods to measure topological similarity in directed and temporal networks. We evaluate the methods both on synthetic networks and on real networks. On synthetic networks we know for certain that topology indicates function, thus these tests are necessary to evaluate the methods, while tests on real networks might lead to new findings.

3.1 Network classification of directed networks

3.1.1 Motivation

Numerous measures can be used to evaluate and compare network topology. Simple statistics such as the degree distribution, clustering coefficient, or the average path length are often used to get an initial feel of the network's structure [ARS$^+$07].

Subgraph-based measures offer rich topological information. Motifs and graphlets are small non-isomorphic subgraphs, but graphlets account for the position (or orbit) of the nodes in the subgraph while network motifs do not. Another difference between graphlets and networks motifs is that the latter require a null model to verify if the subgraphs are over-represented. Usually the null model is an artificial random network that maintains the original network's node degree sequence. On the other hand, graphlets do not require a null model and use the information of all subgraphs to perform a full-scale network comparison.

Graphlets are induced subgraphs. Sometimes network motifs are used as partial occurrences [SS04] however, like graphlets, researchers use them more commonly to account only for induced occurrences [MSOI$^+$02, RS10b]. Contrarily to partial occurrences, induced occurrences are not ambiguous because existing and non-existing edges are given equal importance, leading to more revealing and less convoluted results.

Graphlet usage is often restricted to analyzing only the set of 30 undirected graphs of up to five nodes (Figure 2.5), originally presented by [Prž07a], due to computational limitations. However, a substantial gain in topological information might be attained by examining different sets of graphlets. One possibility is to enumerate larger graphlets since, by definition, they capture more topological information about the network's structure than smaller graphlets, and this added information might be valuable. For instance, Hulovatyy et al. observed that larger graphlets of 6 or 7 nodes led to a higher accuracy for node classification in dynamic networks than smaller ones [HCM15]. Additionally, in directed networks, edge direction should be taken into account since it can potentially reveal information about the network's structure that undirected graphlets do not capture. Despite the fact that graphlets are a general model, at the time of our proposed extension for directed networks [ARS15], the only other extensions to the original concept were relative to ordered graphlets [MDP14] and dynamic graphlets [HCM15]. The latter work by Hulovatyy also goes beyond the usual 5-node graphlets (up to size 7). Since then, another extension for directed networks was proposed [SMDYP16].

In the case of biological networks, for instance, many cellular biological networks are intrinsically directed such as metabolic, cell signaling, and gene transcriptional regulation networks. Methods and metrics that ignore the edge direction of these networks might be losing important information. Garlaschelli and Lofredo [GL04] proposed a new measure ρ to calculate the link reciprocity of a network, which can be used to assess if its edge direction is important or not. Their measure is an absolute quantity ranging from -1 (no reciprocity) to 1 (completely reciprocal). Networks with $\rho \approx -1$ are purely directional networks, meaning that edge direction is an intrinsic aspect of these networks and removing it makes them meaningless. On the other hand, networks with $\rho \approx 1$ can be safely transformed into topologically equivalent undirected networks without losing much information since their edges are always reciprocally connected. Garlaschelli and Loffredo calculated that cellular and food web networks rank closer to the middle of the scale ($\rho \approx 0$) meaning that edge direction in these networks is significant. Additionally, some specific small directed graphs, such as feed-forward loops, have been shown to play a fundamental role in the organization of distinct networks [MA03].

Network motifs have been extensively used to study directed biological networks such as neural, transcriptional, and signal networks [WLY14]. Graphlets on the other hand are mostly restricted to undirected networks since, as mentioned previously, they consist of a set of undirected graphs. Park et al. [PHHT10] examined numerous directed biological networks using both directed motifs and undirected graphlets. Such studies could have been enriched if a tool for enumerating directed graphlets was available at the time.

3.1.2 Overview of our contribution

In this chapter we present GT-Scanner[1], an efficient general-purpose tool to enumerate and compare both directed and undirected GDVs. Furthermore, GT-Scanner can be used to enumerate arbitrarily large subgraphs (as long as they fit into memory) since it is not targeted for specific graphlets. Previous approaches either restricted the application of graphlets to undirected networks or had to ignore edge direction in directed networks, in practice reducing them to undirected networks. To achieve this objective we extend both i) the original concept of graphlets to directed graphlets (Section 3.1.3) and ii) upgrade a tree data-structure specialized in efficiently storing

[1] GT-Scanner is available at `www.dcc.fc.up.pt/~daparicio/software`.

graphs, the g-trie, to a graphlet-trie (Section 3.1.4). GT-Scanner can thus be used to enumerate directed and undirected graphlets as well as network motifs. GT-Scanner is an extension of a previous tool[2] which did not consider orbits.

To assess the applicability and performance of GT-Scanner, we organize a set of experiments into two parts: i) classification accuracy on synthetic data (Section 3.1.5) and ii) performance evaluation on real biological data (Section 3.1.6). The first measures how well directed graphlets can group a set of directed networks and compares it with undirected graphlets, while the latter analyses the performance of our tool on a set of directed biological networks of different types (biological functions) by comparing its execution time with state-of-the-art approaches.

3.1.3 Directed graphlets

Graphlets [Prž07a] are small induced non-isomorphic subgraphs that include information about the position or orbit that nodes occupy in the graphlet (more details in Section 2.1). For instance, considering graphlet G_{11} from Figure 2.5, a node in the center of a star-graph (i.e., in orbit O_{23}) is different from the nodes on its periphery (i.e., in orbit O_{22}). A set containing a specific set of graphlets is referred to as $\mathcal{G}$, and the corresponding set of orbits as $\mathcal{O}$. Again, considering only G_{11}, then $\mathcal{G} = \{G_{11}\}$ and $\mathcal{O} = \{O_{22}, O_{23}\}$. In practice, $\mathcal{G}$ contains not just a single graphlet but all graphlets with $\leq k$ nodes, represented by $\mathcal{G}_k$. We use notation $d\mathcal{G}k$ and $d\mathcal{O}_k$ for directed graphlets and directed orbits, respectively, and $u\mathcal{G}_k$ and $u\mathcal{O}_k$ for the undirected counterparts.

We use graphlets to obtain a graphlet-degree distribution (GDD) of a network (Section 2.1). A GDD can be regarded as a set of (topological) features that describe a given network (Table 3.1). For a given network G, $d_j^G(k)$ from GDD_G denotes how many nodes appear k times in orbit j. k ranges from 0 (i.e., no nodes appear k times in orbit j) to $+\infty$ (i.e., not numerically bounded).

$$GDD_G = \begin{bmatrix} d_G^0(1) & d_G^0(2) & \cdots & d_G^0(+\infty) \\ d_G^1(1) & d_G^1(2) & \cdots & d_G^1(+\infty) \\ \vdots & \vdots & \ddots & \vdots \\ d_G^m(1) & d_G^m(2) & \cdots & d_G^m(+\infty) \end{bmatrix}$$

Table 3.1: Graphlet-degree distribution (GDD) matrix.

[2] Available at `www.dcc.fc.up.pt/gtries`.

We compare two networks G and H by measuring the distance between their respective GDD matrices. GDD-agreement (GDA) is typically used for this task [Prž07a, ZYHW13, YFCM17]. Other possible measures to compare graphlets include relative graphlet frequency distance (RGFD) [PCJ06] and graphlet correlation distance (GCD) [YMDD$^+$14]. However, RGFD disregards orbits (i.e., it only considers graphlets' frequencies) and GCD requires a pre-processing step where a set of equations is calculated for $\mathcal{G}$ (i.e., obtain the correlations between graphlets in $\mathcal{G}$). Thus, we use GDA to compare networks in our experiments. GDA consists in four steps: (1) reduce the weight of high degrees (Equation 3.1), (2) normalize the GDD matrices by row/orbit (Equation 3.2), (3) for each row/orbit, compute the orbit-GDA between the two networks (Equation 3.3), and (4) average out the orbit-GDA into a global GDA (Equation 3.4). Step 1 acts as a decay function, reducing the weight of nodes in the tail of the graphlet-distribution. Step 2 ensures that all orbits have the same weight (i.e., $\forall j, \sum \lim_{k} s^j_G(k) = 1$), thus, rare orbits are as important for the GDA as common orbits (otherwise common orbits would dominate the GDA). Thus, GDA's assumption (and ours) is that rare orbits are important to differentiate two networks. Step 3 computes the Euclidean distance of G's and H's GDDs for each orbit and normalizes it between 0 and 1 (i.e., $GDA(G, H)^j \in [0, 1]$). Finally, step 4 computes the arithmetic mean of the orbit-GDAs, thus $GDA(G, H) \in [0, 1]$.

$$s^j_G(k) = \frac{d^j_G(k)}{k} \qquad (3.1) \qquad\qquad n^j_G(k) = \frac{s^j_G(k)}{\sum\limits_k s^j_G(k)} \qquad (3.2)$$

$$GDA^j(G, H) = \frac{1}{\sqrt{2}} \sqrt{\sum_k (n^j_G(k) - n^j_H(k))^2} \qquad (3.3)$$

$$GDA(G, H) = \frac{1}{|\mathcal{O}|} \sum_{\mathcal{O}} GDA(G, H)^j \qquad (3.4)$$

When GT-Scanner compares two (or more) networks, it outputs $GDA(G, H)$ as well as GDD_G and GDD_H. A high $GDA(G, H)$ means that G and H are topologically similar. Since GT-Scanner can calculate both directed and undirected GDAs, $uGDA_k$ and $dGDA_k$ represent the GDAs when comparing undirected and directed graphlets, respectively, of size k.

We adjust the original GDA measure to only consider orbits that appear in at least one of the networks. For simplicity and clarity, we refer to the original measure as GDA' and our own as GDA. This modification in GDA is necessary since non-appearing orbits lead to unreasonably high GDA's when a large number of orbits are enumerated

	Undirected graphlets		Directed graphlets	
k	$u\|\mathcal{G}_k\|$	$\|u\mathcal{O}_k\|$	$\|d\mathcal{G}_k\|$	$\|d\mathcal{O}_k\|$
2	1	1	2	3 (1.5 × $\|d\mathcal{G}_k\|$)
3	3	4	15	33 (2.3 × $\|d\mathcal{G}_k\|$)
4	9	15	214	730 (3.5 × $\|d\mathcal{G}_k\|$)
5	30	73	9,578	45,637 (4.8 × $\|d\mathcal{G}_k\|$)
6	142	480	1,540,421	9,121,657 (5.9 × $\|d\mathcal{G}_k\|$)
7	965	4,786	882,011,563	$\approx 7 \times \|d\mathcal{G}_k\|$
8	12,082	77,275	1,793,355,966,869	$\approx 8 \times \|d\mathcal{G}_k\|$
9	273,162	2,188,288	13,027,955,038,433,121	$\approx 9 \times \|d\mathcal{G}_k\|$

Table 3.2: Number of undirected and directed graphlets, as well as their respective orbits, depending on the size of the graphlets. For each case, we count all graphlets of sizes 2..k. It is impractical to enumerate all possible orbits for $d\mathcal{G}_k$ when k is larger than 6 due to the size of $|\mathcal{O}_k|$.

or when small networks are used. This happens because, by definition, the GDA' of two networks increases if the orbit frequency is zero in both networks. Hulovatyy et al., which used bigger graphlets than the usual 5-node undirected subgraphs, suggested a similar explanation [HCM15]. This is not very problematic when few orbits are enumerated, such as the original 73 undirected ones (Figure 2.5); however, bigger undirected graphlets ($k > 5$) and directed graphlets require thousands or millions of orbits to be enumerated (Table 3.2). For these cases with more orbits, it is likely that many of the possible orbits do not appear in either network, which may result in higher GDA's than expected. For instance, on tests comparing pairs of small food webs [BM06], we obtained an average GDA' of ≈ 0.5 when enumerating $d\mathcal{G}_4$, and an average GDA' of ≈ 0.85 when enumerating $d\mathcal{G}5$. This does not mean that two food webs are much more alike when looking at their larger graphlets but rather that the GDA' measure is not well-suited, because there are many orbits that do not appear in either food web. Figure 3.1 illustrates the difference in agreement values given by the original GDA' metric and our own GDA.

Another problem lies in the huge number of possible orbits (i.e., $|\mathcal{O}_k|$) that directed graphlets with more than 5 nodes have (i.e., $k > 5$). For instance, when $k = 6$ there are more than 1 million potential orbits (Table 3.2). Assuming that the frequency of each orbit is stored in an 4-byte integer, computing $\mathcal{O}k$ requires $|V(G)| \times 4 \cdot |\mathcal{O}_k|$ bytes of memory. Therefore, enumerating $d\mathcal{G}_5$ on a network with 105 nodes requires $\approx$ 4GB of RAM, which is still feasible in most modern PCs; however, enumerating $d\mathcal{G}6$ is only possible on networks with a few hundred nodes, and if $k \geq 6$ the enumeration is simply unfeasible. A possible way to reduce the memory footprint is to deal with

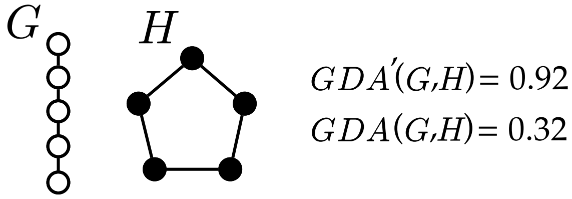


Figure 3.1: Comparison of different GDA measures. The original GDA' was found to produce unreasonably high values for small networks (in the example) or when many graphlets are enumerated. This makes the measure inappropriate to compare directed graphlets since the number of orbits is very high. In our modified GDA only orbits appearing in either G, H or both are considered, discarding non-present orbits.

orbit redundancies [MP08]. Nonetheless, larger values of k still produce too many non-redundant orbits that make the computation unfeasible in terms of memory. Another option is to avoid generating all possible graphlets and orbits before the enumeration and instead build their representation during the enumeration phase as they occur in the network [PR13] since it is reasonable to expect that only a fraction of all possible graphlets/orbits actually appear in a given network. This strategy introduces an overhead in computational time but makes it attainable to analyze larger graphlets.

3.1.4 Graphlet-tries

A g-trie [RS14c](Section 2.2.1.1) is a tree-like data-structure, initially created to calculate network motifs (Section 2.2.5.1), but which efficiently solves the more general subgraph counting problem (Section 2.2). Ribeiro and Silva [RS10b] presented a g-trie algorithm that was one or two orders of magnitude faster than previous approaches, showcasing its efficiency. Since then, faster algorithms have been proposed (Section 2.2.2) but they are specific to certain subgraphs, while g-tries remain one of the fastest algorithms for general subgraph counting (i.e., they can be used to count any subgraph set efficiently).

Graphlet-tries are an extension of g-tries that also consider the nodes' orbits. The broader term g-trie is used whenever a concept applies to both g-tries and graphlet-tries. A graphlet-trie containing all 30 undirected graphlets (Figure 2.5) is shown in Figure 3.2. The original graphlet and orbit numbers from [Prž07a] are kept only for convenience since they are generated automatically in our implementation. All 2 and 3-node directed graphlets are illustrated in Figure 3.3 as well as the non-bidirectional 4-node directed graphlets (the bidirectional graphlets were removed for space concerns). An additional graphlet-trie containing them is presented in Appendix A.

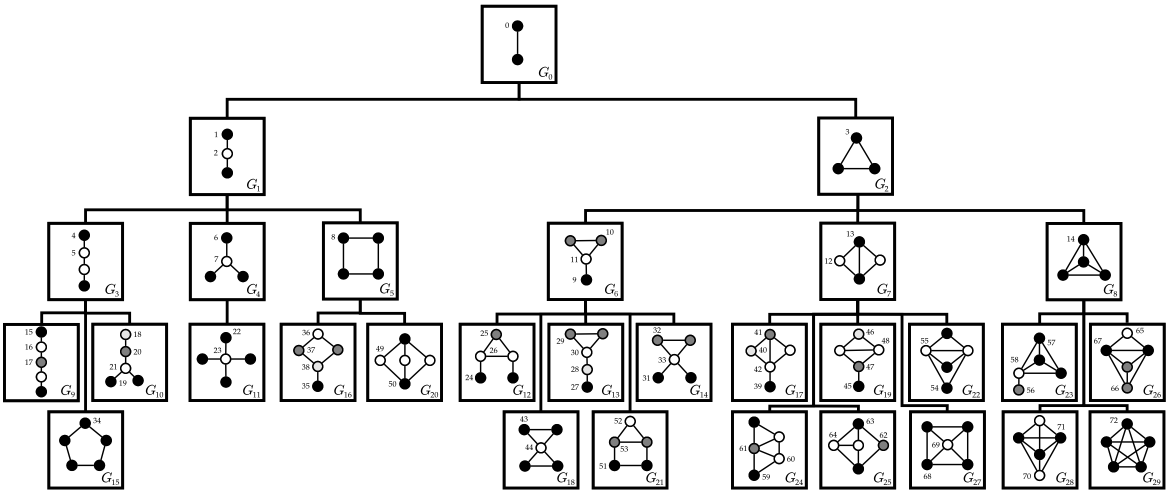


Figure 3.2: A graphlet-trie containing all 2, 3, 4 and 5-node undirected graphlets $G_0, ..., G_{29}$ as they were presented in [Prž07a]. In a g-trie the common topologies between graph(let)s of different sizes become evident.

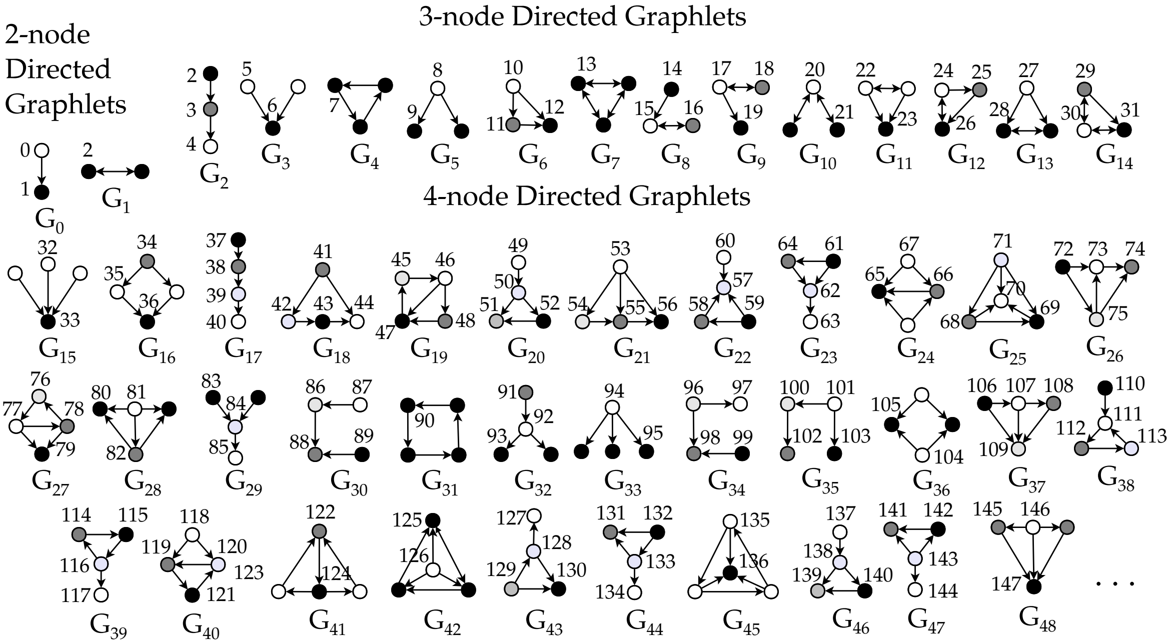


Figure 3.3: A subset of $d\mathcal{G}_4$ containing all 2 and 3-node directed graphlets and the 4-node directed graphlets that have no bidirectional edges (for space concerns).

3.1.4.1 Graphlet-trie creation

A graphlet-trie is built by iterative insertion; Figure 3.4 illustrates the process and Algorithm 3.1 contains the pseudo-code. When graph G_{15} is inserted into an initially empty graphlet-trie (lines 2–4) no common subtopologies are found (lines 6–7) and

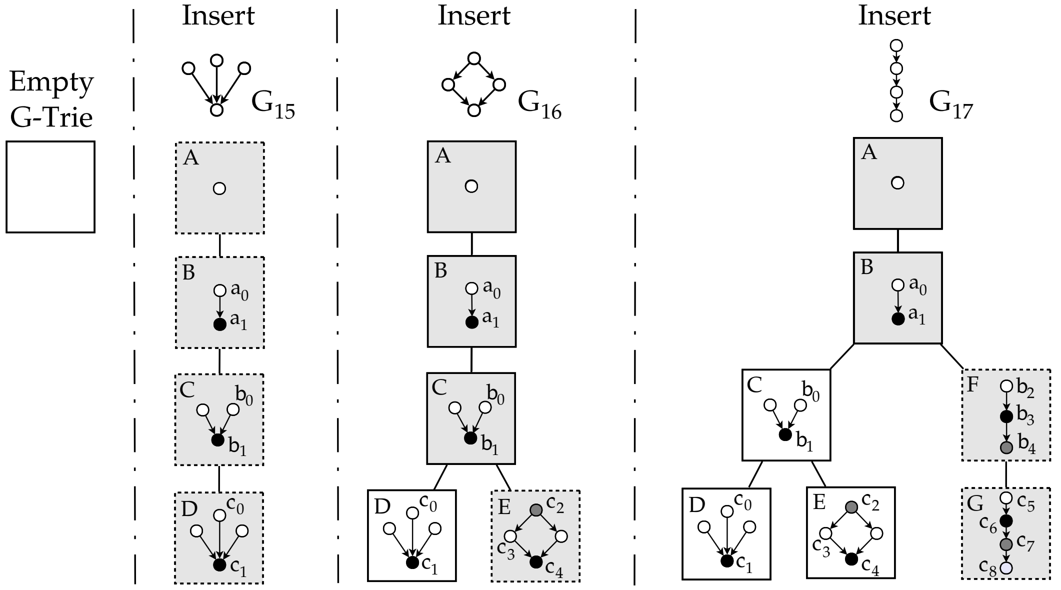


Figure 3.4: Iterative insertion of 3 graphlets to a graphlet-trie. Graphlet-trie vertices colored in gray represent the path to the newly added graphlet. Vertices with dotted contours were added in the most recent insertion.

Algorithm 3.1 Populate a graphlet-trie T with subgraphs $G_i \in \mathcal{G}$

```
 1: procedure CreateGraphletTrie($\mathcal{G}$)
 2:     $T \leftarrow$ EmptyGraphletTrie()
 3:     for all $G_i \in \mathcal{G}$ do
 4:         Insert($T.root, G_i, 1$)
 5: procedure Insert($N, G_i, depth$)
 6:     for all $C_i \in N.children$ do
 7:         if ShareCommonTopology($C_i, G_i, depth$) then
 8:             Insert($C_i, G_i, depth + 1$)
 9:             return
10:     $NewChild \leftarrow N$.addChild($G_i, depth$)
11:     Insert($NewChild, G_i, depth + 1$)
```

four new graphlet-trie nodes need to be created (lines 10–11), A, B, C and D, each containing a subgraph of size equal to the depth level of the graphlet-trie (A has 1 vertex, B has 2 vertices, and so forth). A graphlet-trie, unlike a traditional g-trie, also evaluates the subgraph orbits and stores them alongside the graphlet. When G_{16} is inserted to the graphlet-trie, only node E needs to be added to it since G_{15}, which was previously inserted, and G_{16} share the common path $A \Rightarrow B \Rightarrow C$ in the graphlet-trie (lines 6–9). Finally, G_{17} requires two nodes to be created, F and G, because G_{17} only shares the path $A \Rightarrow B$ with the other two graphlets already in the graphlet-trie.

In this very short example, the compression rate achieved by the graphlet-trie is $\frac{7}{12} \approx$

42% since each 4-node graphlet would require 4 different subgraphs to be added (one for each $k \in \{1, 2, 3, 4\}$, giving a total of 12 graphlet-trie nodes for the 3 subgraphs) but, by using the common topology of the graphlets, only 7 subgraphs are added to the graphlet-trie. Inserting all 4-node undirected graphlets gives a higher compression ratio of $\approx 80\%$ since there are more opportunities for the graphlet-trie (and g-tries in general) to find common topologies between the subgraphs [RS14c].

Usually graphlet enumeration requires subgraph counting of not only size k but also all sizes $s < k$. Traditional g-tries enumerate subgraphs all of the same size k, so we had to adapt graphlet-tries to support $s \leq k$ enumeration. G-Tries guarantee that all subgraphs on the g-trie leaves are non-isomorphic, however the same is not true for non-leaf g-trie nodes (illustrated in Figure A.1 of Appendix A). Thus, repeated occurrences can be found for $s < k$ graphlets. To avoid counting them, we consider isomorphic graphlets as being the same subgraph, and thus only one of them is considered the non-isomorphic class (i.e., while two or more isomorphic graphlets might appear in a graphlet-trie, and all are used for enumeration, only the frequency of one of them is considered). Like g-tries, the set of subgraphs/graphlets that the user wants to query on the network is fully customizable and given as input, making graphlet-tries a very flexible approach.

3.1.4.2 Graphlet-trie enumeration

Figure 3.5 illustrates how we use the graphlet-trie from Figure 3.4 to perform subgraph enumeration. The graphlet-trie search algorithm, GT-Scanner, is essentially a depth-first search algorithm, mapping the input network to the graphlet-trie. Notation $(N, \{v_1, v_2, ..., v_k\})$ represents that vertices $v_1, v_2, ..., v_k$ from the input network are mapped to node N on the graphlet-trie. For simplicity, N refers to both the graphlet-trie node and its respective subgraph.

At the start, GT-Scanner maps each of the seven vertices of the network, one at a time, to the root of the graphlet-trie, which is node A (lines 2–4 of Algorithm 3.2). Starting from $(A, \{a\})$, the search descends on the graphlet-trie and looks for valid candidates (line 7). For an efficient graph traversal, GT-Scanner picks the vertex from V_{used} that is connected to the newly added node in graphlet-trie T (line 12) that has the smallest neighborhood (line 13). At the beginning A is the only possible choice. The candidate vertices $c \in V_{cand}$ are the neighbors of A that respect both the graphlet-trie connections (in this case, candidate c only needs to have an incoming edge from A) and symmetry breaking conditions, which are needed to avoid isomorphic

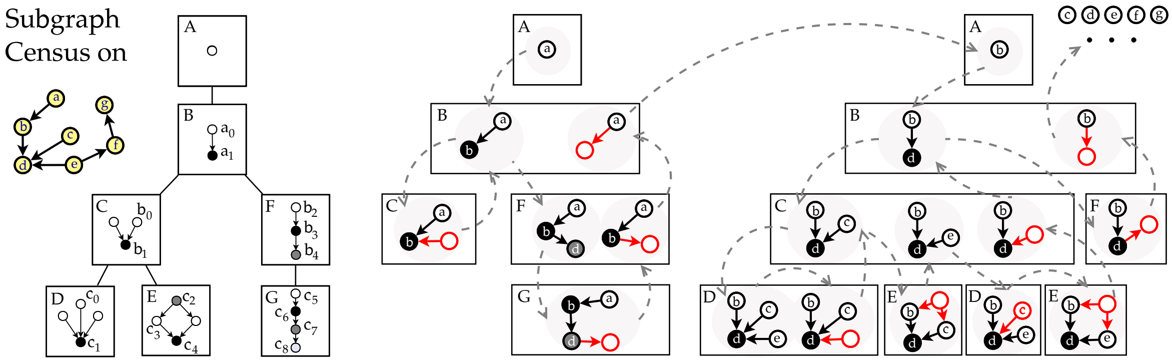


Figure 3.5: Subgraph census using a graphlet-trie. Nodes in red mean that no candidate was found for that particular graphlet. The dotted arrows represent the search path.

Algorithm 3.2 Count all orbits from graphlet-trie T in network G

```
 1: procedure COUNTALL($T, G$)
 2:     for all vertex $v$ of $G$ do
 3:         for all children $c$ of $T.root$ do
 4:             COUNT($c$, $\{v\}$)
 5: procedure COUNT($T, V_{used}$)
 6:     if $T.isNonIsomorphic()$ then
 7:         $T$.ORBITS($V_{used}$)++
 8:     $V \leftarrow$ GETVALIDCANDIDATES($T, V_{used}$)
 9:     for all vertex $v$ of $V$ do
10:         for all children $c$ of $T$ do
11:             COUNT($c, V_{used} \cup \{v\}$)
12: function GETVALIDCANDIDATES($T, V_{used}$)
13:     $V_{conn} \leftarrow$ vertices connected to the vertex being added
14:     $m$      $\leftarrow$ vertex of $V_{conn}$ with smallest neighborhood
15:     $V_{cand} \leftarrow$ neighbors of $m$ that respect both
16:                connections to ancestors and
17:                symmetry breaking conditions
18:     return $V_{cand}$
```

cases (lines 14–17). The first viable candidate is vertex B, so GT-Scanner finds the mapping $(B, \{a, b\})$ and increments the frequency of orbits a and b since B is a valid non-isomorphic graphlet-trie node (line 6). The search process continues in depth-first fashion (lines 8–10) and finds that no valid mapping exists between C and $\{a, b, v_i\}$ since no v_i can be joined to $\{a, b\}$ so that the resulting subgraph forms a subgraph isomorphic to C. Therefore, GT-Scanner backtracks to $(B, \{a, b\})$ and instead looks for a valid mapping of F, finding $(F, \{a, b, d\})$ and incrementing the respective orbit frequencies. GT-Scanner finds no valid mappings for $(G, \{a, b, d, v_i\})$ and backtracks, finding no further alternatives for $(F, \{a, b\})$. GT-Scanner backtracks again and, since it finds no other mappings for $(B, \{a, v_i\})$, it moves on to vertex b. GT-Scanner finds $(A, \{b\})$, $(B, \{b, d\})$, $(C, \{b, d, c\})$, and $(D, \{b, d, c, e\})$ until it has to backtrack since there are no more alternatives for $(D, \{b, d, c, v_i\})$. There are also no occurrences of $(E, \{b, d, c, v_i\})$ so GT-Scanner proceeds and finds $(C, \{b, d, e\})$. When GT-Scanner reaches D, $(D, \{b, d, e, c\})$ is a valid mapping topologically, however it would be the same occurrences as the previously found $(D, \{b, d, c, e\})$. In practice graphlet-tries, and g-tries in general, do not find repeated occurrences thanks to *symmetry breaking* mechanisms embedded in the g-trie nodes (line 16): vertices that appear later in the same orbit are only valid if they have a bigger index than the previous vertices of the same orbit (for more details, see Section 2.2.1.1). After failing to find a valid mapping for $(D, \{b, d, e, v_i\})$, the search also fails for $(E, \{b, d, e, v_i\})$ and keeps backtracking until it proceeds to $(A, \{c\})$, etc.

3.1.5 Classification accuracy on synthetic networks

3.1.5.1 Synthetic directed networks

We evaluate the advantages of using directed graphlets over using undirected graphlets by comparing them in the task of network classification of synthetic networks. We perform these tests on synthetic networks pertaining to different graph models: Erdős-Rényi random graphs (ER) [ER60], scale-free networks (SF) [GKK01], and Forest Fire graphs (FF) [LKF05]. All ER and SF graphs have $\approx 1\%$ edge density, mimicking real world networks such as PPIs, internet routing, and email networks [Mel06]. We create five classes, described next.

We create two classes of directed ER networks, $ER_{\rho=0.2}$ and $ER_{\rho=0.8}$, where ρ is the probability of edges to be reciprocal. In $ER_{\rho=0.2}$ edges are reciprocal 20% of the times, and in $ER_{\rho=0.8}$ edges are reciprocal 80% of the times. For both classes, non-reciprocal edges of v have an equal chance of being an in-edge (u, v) or an out-edge (v, u). While these two classes are indistinguishable if one disregards their edge direction, they are different when we consider it. Thus, an appropriate method should be able to distinguish between the two.

We create two classes of directed SF networks, SF_{ORD} and SF_{RAND}. In SF_{ORD} we transform each undirected edge (u, v) into a similar edge (u, v) if $u < v$ or (v, u) if $v < u$, and in SF_{RAND} we randomly shuffle edge direction (i.e., (u, v) is kept as (u, v) or shuffled to (v, u) with 50% probability). Networks from these two classes have very similar node-degree distributions but very different in- and out-degree distributions.

Finally, we create one class of directed FF networks, FF, with $p = 0.37$ (*forwards burning probability*) and $p_b = 0.32$ (*backwards burning probability*), as suggested by Leskovec et al. [LKF05] in order to build the most realistic networks.

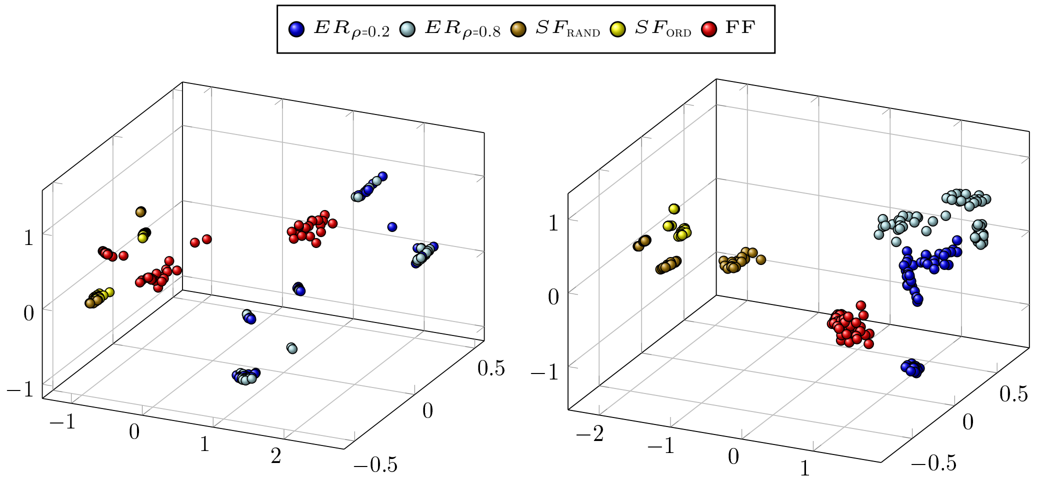
We generate 20 networks of each of the 5 classes, with 500, 1000 or 2000 nodes, for a total 300 networks.

3.1.5.2 Methodology

Section 2.3 describes in detail how network classification is performed and evaluated.

In short, for three distinct sets of graphlets (i.e., features), namely $u\mathcal{G}_4$, $d\mathcal{G}_4$ and $d\mathcal{G}_5$, we compute the GDA for all pairs of networks (i.e., we obtain the respective similarity matrices). We evaluate the performance of each set of graphlets by comparing their obtained precision-recall when classifying the networks. This methodology was previously adopted by [YMDD$^+$14] to demonstrate how well their metric (GCD) could group different undirected networks using undirected graphlets.

3.1.5.3 Classification accuracy



(a) $u\mathcal{G}_4$

(b) $d\mathcal{G}_4$

Figure 3.6: MDS representation applied to the GDA matrices obtained for undirected graphlets $u\mathcal{G}_4$ and directed graphlets $d\mathcal{G}_4$. (a) Undirected graphlets can not appropriately cluster the different networks whereas (b) directed graphlets clearly group the networks correctly.

We illustrate the classification capabilities of both directed and undirected graphlets using multidimensional scaling (MDS) [CC00] in a 3-dimensional space (Figure 3.6). We plot the 300 networks in the MDS-space using (a) $u\mathcal{G}4$ and (b) $d\mathcal{G}4$. Visually, undirected graphlets $u\mathcal{G}4$ successfully distinguish between different directed graph models (i.e., ER vs SF vs FF) despite ignoring their edge direction. This is possible since the topology of the different models is so distinct that edge direction can be disregarded. However, as expected, when the undirected topology of the networks is similar but the directed topology is not, undirected graphlets fail to separate the classes. Directed graphlets $d\mathcal{G}4$ successfully separate networks of different models (i.e., ER vs SF vs FF) and also successfully separate networks of the same model but of different classes, i.e., they separate networks with different levels of reciprocity ($ER_{\rho=0.2}$ from $ER_{\rho=0.8}$) and with distinct in/out-degree distributions (SF_{ORD} from SF_{RAND}). Furthermore, undirected graphlets inadequately separate FF networks by size while directed graphlets do not.

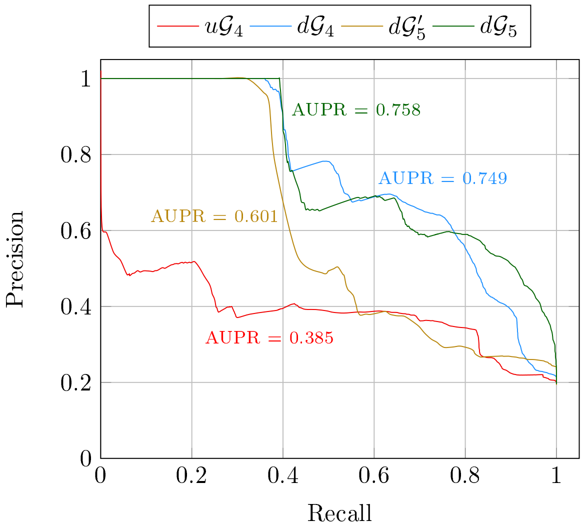


Figure 3.7: Precision-recall curves for undirected graphlets ($u\mathcal{G}_4$) and directed graphlets ($d\mathcal{G}_4$, $d\mathcal{G}_5$ and $d\mathcal{G}'_5$). Undirected graphlets can not correctly group the networks (AUPR = 0.385). Directed graphlets ($d\mathcal{G}_4$ and $d\mathcal{G}_5$) correctly cluster the networks (AUPR $\approx$ 0.75). Non-appearing orbits undermine larger directed graphlets' ($d\mathcal{G}'_5$) capability to classify the networks (AUPR = 0.601 < AUPR = 0.758).

We show precision-recall curves for undirected ($u\mathcal{G}_4$) and directed graphlets ($d\mathcal{G}_4$ and $d\mathcal{G}_5$) in Figure 3.7. They are obtained by computing the similarity matrices using similarity function ϕ as the GDA between each pair of networks and calculating precision-recall for varying thresholds (Section 2.3). The AUPR for $d\mathcal{G}4$ is 0.749, which is $\approx$ 95% higher than the AUPR for $u\mathcal{G}_4$, which is 0.385. We perform two sample Welch t-tests and verify that the results for directed graphlets are statistically significant (p-values ≤ 0.01). We also measure the effect of keeping non-appearing orbits or removing them from the GDA (i.e., using GDA' or GDA, respectively). When GT-Scanner enumerates $d\mathcal{G}_5$ and uses traditional GDA' [Prž07a] (for distinction, this test is referred to as $d\mathcal{G}'_5$) its AUPR is 0.601, and when GT-Scanner enumerates $d\mathcal{G}_5$ and uses our proposed GDA its AUPR is 0.758, a gain of $\approx 26\%$.

In conclusion, our best results on synthetic data were obtained by (i) using directed graphlets instead of undirected graphlets (i.e., $d\mathcal{G} > u\mathcal{G}$), (ii) removing non-appearing orbits from the GDA calculation (i.e., $GDA > GDA'$), and (iii) using larger graphlets (i.e., $d\mathcal{G}_5 > d\mathcal{G}_4$). These experiments show that directed graphlets can be successfully used to group directed networks pertaining to the same class and likewise distinguish between directed networks of different types.

It could be argued that directed graphlets are not necessary to correctly cluster these types of networks since one could simply analyze their reciprocity and the in- and out-degrees to be able to separate them. In these experiments our main concern was to show that undirected graphlets are not suitable to study directed networks. However, directed graphlets offer much richer topological information than reciprocity or degree distributions, in the same way that undirected graphlets give much more details on the network's structure than just the node-degree. In fact, reciprocity and in- and out-degrees are embedded in the 2-node directed graphlets (G_0 and G_1 from Figure 3.3) in the same way that the undirected node-degree is embedded in the 2-node undirected graphlet (G_0 from Figure 2.5). Enumerating directed graphlets of more than 2-nodes captures not only these two basic measures but also the more intricate connections between the nodes.

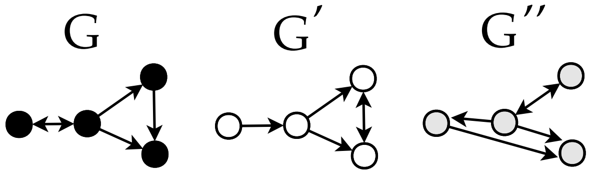


Figure 3.8: Reciprocity or degree distribution information is not sufficient to separate these graphs: G and G' have the same reciprocity, and G and G'' have the same in- and out-degree distributions.

Consider Figure 3.8: networks G and G' have the same reciprocity ($\rho = \frac{1}{3}$) but they are not isomorphic, while networks G and G" have the same in and out-degree distributions but they are also not isomorphic. Therefore, reciprocity can not separate G from G' and the degree distribution can not separate G from G". Undirected graphlets can not separate G from G' but they can separate G from G" since their undirected topology is not the same. On the other hand, directed graphlets are capable of distinguishing all three cases since they are more general than a) undirected graphlets, b) reciprocity, and c) in- and out-degree distributions. In fact, directed graphlets combine all three measures, providing a powerful way to analyze directed networks' topology and overcoming the limitations of undirected graphlets for such networks.

3.1.6 Performance on real biological networks

3.1.6.1 Real-world directed networks

We apply GT-Scanner to biological networks because graphlets are often used in computational biology, thus demonstrating a real practical use of our method since practitioners might want methods that capture more information from their data. Nevertheless, our method is general and thus applicable to any directed network.

There are numerous kinds of intra-cellular networks, such as metabolic, transcriptional regulatory, and cell signaling networks, where edge direction is intrinsically related to its function. Metabolic networks represent the set of biochemical reactions occurring within a cell that allow the organism to grow, reproduce, respond to the environment, and other biological functions essential for the organism's survival. These reactions are catalyzed by enzymes that act upon substrates. Therefore, in metabolic networks a node can be an enzyme or a substrate and the connections are directed edges going from enzymes to substrates. Transcriptional regulatory networks model the process by which the information in the genes is transcribed into proteins or RNA, also called gene expression. In these networks nodes are either transcription factors or proteins that are connected by directed edges representing how the transcription factors influence the gene by stimulating or repressing its expression. A cellular signaling network is comprised of a sequence of biochemical reactions between cells of the same organism. A great number of tasks such as the development, repair and immunity of cells depend on the proper functioning of cell signaling networks. Nodes in these networks are proteins and edges exist between activator and receptor proteins that communicate through signals from the first to the latter.

3.1.6.2 Classification accuracy

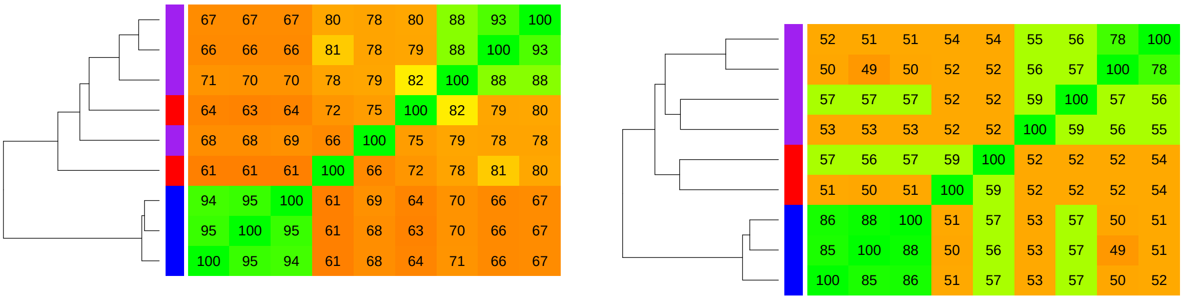
The computational networks used in these experiments, detailed in Table 3.3, are evidently a translation of real biological networks, thus potentially making the process of

G	SIG_NCI	SIG_NH	SIG_SH	SIG_SM	MET_BS	MET_DR	MET_TY	TR_EC	TR_YST
$\|V(G)\|$	15,533	1,634	529	477	453	2,280	2,361	99	688
$\|E(G)\|$	23,682	4,665	1,223	1,056	2,025	5,588	5,822	212	1,078
Type	Signaling				Metabolic			Transcriptional	
Source	[SAK+09]	[CMJ+07]	[MJW+09]		[JTA+00]			[MA03]	[MSOI+02]

Table 3.3: Set of biological networks used for experimental evaluation: cell signaling, metabolic and transcriptional regulatory networks.

finding their similarities harder since their real structure may not be fully represented. Nevertheless, as stated in the introduction to this chapter, it is usually assumed that network structural similarity of computational networks may also indicate functional similarity in the systems. Thus, one can expect that networks belonging to the same type to be more topologically similar than networks of different types.

In order to verify if that is the case for these specific networks we assess their topological similarity in terms of their relative graphlet distributions. We perform this comparison for each pair of networks (G, H) from Table 3.3 by first enumerating all graphlet orbits of both G and H and then comparing their GDDs by computing the $dGDA_4(G, H)$ and $uGDA_5(G, H)$. We show results for (a) $d\mathcal{G}_4$, since 4-node directed graphlets were already successful in correctly clustering the networks by type, removing the need to look for larger and more computationally expensive graphlets, and (b) $u\mathcal{G}_5$, since they are the set of (undirected) graphlets most often used in the literature.



(a) $uGDA_5$

(b) $dGDA_4$

Figure 3.9: Heatmaps and dendrograms of the $uGDA_5$ (a) and $dGDA_4$ (b) obtained for the tested networks. Undirected graphlets accurately clustered the metabolic networks (blue) but incorrectly grouped cell signaling (purple) with transcriptional regulatory networks (red). Directed graphlets were able to cluster all networks by type without error.

We study the network similarity matrices for $uGDA_5(G, H)$ and $dGDA_4(G, H)$ along with their corresponding dendrogram and heatmap (Figure 3.9). We observe that networks of the same type are correctly grouped by their GDA using directed graphlets.

This is an indicator that directed graphlets can detect topological similarities between real directed biological networks of the same type and can likewise find structural differences between networks of different types. We also find that undirected graphlets do not correctly separate cell signaling from transcriptional regulatory networks. Again, since the networks come from distinct sources and noise is embedded in the data, it is not guaranteed that these networks are actually being separated by function and not by bias in their representation. Nevertheless, graphlets are a powerful tool to assess functional similarity and directed graphlets, by definition, capture more functional similarity in directed networks than undirected ones.

3.1.6.3 Speed comparison

Taking edge direction into account augments the complexity of the already computationally demanding subgraph counting process (Section 2.2). Furthermore, it might be interesting to count k-graphlets (both directed and undirected) that have more than the typical five nodes, but execution time grows exponentially with k. Therefore, a very efficient tool is required for this task to be feasible.

We compare our tool, GT-Scanner, with other subgraph counting tools.

Our experimental results were gathered on a 8-core machine consisting of two quad-core Intel Xeon Processor E5620 processors at 2.4GHz with a total of 12GB of memory. We developed GT-Scanner in C++11 and compiled it using gcc 4.8.2. The tools we compare GT-Scanner against were also developed in C++ and are available as open-source: GraphCrunch[KSHP14], ORCA [HD15], Kavosh [Moh14] and ESU [WR06b]. The execution times of each tool are relative only to the graphlet enumeration phase, not taking into account the time taken to load the graph into memory nor perform other initialization or finalization tasks.

GraphCrunch is one of the most popular graphlet discovery tools [MLP08]. By observing its source code one notices that the possible 30 subgraphs are enumerated manually, therefore it is not possible to enumerate a different set of graphlets. Until recently it performed a full enumeration of all graphlets of up to size 5 in order to calculate their orbit frequency. A more recent tool, ORCA [HD14], was shown to perform one or two orders of magnitude faster than the original GraphCrunch and has since been integrated into it. Henceforth, when GraphCrunch is referenced we are alluding to its version before adopting ORCA's algorithm to perform the enumeration. ORCA achieves its performance by observing that, given a limited set of graphs of size

k, it is possible to build a system of equations to calculate their frequencies by using the frequencies of the size $k-1$ graphs and the frequency of a single graph of size k. This greatly reduces execution time (more details in Section 2.2.2). Similarly to GraphCrunch, ORCA manually counts each subgraph, thus it does not allow for different sets of subgraphs to be enumerated. These two tools do not support edge direction, being only applicable for undirected graphlets.

Numerous tools have been proposed for network motif discovery. In our tests, these tools only perform the census on the original network (and not on the randomized networks) because we are only concerned with the subgraph enumeration itself and not with assessing motif significance. In this setting, tools for network motif discovery perform less computational work than graphlet tools since they do not calculate orbit frequencies specific to each node. Kavosh [KAE+09] and Fanmod [WR06a] are two well-known motif tools, the latter being an implementation of the ESU algorithm (more details in Section 2.2.1); in our work we use a more efficient implementation of ESU than Fanmod [Rib11].

Because i) GraphCrunch, ii) ORCA and iii) ESU/Kavosh methodologies are not directly comparable, we create three distinct versions of GT-Scanner and use them accordingly: i) a version that enumerates all graphlets and orbits of up to size k, ii) a version that enumerates up to size $k-1$ graphlets and orbits and then computes a set of equations to calculate the frequencies of the size k graphlets and iii) a version that only enumerates the subgraphs (and not the orbits) of size k.

	#Occurrences (millions)				#Subgraphs			
	$u\mathcal{G}_5$	$u\mathcal{G}_6$	$d\mathcal{G}_4$	$d\mathcal{G}_5$	$u\mathcal{G}_5$	$u\mathcal{G}_6$	$d\mathcal{G}_4$	$d\mathcal{G}_5$
SIG_NCI	1754	7883	24	1754	17	74	89	842
SIG_NG	4	1078	2	4	21	112	191	5219
SIG_SH	3	41	0.2	3	20	103	79	789
SIG_SM	2	27	0.1	2	20	102	70	677
MET_BS	1455	1510	17	1455	5	15	36	371
MET_DR	1782	2799	20	1782	5	16	37	271
MET_TY	1953	960	21	1953	5	16	37	217
TR_EC	0.01	0.04	0.002	0.01	21	98	24	217
TR_YST	3	32	0.3	3	20	81	34	174

Table 3.4: Number of subgraph occurrences and different subgraphs found in each directed biological network.

From Table 3.4 we observed that, even though the networks from Table 3.3 are relatively small, sometimes more than a 1 billion occurrences are found, showcasing the computational complexity of subgraph counting and the necessity of an efficient tool. The number of occurrences of $u\mathcal{G}_5$ and $d\mathcal{G}_5$ for the same networks is necessarily the same, however there will probably be many more different directed than undirected graphlets types (e.g., only 17 different $u\mathcal{G}_5$ appear in SIC_NCI, while 842 different $d\mathcal{G}_5$ appear in it). This highlights the gain in topological information brought by using directed graphlets that is disregarded by undirected graphlets. For instance, directed graphlets $\{G_2$–G_5,G_8–$G_{10}\}$ (Figure 3.3), correspond to the same undirected graphlet G_1 (Figure 2.5) when edge direction is ignored.

G	$\mathcal{G}$	GT-Scanner	GraphCrunch	GT-Scanner	ORCA	GT-Scanner	Kavosh	ESU
SIG_NCI	$u\mathcal{G}_5$	60.73	481.45	3.07	3.08	34.08	3,524.57	2,894.99
	$u\mathcal{G}_6$	4,233.55	n/a	n/a	n/a	2,051.04	> 1 day	> 1 day
	$d\mathcal{G}_4$	1.70	n/a	n/a	n/a	1.13	27.20	26.16
	$d\mathcal{G}_5$	112.09	n/a	n/a	n/a	70.26	3,669.29	3,103.47
SIC_NH	$u\mathcal{G}_5$	1.70	6.08	0.23	0.26	0.96	56.51	45.86
	$u\mathcal{G}_6$	44.29	n/a	n/a	n/a	24.28	1,887.67	1,557.45
	$d\mathcal{G}_4$	0.13	n/a	n/a	n/a	0.10	1.57	1.40
	$d\mathcal{G}_5$	3.71	n/a	n/a	n/a	2.61	56.16	53.79
SIG_SH	$u\mathcal{G}_5$	0.11	0.45	0.02	0.08	0.06	3.74	3.06
	$u\mathcal{G}_6$	2.94	n/a	n/a	n/a	1.10	76.13	62.38
	$d\mathcal{G}_4$	0.01	n/a	n/a	n/a	0.01	0.18	0.16
	$d\mathcal{G}_5$	0.22	n/a	n/a	n/a	0.17	4.18	3.78
SIG_SM	$u\mathcal{G}_5$	0.08	0.34	0.02	0.07	0.05	2.66	2.15
	$u\mathcal{G}_6$	1.30	n/a	n/a	n/a	0.72	50.00	40.38
	$d\mathcal{G}_4$	0.01	n/a	n/a	n/a	0.01	0.14	0.12
	$d\mathcal{G}_5$	0.15	n/a	n/a	n/a	0.12	3.02	2.66
MET_BS	$u\mathcal{G}_5$	42.2	409.04	1.69	1.83	22.11	2,500.87	2,436.82
	$u\mathcal{G}_6$	3,592.02	n/a	n/a	n/a	1,695.38	> 1 day	> 1 day
	$d\mathcal{G}_4$	1.03	n/a	n/a	n/a	0.75	17.43	15.66
	$d\mathcal{G}_5$	96.99	n/a	n/a	n/a	62.93	2,555.38	2,334.01
MET_DR	$u\mathcal{G}_5$	51.12	504.06	1.93	2.06	27.79	3,725.94	3,002.90
	$u\mathcal{G}_6$	4,768.99	n/a	n/a	n/a	2,317.72	> 1 day	> 1 day
	$d\mathcal{G}_4$	1.18	n/a	n/a	n/a	0.86	19.89	18.81
	$d\mathcal{G}_5$	117.64	n/a	n/a	n/a	79.9	3,076.25	2,852.23
MET_TY	$u\mathcal{G}_5$	56.9	551.73	2.09	2.21	30.47	4,113.44	3,506.87
	$u\mathcal{G}_6$	5,177.37	n/a	n/a	n/a	2,473.07	> 1 day	> 1 day
	$d\mathcal{G}_4$	1.26	n/a	n/a	n/a	0.93	22.06	19.45
	$d\mathcal{G}_5$	132.26	n/a	n/a	n/a	82.26	3,352.5	3,141.3
TR_EC	$u\mathcal{G}_5$	< 0.01	0.03	< 0.01	< 0.01	< 0.01	0.02	0.01
	$u\mathcal{G}_6$	0.01	n/a	n/a	n/a	0.01	0.09	0.06
	$d\mathcal{G}_4$	< 0.01	n/a	n/a	n/a	< 0.01	< 0.01	< 0.01
	$d\mathcal{G}_5$	< 0.01	n/a	n/a	n/a	< 0.01	0.02	0.01
TR_YST	$u\mathcal{G}_5$	0.09	0.73	0.02	0.07	0.05	5.00	4.04
	$u\mathcal{G}_6$	1.39	n/a	n/a	n/a	0.71	90.28	74.05
	$d\mathcal{G}_4$	0.01	n/a	n/a	n/a	0.01	0.23	0.21
	$d\mathcal{G}_5$	0.18	n/a	n/a	n/a	0.14	5.13	4.74

(a) Input (b) $\{2, ..., k\}$-graphlets (c) $\{2, ..., k-1\}$-graphlets $+\ k-$equations (d) k-subgraphs

Table 3.5: Execution time (in seconds) of algorithms for subgraph counting (a) over nine directed biological networks and four graphlet sets, both directed and undirected. We compare the three versions of GT-Scanner with algorithms that perform the same tasks: (b) $\{2, ..., k\}$-graphlet enumeration, (c) $\{2, ..., k-1\}$-graphlet enumeration followed by solving a system of equations to obtain the frequencies of the k-size graphlets and (d) k-subgraph enumeration without computing orbits.

Tables 3.5 (b), (c) and (d) show the results for each different version of GT-Scanner and competing tools. Results showing the average (mean) speedup obtained in all nine networks for different sets of graphlets, $u\mathcal{G}_5$, $u\mathcal{G}_6$, $d\mathcal{G}_4$ and $d\mathcal{G}_5$, are presented in Table 3.6. Neither GraphCrunch nor ORCA are capable of enumerating either directed graphlets or undirected graphlets with more than 5 nodes. GT-Scanner counts subgraphs faster than the two aforementioned tools and, in addition to that, can enumerate directed graphlets as well as undirected graphlets with more than 5 nodes. Kavosh and ESU can also perform the census for the cases presented here but they are much slower than GT-Scanner. For instance, for some networks GT-Scanner takes little over a minute to enumerate $u\mathcal{G}_6$ while both Kavosh and ESU need more than a day to complete the same task. On average, GT-Scanner is almost 100 times faster for undirected graphs and about 20 times faster for directed graphs than tools that perform simple subgraph enumeration (Kavosh and ESU). The speedups are lower for directed graphs because the search space is harder to constrain due to the higher number of graphlets sharing less common subtopologies between them. It is noticeable that the speedups of GT-Scanner relative to motif tools are much higher than those relative to graphlet tools. This is due to tools for motif discovery being general, since they can be used to count any set of subgraphs of a given size, while tools to find graphlets can only enumerate undirected graphlets of up to 5 nodes. This allows graphlet tools to have specialized optimizations that motif tools can not match. We also observe that $u\mathcal{G}_5$ takes significantly longer to enumerate than $d\mathcal{G}_4$ since increasing the size of the graphlets greatly increases the computational time. Using GT-Scanner, the time necessary to compute directed graphlets and undirected graphlets of the same size is not substantially different.

$\mathcal{G}$	$\lvert\mathcal{G}\rvert$	$\lvert\mathcal{O}\rvert$	GraphCrunch	ORCA	Kavosh	ESU
$u\mathcal{G}_5$	30	73	7.15 ± 2.56	2.04 ± 1.27	95.00 ± 30.97	80.11 ± 27.85
$u\mathcal{G}_6$	142	480	n/a	n/a	85.89 ± 24.07	70.31 ± 19.88
$d\mathcal{G}_4$	214	730	n/a	n/a	20.61 ± 3.80	18.73 ± 3.86
$d\mathcal{G}_5$	9,578	45,637	n/a	n/a	35.00 ± 9.77	31.75 ± 8.30
(a)			(b)	(c)	(d)	

Table 3.6: Performance comparison of GT-Scanner against other algorithms. *(a)* shows a description of the set of subgraphs being enumerated, as well as the total number of graphlets ($\lvert\mathcal{G}\rvert$) and orbits ($\lvert\mathcal{O}\rvert$). (b), (c) and (d) show average speedups of GT-Scanner against other algorithms over all networks.

From these experiments we can conclude that GT-Scanner can both perform faster than state-of-the-art graphlet tools and also provide a more general approach which supports any directed or undirected subgraph size, as long as the set of subgraphs fits into memory.

3.1.7 Summary

In this chapter we highlighted the importance of extending graphlets to support edge direction with the task of network classification in mind.

We have presented an efficient tool, GT-Scanner, that is able to compute directed and undirected graphlets of arbitrary size, as well as network motifs, as long as they fit into memory. GT-Scanner also allows the user to customize the set of graphs to be counted, further demonstrating the flexibility of our tool. We assess GT-Scanner's performance on a set of synthetic network models and on real-world directed biological networks.

On synthetic networks, where the ground-truth is well-defined, we observe that directed graphlets classify networks more accurately than undirected graphlets, with a gain of $\approx 95\%$ in accuracy (AUPR).

On real-world networks, where new insight can be gained, we observe that GT-Scanner is the fastest available tool for subgraph counting, despite being a general approach that does not target any specific subgraphs.

Therefore, we believe that we have broadened the applicability of graphlets by extending them to directed graphlets and by providing an efficient tool and methodology.

3.2 Network classification of temporal networks

3.2.1 Motivation

Networks are widely used to model real-world systems as a way to uncover their topological features [COJT^{+}11]. Most of these systems are not static; they exhibit a dynamic nature that can only be captured and truly understood by taking into account the network's temporal evolution [HS12]. Consider for instance a co-authorship network, where nodes are authors and edges represent joint publications. By narrowing our focus to static network snapshots we cannot answer relevant questions such as: how stable are connections over time? How is collaboration emerging and dissolving? How did we get to the current state of the network? Can we predict how the network will look like in the future?

Shah et al. [SKZ$^+$15] propose an algorithm that concisely summarizes temporal networks by their characteristic temporal subnetworks. Similarly to their work, we also aim for interpretability, but we do graph comparison instead of graph summarization and our method does not require a null model to assess how a certain interesting pattern deviates from randomness. Yu et al. [YAW17] put forward a matrix factorization method that characterizes the correlations of network's edges as a function of time. Their representation builds a dynamic profile of the network that can be used to predict future states. Here we do not specifically target link prediction; our graphlet-orbit transitions could possibly be used for the task but that is out of the scope of this work. Another task related with both network comparison and network visualization is network condensation [AZBP17]; its aim is to reduce the size of the temporal network significantly without much loss of information. Here we aim for interpretability but we do not address the problem of network condensation directly.

Subgraph-based measures, such as network motifs and static graphlets, have been successfully used to compare static networks [MIK$^+$04, Prž07a, CRBS12, ARS17]. However, measures such as these disregard temporal information which can be crucial for a better understanding of network topology and function. Two extensions of graphlets for dynamic networks have been proposed (described below). For an overview of subgraph counting in temporal network we refer the reader to Section 2.2.6.

Faisal and Milenkovic [FM14] integrate graphlet frequency distributions on the analysis of temporal biological networks, but they only look at the global distribution in each snapshot, without offering the possibility to observe how each individual connected set of nodes is evolving. By contrast, we provide a direct transition matrix.

The work by Hulovatyy et al. [HCM15] provides an extension of graphlets to temporal networks, called dynamic graphlets. However, dynamic graphlets, and the respective dynamic graphlet degree vectors (DGDVs), only allow for a single event (i.e., temporal edge) at each snapshot (i.e. just one edge addition between two nodes), therefore limiting the scope of possible graphlet transitions. Our method differs from these because our transition matrix establishes direct relations between snapshots, and we allow for any number of edge additions or removals in each snapshot, aiming for a broader and fully general set of possible transitions between two consecutive snapshots.

3.2.2 Overview of our contribution

In this work we propose graphlet-orbit transitions (GoTs) as features for characterizing and comparing temporal networks. Note that we use the terms "orbit" and "graphlet-orbit" interchangeably. Our method incorporates the rich topological information provided by graphlets and extends them to the temporal domain. Orbit-transition matrices encapsulate not only how graphlets change but also how the roles of the nodes themselves change, leading to a more detailed fingerprint of the network. We also introduce the Orbit-Transition-Agreement (OTA) as a suitable measure to compare transition matrices of heterogeneous networks.

Next we underline our main contributions:

- **Effectiveness:** GoTs achieves over 30% higher precision (AUPR) on a set of well-known network models than other subgraph-based methods. On real data it classifies networks more accurately than competing approaches.
- **Interpretability:** Results produced by GoTs are very easy to visualize (i.e., analyze specific transition frequencies between orbits). Therefore, GoTs can be used as interpretable fingerprints of temporal networks.
- **Generability:** Our method is used to compare heterogeneous networks from different domains and of different sizes. Furthermore, GoTs is general and easily extensible to directed and multilayered networks, but these extensions are demanding in terms of storage and execution time.

3.2.3 Graphlet-Orbit Transitions (GoTs)

In this section we describe our method and specify how it is used to measure temporal network similarity. Temporal network similarity assumes that there is a set of temporal patterns used as features; in our case those features are graphlet-orbit transitions (GoTs). A measure of similarity is also necessary to compare the networks' feature space; for this purpose we develop orbit-transition agreement (OTA). Thus, from Section 2.3, $\mathcal{F}_i$ are GoTs and ϕ_i is OTA(GoTs(G), GoTs(H)).

Essentially, GoTs are obtained by performing subgraph counting for each snapshot of a given temporal network, also registering the orbits of each node in the subgraph occurrence, and computing how those orbits evolved over time. Our aim is to analyze how the roles of nodes are evolving between snapshots.

Only connected graphlets are taken into account because our focus is to study how groups evolve and, when a group becomes disconnected, that set of nodes is no longer a group. We should point out that disconnected graphlets would be very useful to analyze group formation, but computing their frequencies would require considering all possible $\binom{n}{k}$ subsets of nodes, effectively making it only feasible for small networks and very small k-graphlets.

Possible algorithms to count subgraphs and orbits are discussed in Section 2.2. Note that, since we need to list subgraph occurrences and not only count them, our method performs subgraph enumeration. Here we use graphlet-tries (Section 3.1.4) due to their general applicability and efficiency [RS14c, ARS17].

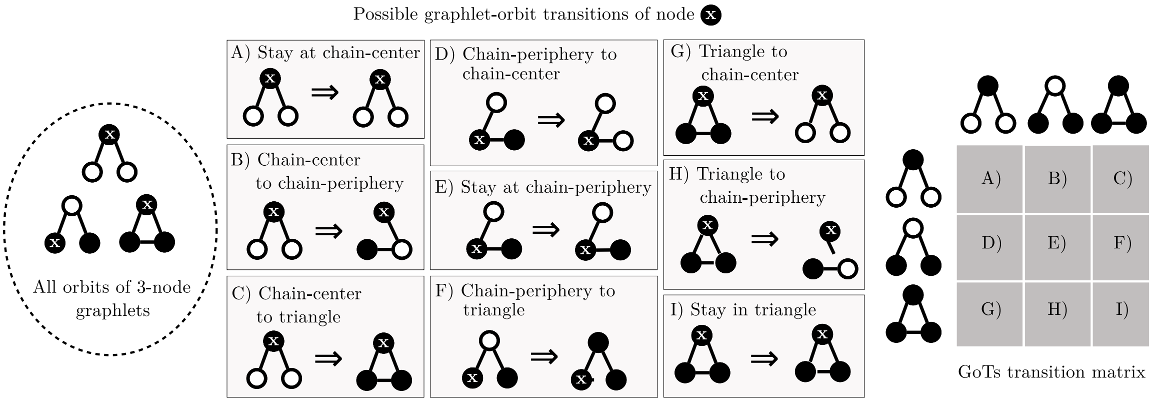

Possible graphlet-orbit transitions of node x
All orbits of 3-node graphlets
A) Stay at chain-center
B) Chain-center to chain-periphery
C) Chain-center to triangle
D) Chain-periphery to chain-center
E) Stay at chain-periphery
F) Chain-periphery to triangle
G) Triangle to chain-center
H) Triangle to chain-periphery
I) Stay in triangle
A) B) C)
D) E) F)
G) H) I)
GoTs transition matrix

Figure 3.10: All possible orbit transitions of 3-node undirected graphlets and corresponding orbit-transition matrix. Node x is the node being currently considered and black nodes are nodes in the same orbit as x.

As an example of GoTs, consider the two 3-node undirected graphlets, $u\mathcal{G}_3$, and their respective three orbits, $u\mathcal{O}_3$, from Figure 3.10. The chain-graph has two possible orbits (i.e., the node can be either at its center or in one of its two leaves) while all nodes in a triangle-graph are topologically equivalent. Given those three orbits, the GoTs are the possible transitions of node x from one orbit to another. There are $3 \times 3 = 9$ possible orbit transitions for $u\mathcal{O}_3$: x can remain in its previous orbit, be it a (A) chain-center, (E) chain-periphery or (I) triangle-node; x can transition from the chain-center to the chain-periphery (B) or to a triangle-node (C), etc., all possibilities are shown in the GoTs transition matrix of Figure 3.10.

Given a set of GoTs, we want to compute the frequency of each transition for each node in the network, i.e., the node's GoTs frequency matrix. Figure 3.11 shows an

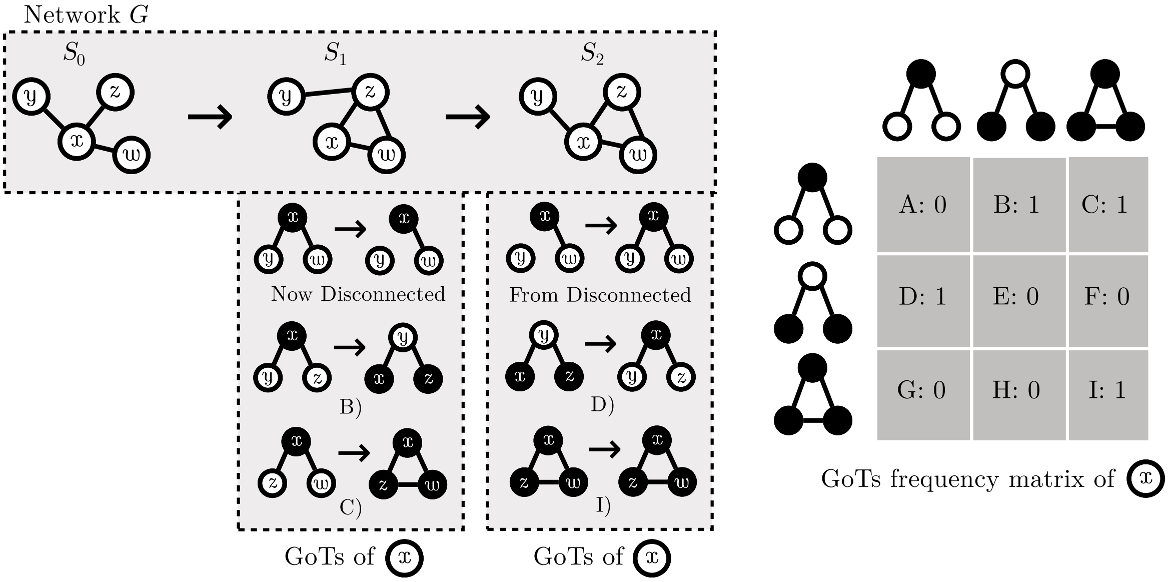


Figure 3.11: Graphlet-orbit transitions of node x. Note that transitions to (and from) disconnected graphlets are not considered.

example of a temporal network and the GoTs frequency matrix of a single node x. The network's GoTs frequency matrix is the sum of its nodes' GoTs frequency matrices.

Algorithm 3.3 gives an overview of the enumeration process that builds the GoTs frequency matrix $F_{GoT}(G) = |\mathcal{O}_k| \times |\mathcal{O}_k|$. Our method enumerates, for each snapshot $S_t(G) \in \mathcal{S}(G)$, all k-node subgraph occurrences S_G, where $\mathcal{V}(S_G) = \{n_1, n_2, ..., n_k\} \subseteq \mathcal{V}(G)$ (lines 3-5), as well as the orbits of each node on that subgraph $\mathcal{O}(S_G) = \{o_1, o_2, ..., o_k\}$ (line 6) – i.e., in snapshot $S_t(G)$, node n_1 from network G appears in subgraph S_G in orbit o_1, etc.. Our algorithm pushes each occurrence into a vector $Fr(G)$ of the form $\{n_1, n_2, ..., n_k, t, o_1, o_2, ..., o_k\}$ (line 7). When all occurrences have been enumerated in every $S_t(G)$, we sort $Fr(G)$ (line 9) and check if two consecutive vector positions contain the same subgraph S_G (i.e., S_G was found in consecutive snapshots) (lines 10-11). As an example, occurrences $\{5, 8, 10, 12, t = 1, x, x, y, y\}$ and $\{5, 8, 10, 12, t = 2, x, y, x, y\}$ increment $F_{GoT}(G)[x, x]$, $F_{GoT}(G)[x, y]$, $F_{GoT}(G)[y, x]$ and $F_{GoT}(G)[y, y]$ all by 1 (lines 12-14).

GoTs frequency matrices offer rich topological information that can be used for network summarization, Data Mining (e.g. they can be used as features for prediction tasks), network comparison and model fitting. In this chapter we compare and classify different networks according to GoTs frequency matrices.

Algorithm 3.3 Enumerate GoTs of orbits $\mathcal{O}_k$ on temporal network G.

```
procedure ENUMERATEORBITTRANSITIONS($G$, $\mathcal{O}_k$)
    $Fr = \varnothing$
    for all snapshots $S_t(G) \in \mathcal{S}(G)$ do
        while ENUMERATESUBGRAPH($S_t(G), \mathcal{O}_k$) finds an occurrence $S_G$ do
            $\mathcal{V}(S_G) = \{n_1, n_2, ..., n_k\}$
            $\mathcal{O}(S_G) = \{o_1, o_2, ..., o_k\} = getOrbits(S_G)$
            $Fr.append(\{n_1, n_2, ..., n_k, t, o_1, o_2, ..., o_k\})$
    $F_{GoT}(G)$ : fill with $|\mathcal{O}_k| \times |\mathcal{O}_k|$ zeros
    SORT($Fr$)
    for all $i < |Fr|; Occ_1 = Fr[i], Occ_2 = Fr[i+1]$ do
        if $\mathcal{V}(Occ_1) == \mathcal{V}(Occ_2)$ and $t(Occ_1) == t(Occ_2) - 1$ then
            for $j \in \{1, ..., k\}$ do
                $(o_a, o_b) : (\mathcal{O}(Occ_1)[j], \mathcal{O}(Occ_2)[j])$
                $F_{GoT}(G)[o_a, o_b] + +$
    return $F_{GoT}(G)$
```

3.2.4 Orbit-transition Agreement (OTA)

Next we describe OTA, our measure of network similarity based on the networks' GoTs frequency matrices.

First, we normalize $F_{GoT}(G)$ for each $G \in \mathcal{N}$ in order to reduce bias induced by different network sizes. Normalization is performed by row, as shown in Equation 3.5. This normalization gives the same importance to common and rare orbits. Normalizing $F_{GoT}(G)$ both by row and column could be performed instead, if the scale of the original values was important. However, the second normalization proposed would give high importance to common orbits and low importance to rare orbits. To us, it seems more reasonable to expect that rare orbits are equally important to distinguish networks, thus we chose normalization by row only. Recall that $F_{GoT}(G)[o_a, o_b]$ is the number of times that any node x from G transitioned from orbit o_a to orbit o_b in subsequent snapshots.

$$nF_{GoT}(G)[o_a, o_b] = \frac{F_{GoT}(G)[o_a, o_b]}{\sum\limits_{o_c}^{|\mathcal{O}|} F_{GoT}(G)[o_a, o_c]} \tag{3.5}$$

The OTA similarity of two networks G and H is given by the average similarity of their normalized GoTs frequency matrices NF_{GoT} (Equation 3.6).

$$OTA(G,H) = \frac{1}{|\mathcal{O}|^2} \times \sum_{o_a}^{|\mathcal{O}|} \sum_{o_b}^{|\mathcal{O}|} \Big(1 - |nF_{GoT}(G)[o_a, o_b] - nF_{GoT}(H)[o_a, o_b]|\Big) \tag{3.6}$$

Equation 3.6 produces an *absolute value of agreement*, i.e., $OTA(G,H)$ is always the same regardless of $\mathcal{N}$. However, for our purposes a *relative value of agreement* is more suitable since we want to compare networks within a specific set. Consider $max(OTA_{\mathcal{N}})$ and $min(OTA_{\mathcal{N}})$ as the highest and lowest OTA between any two networks in set $\mathcal{N}$: we normalize the OTA matrix to values between 0 and 1 (Equation 3.7). Using the normalized $nOTA$, the two most similar networks on the set $\mathcal{N}$ have $nOTA = 1$, and the two most different have $nOTA = 0$, while the other pairs have a normalized $nOTA$ between 0 and 1. Henceforth, when we mention OTA we are referring to $nOTA$.

$$nOTA(G,H) = \frac{OTA(G,H) - min(OTA_{\mathcal{N}})}{max(OTA_{\mathcal{N}}) - min(OTA_{\mathcal{N}})} \tag{3.7}$$

Algorithm 3.4 shows our overall methodology.

Algorithm 3.4 Compute network similarity of set $\mathcal{N}$ using k-node GoTs

```
procedure COMPUTENETWORKSIMILARITY($\mathcal{N}$, $k$)
    $\mathcal{O}_k$ : generate all $k$-node orbits.
    for all networks $N \in \mathcal{N}$ do
        $F_{GoT}(G)$ = ENUMERATEORBITTRANSITIONS($G, \mathcal{O}_k$)
        $nF_{GoT}(G)$ = NORMALIZE($F_{GoT}(G)$)                    (Eq. 3.5)
    for all pairs $\{(G,H) \mid G,H \in \mathcal{N}\}$ do
        $OTA(G,H)$ = GETOTA($F_{GoT}(G), F_{GoT}(H)$)              (Eq. 3.6)
    for all pairs $\{(G,H) \mid G,H \in \mathcal{N}\}$ do
        $nOTA(G,H)$ = NORMALIZE($OTA(G,H)$)                      (Eq. 3.7)
```

3.2.5 Classifying synthetic data

We assess GoTs classification efficiency on a set of well-known graph models, and compare it against other subgraph-based methods. Our assumption is that an efficient method should report networks from the same model as more topologically alike than networks from different models due to their inherent structure (a similar approach was followed in [HCM15]).

All of the following experiments were conducted on an Intel i7-6700 CPU with 4 cores at 3.40 GHz; nevertheless, all programs were executed using a single-thread. Our code was written in C++11 and compiled with gcc 6.3.1 with O3 optimizations, while dynamic graphlets [HCM15] were computed using the executable available at `http://www3.nd.edu/~cone/DG/`. Network motifs, graphlets and static-temporal graphlet vectors were obtained using our own code, available at `http://www.dcc.fc.up.pt/~daparicio/software`. The source code for graphlet-orbit transitions computation, as well as the data used for experimental purposes, can be found at `http://www.dcc.fc.up.pt/got-wave/`.

3.2.5.1 Synthetic networks

We developed dynamic versions of three random-graph models: Erdos-Rényi [ER60], Baràbasi-Albert [BA99] and Watts-Strogratz [WS98]. All synthetic networks have 5 snapshots and start with 250 nodes; these values were chosen in order to obtain results from every method in a reasonable time. New nodes and edges arrive in the networks [LBKT08], while the network's density remains stable throughout all snapshots (this behavior was observed in online social networks [HW09], for instance). New edges are created according to the model's criteria: either randomly [ER60], by preferential attachment [BA99], or through rewiring of past edges [WS98]. Noise is also injected in some of the networks by having edges randomly deleted: if $P(e^-) = 0.5$, half of the edges from $S_t(G)$ are removed in $S_{t+1}(G)$, whereas if $P(e^-) = 0$, all edges are permanent. Strogratz models control how much rewiring is performed; we use either no rewiring ($\beta = 0$) to build regular ring-networks or some rewiring ($\beta = 0.2$) to create small-world networks.

From these different models and parameters, we obtain a total of six different network classes (see Table 3.7) and generate 25 networks of each.

Table 3.7: Set of random network models used for evaluation.

	$\|\mathcal{S}(\mathbf{G})\|$	$\|\mathcal{V}(\mathbf{S_1}(\mathbf{G}))\|$	$\|\mathcal{V}(\mathbf{S_{t+1}}(\mathbf{G}))\|$	$\frac{\|\mathcal{E}(\mathbf{G})\|}{\|\mathcal{V}(\mathbf{G})\|^2}$	$\mathbf{P}(\mathbf{e}^-)$	$\mathbf{P}(\mathbf{e}^+)$	$\mathbf{P}(\beta)$
Erdos	5	250	$1.1 \times \|\mathcal{V}(S_t(G))\|$	0.01	0.0 0.5	Random	–
Barábasi	5	250	$1.1 \times \|\mathcal{V}(S_t(G))\|$	0.01	0.0 0.5	Degree	–
Strogratz	5	250	$1.1 \times \|\mathcal{V}(S_t(G))\|$	0.01	–	Ring creation and Rewiring	0.0 0.2

3.2.5.2 Measures

In our experiments we assess GoTs' classification accuracy and compare it against other subgraph-based measures. All methods perform subgraph counting (Section 2.2), thus an appropriate set of k-subgraphs needs to be chosen. Hulovatyy et al. [HCM15] reported that dynamic graphlets with 4-nodes and 6-events achieved the best results for node classification tasks, and that increasing their size did not improve results significantly while greatly increasing computational time. Therefore, 4-node and 6-event dynamic graphlets are enumerated and, for results to be directly comparable, 4-node subgraphs are enumerated for every other method. For generability, all possible 4-node subgraphs are enumerated instead of just a specific set.

- **Static motifs (SM)** – 4-node network motifs (Section 2.2.5.1) are enumerated on a single aggregate network, and their *motif score* is evaluated on a set of 100 similar randomized networks (see [MIK$^+$04]).
- **Static graphlets (SG)** – 4-node graphlet-orbits are enumerated on a single aggregate network (see [Prz07b]).
- **Static-temporal graphlets (STG)** – 4-node graphlet-orbits are enumerate on each network snapshot (see [FM14]).
- **Dynamic graphlets (DG)** – 4-node graphlets with at most 5 events are enumerated on the temporal network (see [HCM15]).
- **Graphlet-orbit transitions (GoTs)** – 4-node graphlet-orbit transitions are enumerated. However, for the methods to be more easily comparable, the OTA is not computed.

3.2.5.3 Accuracy and speed comparison

For each node, we compute its SM, SG, STG, DG and GoTs and use them as the node's features. For instance, when considering GoTs, each node is represented by its graphlet-orbit transitions. The feature vectors over all nodes in a network form a #Nodes × #Features matrix. For two networks being compared, this results in two corresponding matrices with the same number of columns, whose rows are then joined together. Due to high dimensionality and sparsity of the joined matrix, we perform dimensionality reduction on the matrix using principal component analysis, keeping

99% of its variance. Then, we compute the topological similarity between every two networks as the Euclidean similarity between their PCA-reduced matrices.

Precision-recall curves were calculated for each measure and presented in Figure 3.12 (for details on how they are calculated as well as their areas, we refer the reader to Section 2.3).

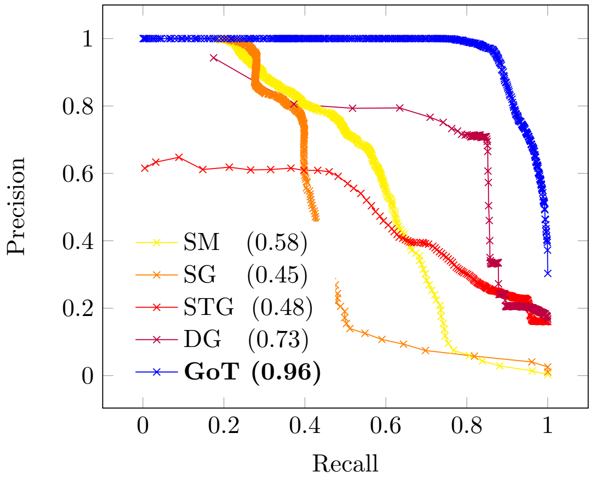


Figure 3.12: Obtained precision-recall curves on synthetic data (AUPR inside parentheses).

Our method (GoT) achieves the highest AUPR and has a gain of ≈ 30% when compared to the second best (DG). STG obtains a higher AUPR than SG, but only by a small fraction, while DG performs significantly better than both, corroborating the results from [HCM15]. Table 3.8 compares the execution times of the two approaches that achieve highest AUPR: GoTs and DG [HCM15]. For Barábasi networks times are comparable; this is due to the high density of Baràbasi networks that induce a larger number of GoTs transitions.

In our experiments it is clear that our method is both faster and more accurate than competing measures on a set of well-known temporal graph models. For the Strogratz model with no rewiring ($P(\beta) = 0.0$), in particular, our method is over 400 times faster than DG [HCM15]. This high efficiency comes from the data-structure and algorithm that we use, based on graphlet-tries [RS14c, ARS17].

3.2.6 Grouping and analyzing real data

In this section we show the effectiveness of our proposed method in (a) grouping a set of real-world temporal networks by predetermined categories and (b) visualizing their

Table 3.8: Time comparison of our method (GoTs) against dynamic graphlets (DG). We show the speed gain of GoTs over DG inside parentheses (e.g., 2x means 2 times faster).

	Erdos: $P(e^-)$		Baràbasi: $P(e^-)$		Strogratz: $P(\beta)$	
	0.0	0.5	0.0	0.5	0.0	0.2
DG	5.92s	17.84s	5.96s	471.68s	52.44s	33.84s
GoTs	0.76s (8x)	0.84s (21x)	4.50s (1.3x)	5.36s (88x)	0.12s (437x)	0.52s (65x)

characteristics. Therefore, our goals are to assess grouping capabilities but also interpretability. The set of real-world networks $\mathcal{N}$ comprises (i) co-authorship, (ii) crime, (iii) e-mail communication, (iv) physical interaction, (v) bipartite, (vi) soccer transfers and (vii) social media friendship networks, as shown on Table 3.9.

We start by analyzing how networks are evolving over time (*growing* vs. *shrinking*, becoming *more-connected* vs. *less-connected*) as well as some of their global metrics, namely the *average-degree*, the *clustering-coefficient* and the *characteristic path-length.* These measures are easy to analyze visually and give some temporal information about the networks, but they are not successful when grouping the networks due to their limitations.

We also conduct static network motif (SM) and graphlet (SG) analyses since they capture richer topological information than aforementioned global metrics. We compare the networks' *motif-fingerprints* and *graphlet-degree distributions* for 4-node subgraphs and assess how well the networks are being grouped using these metrics. We assess the clustering capabilities of static graphlet-orbits by computing the graphlet-degree-agreement (GDA) for each pair of networks and clustering set $\mathcal{N}$ accordingly: networks with high agreement are grouped together.

We proceed in a similar fashion for our own GoTs by computing the orbit-transition-agreement (OTA) for each pair of networks. Finally, we show that graphlet-orbit transition matrices offer highly interpretable information which displays both (a) clear differences between networks of different categories and (b) characteristic transitions in networks of the same category.

Here we do not show results for static temporal graphlets (STG) [FM14] because they did not show significant improvement in our synthetic data (Table 3.7) and they are harder to visualize than static graphlets. Dynamic graphlets (DG) [HCM15] with 4 nodes and 5 or 6 events were computed in our set of networks $\mathcal{N}$ but, for some networks, the method did not output graphlet counts in a manageable time, making it impossible to compare with our method. Table 3.10 shows a comparison of the

Table 3.9: Set of temporal networks $\mathcal{N}$ grouped by category.

Name	$\vert\mathcal{V}(G)\vert$	$\vert\mathcal{E}(G)\vert$	ρ	$\vert\mathcal{S}(G)\vert$	**Source**
`Authenticus`	authors 7k	co-author a paper 120k	1 year	16	Our own.
`arXiv`	authors 2k	co-author a paper 357k	1 year	7	[LKF07]
`Minneapolis`	streets 454	crime in intersection 12k	3 months	16	[Ris18]
`Philadelphia`	streets 1k	crime in intersection 10k	3 months	16	[Chi18]
`Emails`	workers 167	sends email 83k	1 month	9	[MPK11b]
`Enron`	workers 6k	sends email 51k	2 months	16	[LLDM09]
`Gallery`	visitors 420	physical interaction 43k	4 days	16	[ISB+11]
`Conference`	visitors 113	physical interaction 21k	12 hours	6	[ISB+11]
`School`	students 327	physical interaction 189k	1 day	5	[SVB+11]
`Workplace`	workers 92	physical interaction 10k	10 days	10	[GVF+15]
`Escorts`	clients + escorts 10k + 7k	hires 51k	3 months	16	[RLH10]
`Twitter`	users + tags 12k + 16k	tweet in hashtag 327k	3 months	16	[DCLS+10]
`Transfers`	soccer teams 2k	player transfer 20k	1 year	16	Our own.
`Facebook`	friends 47k	post on the wall 877k	3 months	16	[VMCG09]

execution times between our method (GoTs) and dynamic graphlets (DG). All possible 4-node graphlets were enumerated by both methods. Dynamic graphlets have the number of events as an additional parameter; thus, dynamic graphlets with 5 events (DG-5) or 6 events (DG-6) were separately computed. For some of the largest networks from Table 3.9 neither DG-5 nor DG-6 produced an output in a reasonable time (we allowed it to run for over a week). For the networks that both GoTs and DG finished their computation it is clear that DG is much more computationally heavy. Furthermore, growing the number of events from 5 to 6 greatly increased computational time. For these reasons, dynamic graphlets were not included in our discussion of real-world networks.

Table 3.10: Execution times of GoTs with four nodes (GoTs), of DG with four nodes and five events (DG-5), and of DG four nodes and six events (DG-6). An asterisk (*) means that the method did not finish in the maximum running time of 1 week.

	GoTs	**DG-5**	**DG-6**
`Escorts`	8 sec	2 hours	4 hours
`Philadelphia`	0.5 sec	25 hours	*
`Minneapolis`	2 sec	12 hours	*
`Enron`	2 min	1 day	4 days
`Gallery`	24 sec	16 hours	3 days
`Escorts`	8 sec	2 hours	4 hours
`Transfers`	3 sec	40 min	1 hours

3.2.6.1 Network overview

We collect a set of 14 temporal networks $\mathcal{N}$ from various sources (Table 3.9). $\mathcal{N}$ is comprised of active-edge networks, meaning that edges are only present in the snapshot $S_t(G)$ in which they appear at and need to be re-activated in subsequent snapshots. The number of snapshots $|\mathcal{S}(G)|$ depends on the amount of available data of G. Long-term networks, such as co-authorship networks, have a bigger time-interval ρ when compared with short-term networks, such as physical interactions in social events.

Figure 3.13 shows how the networks are evolving size-wise. Most of them are growing as time goes by. The fastest growing networks are `arXiv`, `Twitter`, `Facebook` and `Enron`, which start at only $\approx 10\%$ of their largest state, but `Enron` begins shrinking at $t = 11$ and almost disappears by $t = 16$. `Authenticus`, `Escorts` and `Transfers` are also growing networks but they grow at a slower rate and become almost stagnant at the end, where they might have reached their full potential in terms of growth. Crime, physical interaction networks and `Emails` stay relatively stable in size.

Figure 3.14 presents the evolution of the networks' average degree. `arXiv`, `Emails` and physical interaction networks are the ones with higher average degree. `arXiv`, `Twitter` and `Facebook` are the fastest growing in terms of their average degree and most networks have a stable average degree.

By observing Figure 3.15 one can conclude that all networks from $\mathcal{N}$ are small-world since their characteristic-path-length at latter stages ($t \approx 16$) is between 2 and 7.

No clear correlation linking category with characteristic-path-length evolution, growth or average degree is observed from Figure 3.13, 3.14 and 3.15, respectively. Clustering coefficients were also computed for each network snapshot and it was found that they do not change with t. Co-authorship networks have the highest clustering coefficient at

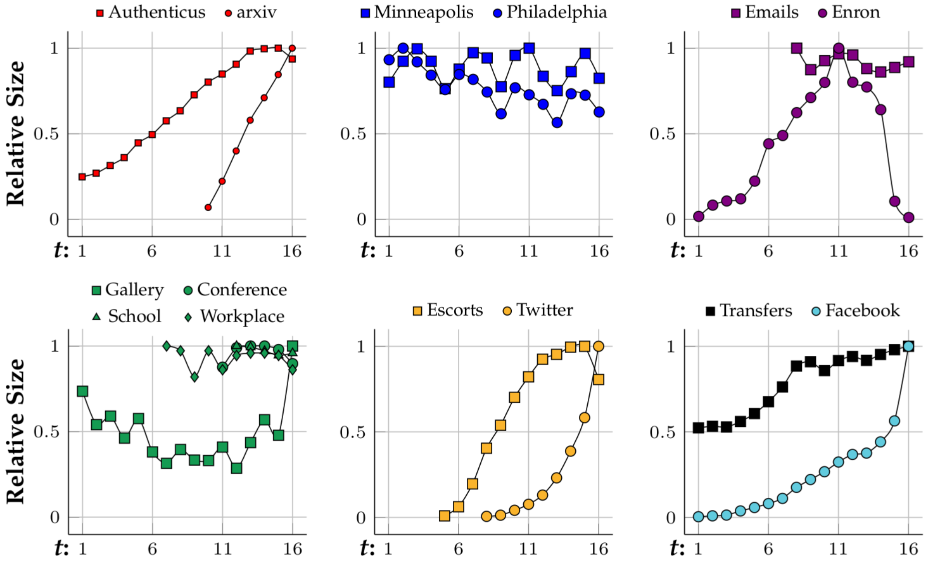


Figure 3.13: Network growth according to its number of nodes – grouped by type.

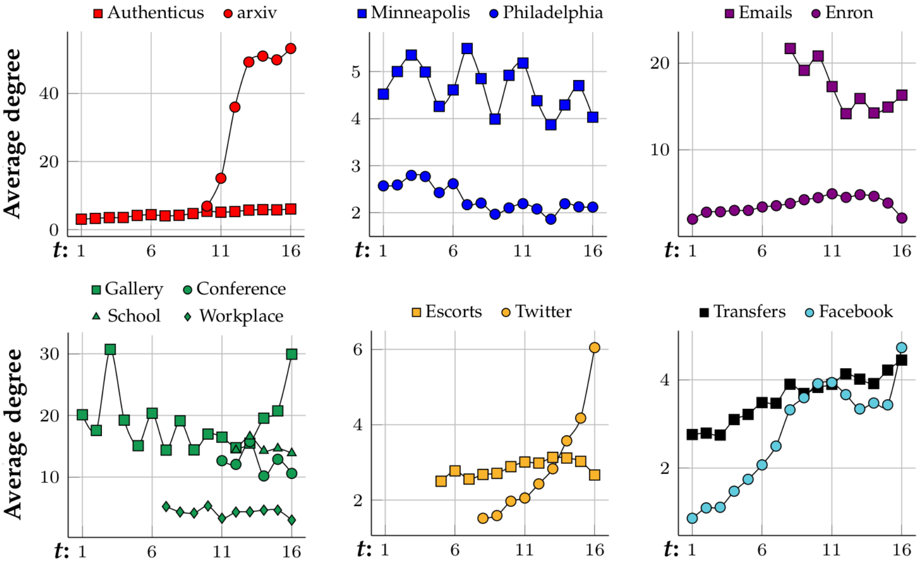


Figure 3.14: Average degree of the networks by time – grouped by type.

0.5 while crime, bipartite, `Facebook` and `Tranfers` networks have near-zero clustering coefficient. The clustering coefficient is capable of grouping co-authorship networks together despite only considering 3-node subgraphs (triangles and 3-node chains). However, it does not distinguish between crime and bipartite networks, for instance. In these cases, one option to differentiate between networks with similar 3-node subgraphs is to analyze their 4-node network motifs and graphlets.

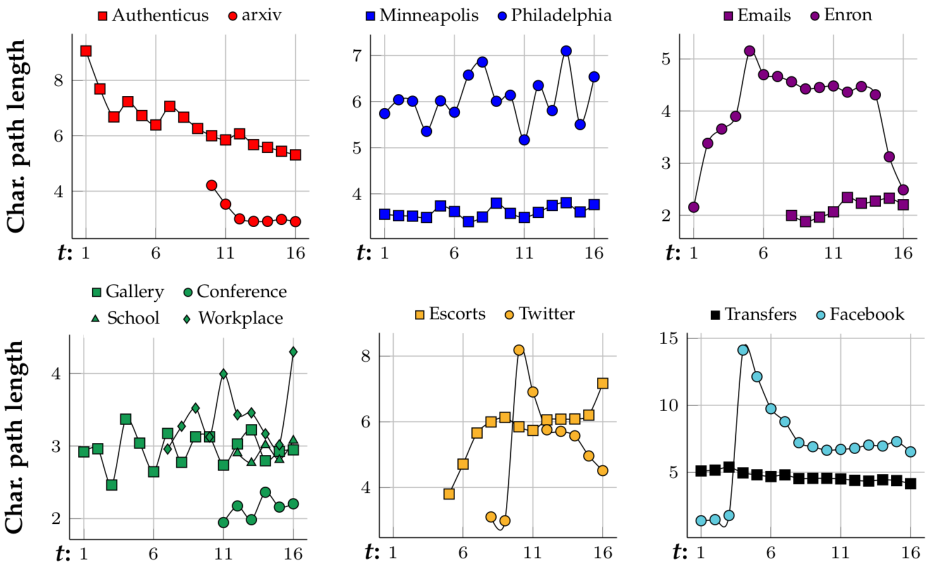


Figure 3.15: Characteristic path length of the networks by time – grouped by type.

3.2.6.2 Network motifs

We enumerate subgraphs with $k = 4$ and $k = 5$ and present results only for the smaller subgraphs since no significant differences were observed between the two sets. We use GT-Scanner (described in Chapter 3.1) for both subgraph counting and to analyze motif significance. We count subgraphs in a single aggregate state of each network G from Table 3.9 and calculate motif scores $\Delta_{(H,G)}$ for each graph $H \in \mathcal{G}_k$ (Equation 2.1 and Equation 2.2). Motif fingerprints between two networks are compared by computing their Euclidean distance.

Figure 3.16 shows the obtained motif-fingerprints for all 4-node undirected subgraphs ($u\mathcal{G}_4$), evaluated against 100 randomized networks. Co-authorship networks have a similar motif-profile where cliques and near-cliques are the most unexpectedly prevalent groups. This comes from the fact that scientific collaboration communities tend to be tightly connected [CRBS12]. The two crime networks have a similar network profile, with cliques and near-cliques being underrepresented while squares (G_3) are very over-represented. This result was expected since our crime networks are geographical graphs with near-zero clustering coefficient and cities have a grid-like structure. Motif-profiles of the email networks are also relatively alike. Similar to co-authorship networks, cliques and near-cliques are the most over-represented subgraphs. However, that is much more obvious in `Enron` than in `Emails`. This is probably because `Emails` is too small for the over-representation to become obvious since the small random networks are also capable of generating cliques and near-cliques. Physical interaction networks have a similar motif-fingerprint but it seems indistinguishable from co-authorship networks. Both types of networks have cliques and near-cliques as the most overrepresented subgraphs but those groups have different meanings. In co-authorship networks they might indicate communities but in the short-term networks they seem to simply indicate that everyone communicates with everyone by the end of the timeframe. Analyzing just the final aggregate network ignores relevant information, it is often more insightful to study how networks evolve. Bipartite networks have similar motif-fingerprints but they are also identical to those of crime networks. It should be pointed out that these networks are not pure bipartite networks but only nearly bipartite, otherwise subgraphs with cycles would never occur (G_3, G_4, G_5 and G_6). The `Transfer` network's motif fingerprint is also similar to the ones of crime and bipartite networks. Finally, `Facebook`'s motif-profile is alike co-authorship network except G_3 is also overrepresented. Since `Facebook`'s density is so low ($\frac{N}{E^2} \approx \frac{183000}{64000^2} \approx 0.004\%$) randomized networks have almost exclusively stars (G_1) and chains (G_2). By observing Figure 3.17 (a) it is clear that motifs can only separate the networks into two big groups.

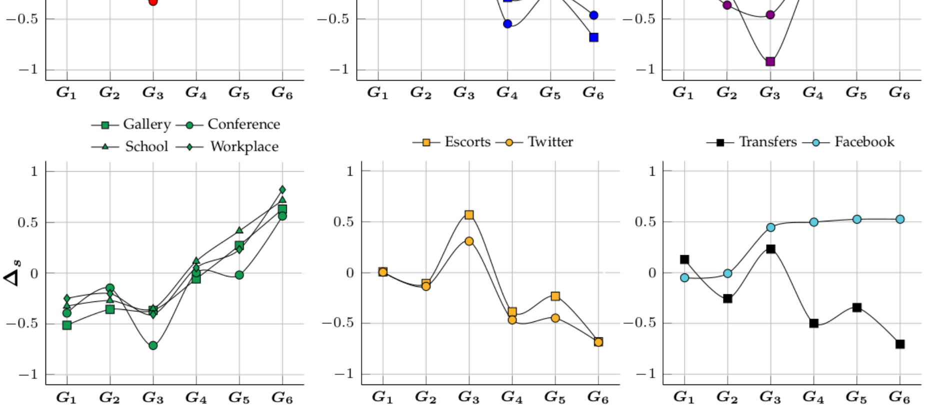
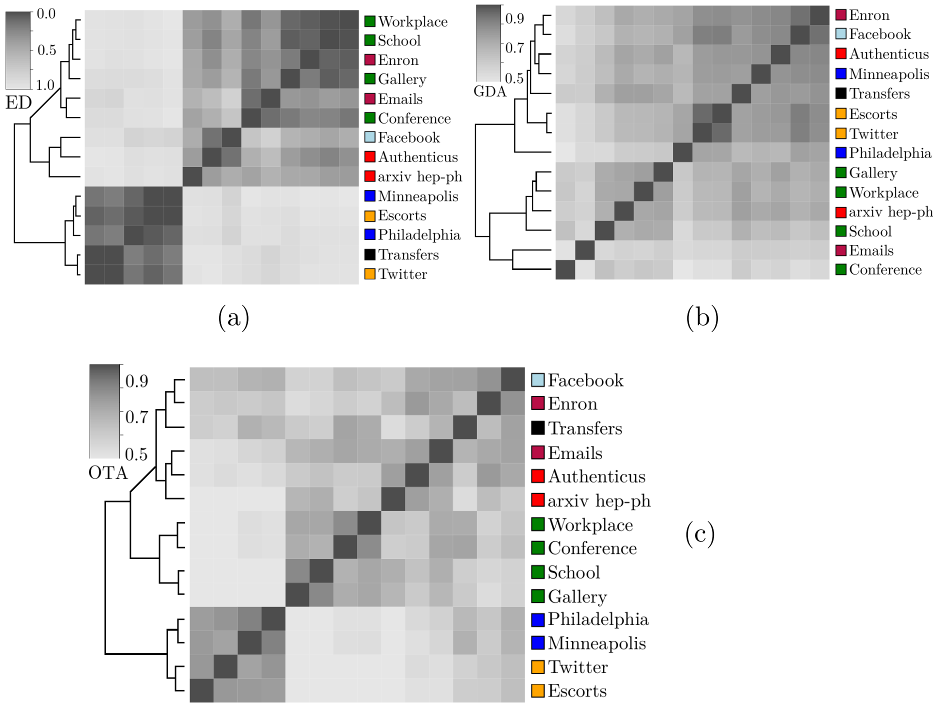


Figure 3.16: Motif-fingerprints of the networks by time – grouped by type



(a)

(b)

(c)

Figure 3.17: Similarity matrices according to (a) motif-fingerprints' Euclidean distance (ED), (b) graphlet-degree-agreement (GDA) and (c) orbit-transition-agreement (OTA). Clustering is performed using hierarchical clustering with complete linkage.

3.2.6.3 Static graphlets

Like we did for network motifs, we perform subgraph counting of $u\mathcal{G}_4$. We obtain the GDD matrices for all $G \in \mathcal{N}$ and compute the GDA for all network pairs. We thus obtain a $GDA = |\mathcal{N}| \times |\mathcal{N}|$ matrix, where $GDA(G, H) \approx 0$ means that networks G and H are very different and $GDA(G, H) \approx 1$ means that G and H are very similar.

Figure 3.17 (b) shows the obtained GDA matrix where each cell is colored according to the GDA value and similar networks have a darker cell. Graphlets group bipartite networks and most of the physical interactions networks correctly. By comparison, motif-fingerprints were only capable of finding two large groups, as discussed in the previous section. Neither motifs nor graphlets were able to cluster the set of networks correctly, which might indicate that temporal information is relevant to understand these networks.

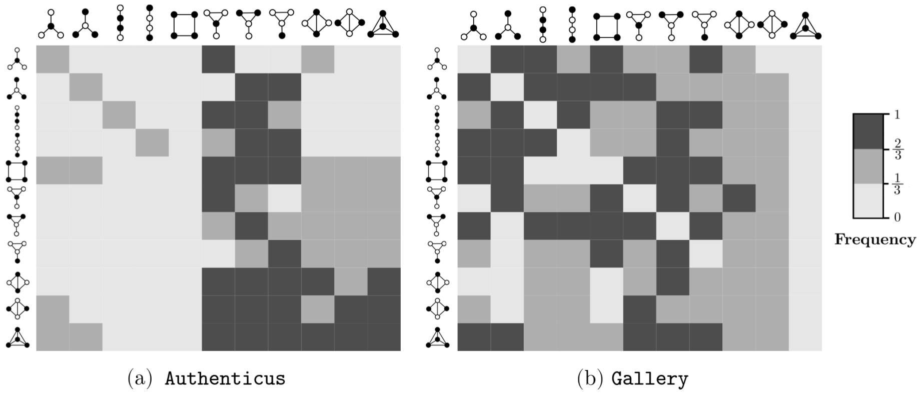
3.2.6.4 Graphlet-orbit Transitions

We consider all possible GoTs of 4-node graphlets. Enumerating larger subgraphs was unnecessary since our method already achieves an adequate grouping for $k = 4$.

Previous studies analyzed graphlet transitions [DBWF11, KKK$^+$12], but orbit transitions give more information since they account for changes of position in the same graphlet, for instance.

Figure 3.18 shows the transition matrices of `Authenticus`, a collaboration network, and `Conference`, a physical interaction network. To simplify visualization, OTA values were discretized into three intervals, indicating *rare* ($[0, \frac{1}{3}]$), *common* ($]\frac{1}{3}, \frac{2}{3}]$) and *frequent* transitions ($]\frac{2}{3}, 1]$). The main diagonal of the matrix suggests that all orbits are relatively stable in `Authenticus` except for the square-orbit O_5. This is expected from collaboration networks since groups forming a square-graph are only loosely connected, therefore these groups tend to either become tighter (transition from O_5 to orbits 6-11) or nearly break apart (transition from O_5 to orbits 1-4). On the other hand, orbits in `Conference` are very unstable, i.e. they almost always change to another orbit. This is explained by the fact that, in short-term physical interaction groups, connections are mostly temporary and not a strong indicator of community. In this example, people meet in a conference and they might meet people that their "group" already met, but they are mostly interested in meeting more people than establishing strong groups. As another example, O_1 shows the effect of hubs in collaboration networks: it is more likely that a hub-like group will gain a new edge between previously unconnected authors (transition from O_1 to O_6) than for to remain unconnected. It is also common that not only one but two new edges appear (transition from O_1 to O_9). However, stars (O_1/O_2) becoming cliques (O_{11}) is rare in `Authenticus`. Interestingly, Figure 3.19 shows that star-to-clique transitions are common in the other collaboration network, `arXiv`. This might come from the fact that, while `Authenticus` data covers multiple areas, `arXiv` only has publications pertaining to physicists; therefore, the observed differences may hint that physicists form tighter connections sooner than the average. It also seems that transitions are relatively slow in collaboration networks since it is rare for a loosely connected subgraph to become a densely connected subgraph in just a single jump. The same cannot be said about `Conference`, where behavior is almost chaotic. These are only some of the possible observations about transition matrices that highlight their interpretive power.

Figure 3.17 (c) clearly shows that graphlet-orbit transitions are able to correctly group our set of temporal networks while motifs and static graphlet-orbits could not (Figure 3.17 (a) and 3.17 (b)).



(a) Authenticus

(b) Gallery

Figure 3.18: Orbit-transition matrices of (a) a collaboration network and of (b) a physical interaction network for all 4-node orbits.

For completeness, Figure 3.19 presents orbit transitions for collaboration, physical interaction, crime and bipartite network. Matrices are discriminated by starting orbit (each matrix) and by network (each matrix-row) for an easier comparison. For instance, the first matrix from Figure 3.19 shows, for each network, the transitions of O_1 to all $O_k \in u\mathcal{O}_4$, the second one of O_2 to all $O_k \in u\mathcal{O}_4$, and so forth. To help visualization we inserted red lines that separate networks of different categories. It is clear that, while networks of the same category have some differences in their orbit-transition profile, they are more alike than networks from different categories. As an example: the transitions of O_1 clearly distinguish co-authorship from physical interaction networks, and also co-authorship from crime and bipartite networks. However, O_1 transitions are very similar for crime and bipartite networks. Distinguishing these two types of networks can be achieved by instead looking at O_5, for instance. Orbit-transition fingerprints are a visual way of interpreting how a network evolves and present very detailed topological and temporal information.

3.2.7 Summary

We put forward a new extension of graphlets for temporal networks (GoTs), as well as a novel metric (OTA) to compare them.

The effectiveness of our proposed method was assessed on (a) synthetic networks pertaining to well-studied graph models and (b) a set of temporal networks with predetermined categories. Our method was shown to be more accurate than competing

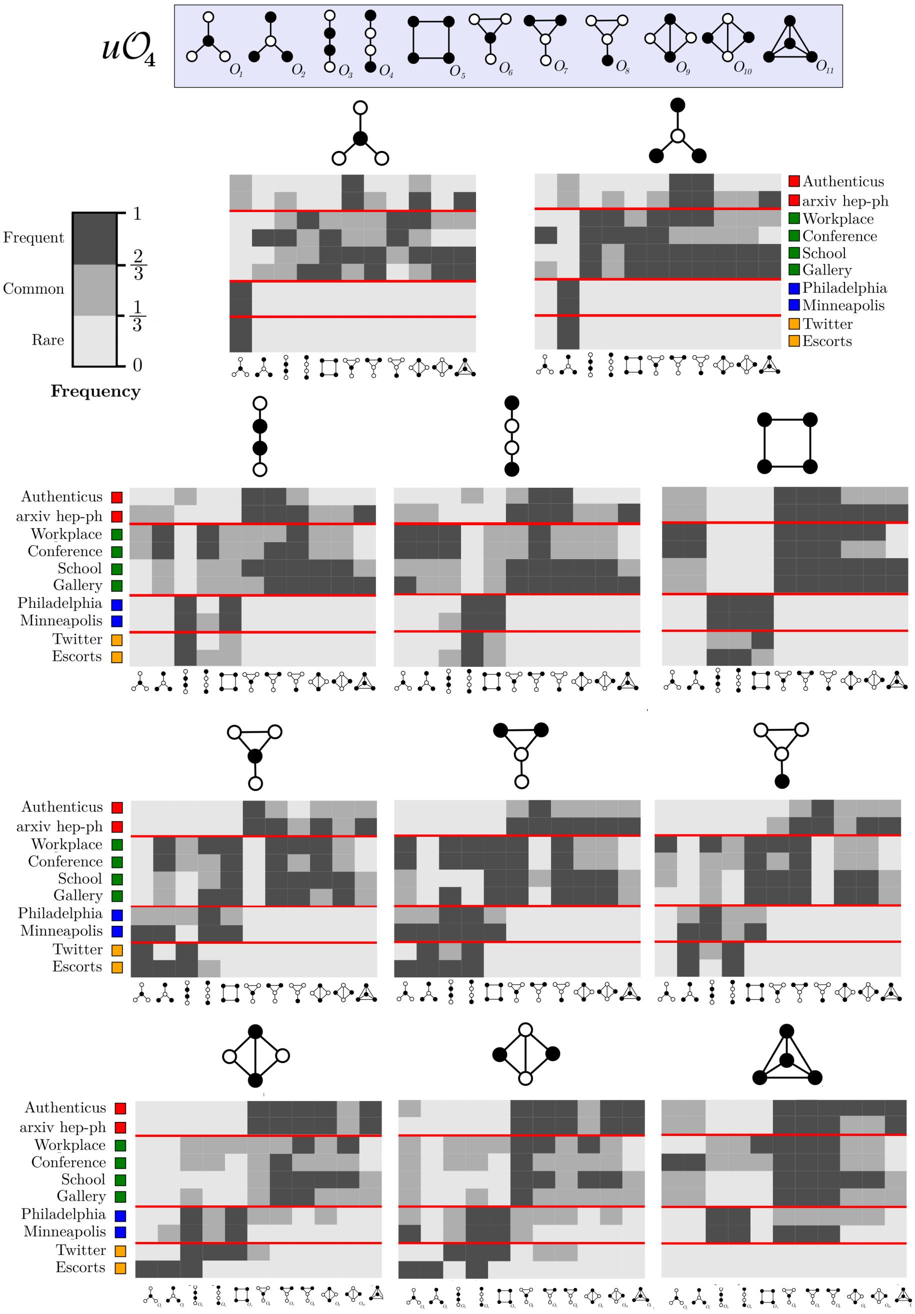


Figure 3.19: Orbit-transition fingerprints for collaboration, physical interaction, crime and bipartite networks. Frequency values are discretized into *rare*, *common* and *frequent* transitions.

approaches on synthetic data. For real networks, we began by analyzing how global metrics evolved over time, namely the average-degree, clustering-coefficient and the characteristic path-length. While these metrics give insight into the topological structure of the networks, we could not visualize that networks of different categories are distinguishable using them. Static network motif and graphlet analyses were also conducted since they capture richer topological information than aforementioned global metrics. However, since they do not take temporal information into account, they are not adequate for temporal network comparison. Our method correctly clustered the set of networks by category, showcasing both the importance of temporal information in these networks and our method's clustering capabilities. Furthermore, our method produces highly interpretable results, leading to a better understanding of network evolution and differences between transitions of distinct networks.

Chapter 4

Network alignment

Network alignment can be used for knowledge transfer from a well-known system to a poorly-studied system between their conserved network regions [ECK16]. For example, in computational biology, network alignment can be used to identify topologically similar (and possibly also sequence-similar) regions of molecular networks of different species and to predict functions of currently unannotated proteins based on functions of their aligned partners in another network [FZM15].

In this chapter we propose a new method to align temporal networks, named GoT-WAVE. We evaluate GoT-WAVE on both synthetic and real networks and compare it with state-of-the-art approaches.

4.1 Motivation

Network alignment (NA) aims to find similar (conserved) regions between compared networks [ESDS16]. These regions are not expected to be perfect fits, and thus, NA deviates from the traditional subgraph isomorphism problem [Ull76] (we formalize the problem in Section 2.3).

Traditional NA methods align *static* networks [VKMM17]. However, because most of real-world systems evolve over time and thus exhibit a dynamic nature, they are intrinsically not static. As such, they can only be truly understood by accounting for their evolution [HS12].

The first methods for NA of *temporal* networks were proposed only recently [VM17b, VCM17]. Since NA is widely applied to biological data (e.g., [MNHP10, SM14, SCTM15]), the lack of NA methods for temporal networks could be due to limitations of current biotechnologies for data collection, which have resulted in a lack of temporal network data on molecular systems, such as protein-protein-interaction networks (PPIs), that are the systems to which static NA methods have been extensively applied [KSK$^+$03, MNHP10, VSM15]. However, as initial temporal PPI data begin to emerge [FM14, YFCM15], and as other temporal network data become available, e.g., brain, ecological, or social networks [GGG$^+$11, BC18, OBEJ08, RA18], temporal NA will gain increasing importance.

4.2 Related work

NA produces either: (a) a many-to-many mapping of highly conserved but small network regions or (b) a one-to-one mapping that covers every node of the smaller network and equally many nodes from the other network and is thus large, but is often suboptimally conserved [ECK16, FMCM15]. Both NA types, called local and global, respectively, have (dis)advantages [MSM16, GM17]. Recent work has tended to focus on global NA [Kaz16] and that is the type we address.

Global NA can be pairwise [Kla09], resulting in aligned pairs of nodes between two networks, or multiple [FND$^+$09, VM17a], resulting in aligned node clusters between three or more networks. Multiple NA is more computationally complex than pairwise NA and, furthermore, recent work suggests that multiple NA is also less accurate than pairwise NA [VKMM17]. So, here, we focus on pairwise NA, and in particular on global pairwise NA (GPNA).

GPNA consists of two algorithmic components: 1) an objective function, typically node conservation (a measure of node similarity) combined with edge conservation, and 2) an optimization strategy (also called alignment strategy) that aims to maximize the objective function.

Regarding the first component and specifically the node conservation part, graphlet degree vectors (GDVs) [Prž07a, MP08] have been widely used as topological properties (features) to measure node conservation in GPNA due to the rich topological information that graphlets capture [MNHP10, MP12, VSM15]. GDV-based node conservation was shown to be superior in the task of GPNA under the same optimization strategy to other node conservation/similarity measures: IsoRank's PageRank [SXB08, LLB$^+$09] and GHOST's spectral signature measures [PK12] from the biological domain [FZM15, CSM15], or node2vec [GL16] and struc2vec [RSF17] network embedding measures from the social domain [GM18]. Regarding the first component and specifically the edge conservation part, several established measures of edge conservation exist: S^3, which rewards an alignment when edges are aligned to each other and penalizes it when an edge is aligned to a non-edge [SM14], and weighted edge conservation (WEC), which is high if many edges are aligned to each other *and* the nodes of the aligned edges are similar with respect to node conservation [SCTM15].

Regarding the second component, existing GPNA algorithms have one of two types of optimization strategy. One type is seed-and-extend, where first two highly similar nodes (with respect to node conservation) are aligned, i.e., seeded. Then, the seed's network neighbors that are similar are aligned, the seed's neighbor's neighbors that are similar are aligned, and so on. The extension around the seed and exploration of the seed's neighbors aims to improve both node and edge conservation of the resulting alignment. The extension continues until all nodes in the smaller network are aligned, i.e., until a one-to-one (injective) node mapping is produced. WAVE is a representative state-of-the-art seed-and-extend optimization strategy (that we focus on for reasons discussed below), which by default optimizes GDV-based node conservation and WEC [SCTM15]. The other type of optimization strategy is a search algorithm. Here, instead of aligning node by node as with the seed-and-extend approach, entire alignments are explored and the one with the best objective function score is returned. MAGNA++ [VSM15] is a representative state-of-the-art search algorithm (that we focus on for reasons discussed below), which uses a genetic algorithm to, by default, optimize GDV-based node conservation and S^3. Importantly, a typical optimization strategy, including WAVE and MAGNA++, can optimize any objective function, i.e., it is not limited to e.g., GDV-based node conservation and S^3 or WEC. Note that in a recent comprehensive evaluation of different methods [MSM16], WAVE and MAGNA++ rose to the top, although newer GPNA methods have appeared since, such as SANA [MH17].

The only temporal GPNA methods that were available before our contribution were DynaMAGNA++ [VCM17] and DynaWAVE [VM17b], which are temporal extensions of MAGNA++ and WAVE. DynaWAVE was shown to be more accurate and faster than DynaMAGNA++ on medium- and large-size networks; DynaMAGNA++ was more accurate (yet slower) on small-size (≈100-node) networks. Since most of real-world networks are not small, we focus on DynaWAVE. This method uses the same seed-and-extend optimization strategy as static WAVE, but it uses it to optimize *dynamic* node and edge conservation. As its dynamic node conservation, DynaWAVE uses a temporal extension of GDVs, dynamic GDVs (DGDVs), which were originally proposed for tasks of node and network classification by [HCM15]. DGDV of a node uses dynamic graphlets to describe the node's neighborhood in a temporal network. Comparing nodes' DGDVs yields a measure of similarity between the nodes' evolving neighborhoods, i.e., dynamic node conservation. As its dynamic edge conservation, DynaWAVE uses dynamic WEC (DWEC), a temporal analog of WAVE's WEC that generalizes an aligned edge to an aligned event (temporal edge) [VM17b]. Just as WAVE, DynaWAVE can use its optimization strategy in combination with any objective function.

4.3 Overview of our contribution

We recently developed graphlet-orbit transitions (GoTs) [ARS18], a different temporal graphlet measure of node similarity. GoTs describe how a node's neighborhood is evolving by measuring how its participation in different graphlet positions (orbits) changes with time. For example, GoTs can capture when a node in the center of a k-node star at time t becomes a part of a k-node clique at time $t + 1$. GoTs are discussed in more detail in Section 4.5.

Here, we aim to use GoTs for temporal GPNA as a new dynamic node conservation measure within DynaWAVE. We refer to our GoT-modified version of DynaWAVE as GoT-WAVE.

We evaluate whether GoT-WAVE improves upon DynaWAVE by following the same evaluation methodology from the DynaWAVE study [VM17b]. Namely, we evaluate on synthetic data containing 50 temporal networks produced by dynamic versions of five well known graph models. Here, we align all pairs of networks to each other. A good temporal GPNA method should identify as similar those networks that originate from the same model and as dissimilar those networks that originate from different models. Also, we compare the methods on eight real-world networks from biological and social domains. Here, we align each network to its noisy version, in which a percentage of the original network's edges is rewired. Since the aligned networks have the same nodes, we know which nodes should be mapped to which nodes. The more nodes are correctly mapped, the better the method. In all evaluation tests, we compare the two methods when they optimize: 1) only their respective dynamic node conservation measures (GoTs versus DGDVs), to *fairly* evaluate the two measures against each other, and 2) both node and edge conservation, to give each method the *best-case* advantage (it was already shown that DynaWAVE performs better when it optimizes both rather than only one of node and edge conservation).

We find that on synthetic networks, under both the fair and best-case scenario, GoT-WAVE is more accurate than DynaWAVE by 25% and faster by 64%. On real networks, under the fair scenario, GoT-WAVE is more accurate that DynaWAVE for four of the eight networks, performing better on the denser networks and worse on the sparser ones. We observe the opposite in terms of their running times, i.e., GoT-WAVE is slower than DynaWAVE for denser networks and faster for sparser networks. Thus, the two methods are complementary. Under the best-case scenario, DynaWAVE's performance is more enhanced than GoT-WAVE's when dynamic edge conservation is considered as well, as DynaWAVE is now better for all eight networks. However, because GoTs is the only current temporal graphlet-based measure of node similarity that supports edge direction, GoT-WAVE is the only temporal GPNA method that can deal with directed networks (DGDVs and thus DynaWAVE always assume that edges are undirected).

Thus, GoT-WAVE is a promising new temporal GPNA method that efficiently optimizes dynamic node conservation. Finding new measures of dynamic edge conservation better suited for GoT-WAVE could further enhance its performance, which is the subject of future work. Also, GoTs, when used as node conservation within any newer or future GPNA optimization strategies, such as SANA, could yield further improvements.

4.4 Static and temporal GPNA

Static GPNA produces an injection $f : V(G) \to V(H)$, where $V(G)$ is not bigger than $V(H)$, maximizing node or edge conservation between aligned node pairs (Section 2.4 offers more details). Temporal GPNA, extending static GPNA, aims to optimize dynamic node or edge conservation. As dynamic node conservation, we optimize similarity between nodes' temporal graphlet-based features, namely GoTs, which we define in Section 4.5. We compare nodes' GoTs as described in Section 4.6. As dynamic edge conservation, when we also optimize this measure, we use DWEC, just as DynaWAVE does. That is, compared to DynaWAVE, the only aspect that we modify is its DGV-based dynamic node conservation measure, replacing it with our GoT-based measure. This ensures a fair comparison between GoTs and DGDVs as two different temporal graphlet-based node features.

4.5 GoTs as node conservation features

Just like DGDVs, GoTs only account for connected graphlets, because our focus is to study how groups evolve. We should point out that considering transitions to/from disconnected graphlets could be useful to study group formation, but computing their frequencies requires considering $\binom{n}{k}$ subsets for each snapshot, which is only feasible in practice for small networks and very small k-graphlets. All possibilities for the 3-node graphlets are illustrated in Figure 3.10; in practice, we use larger graphlets as well (see below). The matrix from Figure 3.10 illustrates the GoTs of node x that we use as x's feature vector. For more details on GoTs, namely how they are extracted, we refer the reader to Section 3.2.3. This matrix offers rich topological information that can be used for various tasks [ARS18]. Here, we use it in the task of temporal GPNA.

Regarding the considered graphlet size, [HCM15] recommended the use of all DGDVs with up to four nodes and six events (temporal edges). This is what we do, to give the best-case advantage to DynaWAVE. For a fair comparison, to account for as similar as possible amount of network topology with both DGDVs and GoTs, we also use all undirected GoTs with up to four nodes, unless explicitly stated otherwise.

4.6 GoT-WAVE

For each node, we compute its GoTs frequency matrix, flatten the matrix to a vector, and use the vector as the node's features. The feature vectors over all nodes in a network form a $\#Nodes \times \#Transitions$ matrix. For two networks being aligned, this results in two corresponding matrices with the same number of columns, whose rows are then joined together. Due to high dimensionality and sparsity of the joined matrix, we perform dimensionality reduction on the matrix using principal component analysis, keeping 99% of its variance. Then, we compute the topological similarity between every two nodes from different networks as the cosine similarity between the nodes' PCA-reduced feature vectors. GoT-WAVE uses the resulting node similarities as the dynamic node conservation part of the objective function, which is then optimized using WAVE. In all of the above steps, we do exactly what DynaWAVE does to produce DGDV-based node similarities and perform DGDV-based temporal GPNA.

GoT-WAVE, like DynaWAVE, can optimize dynamic node conservation (GoTs for GoT-WAVE, DGDVs for DynaWAVE), dynamic edge conservation (DWEC), or both. Its objective function is $\alpha S_E + (1-\alpha)S_N$, where S_E and S_N are dynamic edge and node conservation measures, respectively, and $\alpha \in [0,1]$ controls how important each measure is. We use: 1) $\alpha = 0$, to *fairly* evaluate the two measures against each other, meaning that only dynamic node conservation is considered, or 2) $\alpha = \frac{1}{2}$, to give each method the *best-case* advantage, since this α value seems to work the best for DynaWAVE [VM17b].

4.7 Experimental Evaluation

Results were gathered on an Intel i7-6700 CPU at 3.4GHz with 16GB of RAM. Execution times were obtained using a single core for computation.

In the following tests we measure potential improvement of GoT-WAVE over DynaWAVE as follows.

Let us denote by S_G the (accuracy or running time) score of GoT-WAVE, and by S_D the score of DynaWAVE. Also, let us denote by G_A the relative gain of GoT-WAVE over DynaWAVE in terms of accuracy, and by G_T the relative gain GoT-WAVE over DynaWAVE in terms of running time. Since for accuracy, a larger score is better, we define $G_A = \frac{S_G - S_D}{min(S_G, S_D)} \times 100\%$. On the other hand, since for running time, a lower score is better, we define $G_T = \frac{S_D - S_G}{min(S_G, S_D)} \times 100\%$. In both cases, positive gain (i.e., a positive G_A or G_T value) would indicate improvement of GoT-WAVE compared to DynaWAVE, and negative gain (i.e., a negative G_A or G_T value) would indicate degradation of GoT-WAVE compared to DynaWAVE. For example, in terms of accuracy, if GoT-WAVE has accuracy of 1 and DynaWAVE has accuracy of 0.7, then $G_A = \frac{1-0.7}{0.7} \times 100\% = 43\%$ (i.e., GoT-WAVE is superior to DynaWAVE). On the other hand, if GoT-WAVE has accuracy of 0.7 and DynaWAVE has accuracy of 1, then $G_A = \frac{0.7-1}{0.7} \times 100\% = -43\%$ (i.e., GoT-WAVE is inferior to DynaWAVE). As another example, in terms of running time, if GoT-WAVE takes 2 seconds and DynaWAVE takes 6 seconds, then $G_T = \frac{6-2}{2} \times 100\% = 200\%$ (i.e., GoT-WAVE is superior to DynaWAVE). On the other hand, if GoT-WAVE takes 6 seconds and DynaWAVE takes 2 seconds, then $G_T = \frac{2-6}{2} \times 100\% = -200\%$ (i.e., GoT-WAVE is inferior to DynaWAVE).

4.7.1 Evaluation using synthetic networks

As often done [PKSH10, MNHP10, HCM15], we compare DynaWAVE and GoT-WAVE on a set of synthetic networks from different graph models. We develop temporal versions of well-known models: Erdős-Rényi random graphs [ER60], Barabási-Albert preferential attachment [BA99], Watts–Strogatz small-world networks [WS98], geometric gene duplication model with probability cutoff [PKSH10] and scale-free gene duplication [VFMV03]. A good GPNA method should identify networks from the same model as being more topologically alike (that is, a having higher alignment quality, i.e., objective function score) than networks from different models.

4.7.1.1 Synthetic networks

We generate networks with 24 snapshots each (i.e., $|\mathcal{S}(G)| = 24$). In each snapshot $S_t(G)$, new nodes arrive at the network and new edges are added to it until the desired edge density is reached. For a more concise terminology, we use $N_t = |\mathcal{V}(S_t(G))|$, where t is the number of the snapshot. Node arrival is either linear ($N_t = \frac{N_T - N_1}{T-1} \cdot (t-1) + N_1$) or exponential ($N_t = N_1 \cdot e^{\frac{(t-1)}{10}}$), T is the total number of snapshots, N_1 is the number of nodes at the start and N_t is the number of nodes at snapshot t. We set the arrival function of each model according to what was reported as the observed node arrival function for similar models [LBKT08]. How new edges are added (i.e., which nodes they connect) is specific to each model.

- **Erdős-Rényi (Random):** Two nodes are chosen at random and connected. Past edges are kept.
- **Barabási-Albert (ScaleFree):** Two nodes u and v are chosen at random but they become connected only if $\frac{max(deg(u),deg(v))}{|E|} > r$, where r is a randomly generated value $\in [0, 1]$. Therefore, nodes with higher degrees have a bigger chance of gaining new connections ("the rich get richer"). Past edges are kept.
- **Watts–Strogatz (Small-world):** At $t = 0$ an initial ring network is created, where each node is connected to $\approx k$ neighbors. Since $N_0 = 100$ and the edge density is 1%, $k = 1$ for $t = 0$. Edges are then randomly rewired with probability $\beta = 0.2$. For $t > 0$, new nodes arrive and a ring is formed again (with a possibly different k) while keeping the rewired connections previously added. Rewiring is again performed, both on the new ring and on the old randomized edges. Therefore, past edges might disappear.

- **Geometric gene duplication (Geo-GD):** Initially k seed-nodes are placed close-by in a two-dimensional space: $d(u,v)^2 < \epsilon$. As suggested by [PKSH10], $\epsilon = 5 \cdot 10^{-2}$ and $k = 5$ are used. Nodes are incrementally added to the network one at a time, and each new node u is placed at a distance $d(u,f)^2 \leq \epsilon$ from a random *father-node* f already in the network. The model includes parameter p which controls how likely node u is to *cut-off* from f; for our purposes $p = 0.2$ since it gives origin to realistic PPI networks [PKSH10]. When a node cuts-off from its father, it is placed at a distance of 10ϵ at most. Node additions stop when the desired number of nodes is achieved and the closest 1% edges are added to snapshot t, while the other possible edges remain unactivated. It is possible that past edges disappear since close edges in snapshot t are not guaranteed to remain in the 1% closest edges of snapshot $t+1$.

- **Scale-free gene duplication (ScaleFree-GD):** After a few seed edges are added to the network, each new node is (i) connected to a random father-node and it (ii) copies the father's connections. The model has two parameters: p controls the likelihood of child- and father-nodes being connected by an edge, and q sets the probability of the child-node keeping his father's connections to other nodes. For our purposes, $p = 0.3$ and $q = 0.7$ since these values generated realistic PPI networks [PKSH10]. The model also sets a 50% chance that when an edge is not successfully replicated from father to child either (a) the father keeps the edge but the child does not copy it or (b) the child steals the connection from the father; therefore, past edges can be lost.

Edge density is set at $\approx$ 1% for all models, mimicking real-world networks (such as PPIs, internet routing and email networks [Mel06]), and remains stable for all snapshots (e.g., this stability was observed in online social networks by [HW09]). Each network starts with 100 nodes and grows to 1,000 nodes. We generate ten networks for each of the five graph models, giving us 50 networks with 24 snapshots each, totaling to 1200 snapshots. Details of each model in Table 4.1.

4.7.1.2 Performance on synthetic networks

With each of GoT-WAVE and DynaWAVE, we align all pairs of synthetic networks. We compute objective function scores of all alignments. We compute the area under the precision-recall (AUPR) or receiver operating characteristic (AUROC) curve. We compare GoT-WAVE and DynaWAVE with respect to these measures. Note that given

Table 4.1: Set of graph models used in our experiments. All networks (regardless of the model) have 24 snapshots, start with 100 nodes, grow until they reach 1000 nodes, and have edge density of = 1% in all snapshots. Node arrival (linear or exponential) is set to what was reported in [LBKT08] for similar models. How nodes are connected (i.e., how new edges are added) depends on the model.

Model	Node arrival	New edges
Random	linear	random
ScaleFree	exponential	preferential Attachment
Small-world	linear	ring + rewire ($\beta = 0.2$)
Geo-GD	linear	duplication w/ cut-off ($p = 0.2$)
ScaleFree-GD	exponential	duplication w/ edge loss ($p = 0.3$, $q = 0.7$)

the five graph models, the expected AUROC by chance is 0.2. Section 2.3 provides more details on how the precision-recall, ROC curves, and their areas are obtained.

Table 4.2: Results on synthetic networks when only node conservation is optimized ($\alpha = 0$) or when node and edge conservation are optimized ($\alpha = \frac{1}{2}$). In parentheses, we show relative improvement (positive gain) or degradation (negative gain) in performance of GoT-WAVE compared to DynaWAVE.

	AUPR	
α	DynaWAVE	GoT-WAVE
0	0.63	**0.79 (+25%)**
$\frac{1}{2}$	0.59	0.53 (-11%)
	AUROC	
α	DynaWAVE	GoT-WAVE
0	0.59	**0.78 (+32%)**
$\frac{1}{2}$	0.54	0.70 (+30%)

(a) Accuracy.

Model	DGDVs	GoTs
Random	26s	**22s** (+18%)
ScaleFree	**22s**	25s (-14%)
Small-world	23s	**4s (+475%)**
Geo-GD	34s	**11s (+210%)**
ScaleFree-GD	16s	**12s (+33%)**
Total	121s	**74s (+64%)**

(b) Feature extraction times.

Under the fair-case scenario, when optimizing solely node conservation ($\alpha = 0$), GoT-WAVE's AUPR and AUROC are higher by 25% and 32%, respectively, than DynaWAVE's (Table 4.2 (a)).

For this particular dataset, also optimizing edge conservation (i.e., $\alpha = \frac{1}{2}$) decreases performance of both methods, even though it was previously argued that $\alpha = \frac{1}{2}$ is the best-case scenario [VM17b, VCM17]. Actually, we also verify that $\alpha = \frac{1}{2}$ is indeed the best-case scenario on our considered real networks (Section 4.7.2). It is just that on our considered synthetic networks, $\alpha = 0$ happens to be both the fair and best-case scenario for both methods, and under this scenario, GoT-WAVE is superior to DynaWAVE.

On synthetic networks, we also find that extracting GoT features is overall 64% faster than extracting DGDV features (Table 4.2 (b)). Because both methods use their features in the same alignment strategy (WAVE), their alignment times are similar, as expected (Table 4.3).

Table 4.3: Average time to align two networks when using DGDVs or GoTs. We compute the time of aligning each model (e.g., the time to align each of the **Random** networks with any other networks). Since each of the 5 models m has 10 instances n (i.e., networks), we consider $\frac{(n-1)\times n}{2} = 45$ alignments of a given network to networks of its own model (e.g., align two **Random** networks) and $n \times (m-1) \times n = 400$ alignments to networks of different models (e.g., align a **Random** network with a **ScaleFree** network). For each model, we average the time over all 445 considered alignments.

Model	DGDVs	GoTs
Random	6.219s	**6.078s** (+2%)
ScaleFree	6.196s	**6.056s** (+2%)
Small-world	6.181s	**6.049s** (+2%)
Geo-GD	6.027s	**5.925s** (+2%)
ScaleFree-GD	6.009s	**5.868s** (+2%)
Total	30.632s	**29.976s** (+2%)

We also performed a subset of all tests for synthetic networks, and specifically those under the fair evaluation scenario ($\alpha = 0$), using the other existing DGDV-based temporal GPNA method, DynaMAGNA++, and a GoT-modified version of it, which we refer to as GoT-MAGNA++. Here, we used the following values of MAGNA++'s parameters: population size of 1000 and 1000 generations. These results are qualitatively similar to those reported above: GoT-MAGNA++'s AUPR and AUROC are 16% and 22% higher, respectively, than DynaMAGNA++'s. However, we maybe did not give DynaMAGNA++ the best-case advantage, because this method was shown to work the best for $\alpha = \frac{1}{2}$, and under larger values of its parameters than those that we were able to consider due to MAGNA++'s high running time. So, it is possible that the performance of DynaMAGNA++ (and GoT-MAGNA++) could be improved. However, testing this would be unnecessary, given that DynaWAVE was already shown to outperform DynaMAGNA++ in terms of both accuracy and running time on all networks but the smallest ones (with ≈100 nodes). So, we believe that our detailed tests against DynaWAVE are sufficient.

4.7.2 Evaluation using real-world networks

Section 4.7.1 studies GPNA at the network level (whether networks are from the same model), while here we study GPNA at the node level (whether nodes are correctly aligned). A typical process to evaluate GPNA at the node level on a real network is to insert artificial noise into the network, that is, rewire a percentage of its temporal edges (events), and align the original network to the noisy version [VM17b]. Then, since the aligned networks have the same nodes, we can measure the percentage of all nodes that are correctly aligned, i.e., node correctness.

To randomize a dynamic network, we use 3 different randomization schemes. For undirected networks, we use an established randomization scheme [Hol15], which we refer to as *undirected randomization*. This scheme chooses two random events and swaps their time stamps with some probability. For directed networks, we use a variation of the above scheme that has an additional parameter that controls the probability of switching the edge directions of the events, which we refer to as *directed randomization*. For directed networks, we use an additional randomization scheme that only swaps the edge direction of events but not their time stamps, which we refer to as *pure directed randomization*. More details on the randomization schemes are in Appendix B. For a given scheme, we study 10 randomization (i.e., noise) levels, from 0% to 20% in increments of 2%. At each noise level, we produce five random network instances and average the results over the five runs.

First, for a given method, at each noise level, for each alignment, we compute the corresponding objective function score. Ideally, the objective score should decrease as the network is aligned to progressively noisier versions. Furthermore, since we know the perfect alignment between the original network and each of its randomized versions (as their nodes are the same), we compute the ideal objective score – the quality of the perfect alignment, as measured by DynaWAVE's and GoT-WAVE's objective function. We denote the objective scores of the ideal and method-produced alignments for noise n by $S_{i,n}$ and $S_{p,n}$, respectively. The expectation is that a good method's produced objective score should be similar to the method's ideal objective score, i.e., $|S_{p,n} - S_{i,n}|$ should be as close as possible to 0. Also, since we want to account for scaling (e.g., the difference of 0.1 between 0.9 and 0.8 is not the same as the difference of 0.1 between 0.3 and 0.2), we divide the difference between the produced and ideal alignment by their maximum, i.e., $max(S_{p,n}, S_{i,n})$. With these points is mind, we compute the distance $dis(S_p, S_i)$ over all considered noise levels n (from 0% to 20%) as: $dis(S_p, S_i) = \sum_{n=0\%}^{20\%} \frac{|S_{p,n}-S_{i,n}|}{max(S_{p,n},S_{i,n})}$

For each real network, we compute this distance for each of GoT-WAVE and DynaWAVE. Then, we summarize gain of GoT-WAVE compared to DynaWAVE as follows. Let us denote by S_G the distance score of GoT-WAVE, and by S_D the score of DynaWAVE. Since a lower distance score is better, we compute the relative gain of GoT-WAVE over DynaWAVE, denoted by G_O, as: $G_O = \frac{S_D - S_G}{min(S_G, S_D)} \times 100\%$. Positive gains mean than GoT-WAVE is superior to DynaWAVE and negative gains mean that GoT-WAVE is inferior to DynaWAVE.

Second, we compare GoT-WAVE and DynaWAVE in terms of node correctness (see above). Let us denote by S_G the node correctness of GoT-WAVE, and by S_D the node correctness of DynaWAVE. Since higher node correctness is better, we compute the relative gain of GoT-WAVE over DynaWAVE, denoted by G_{NC}, as: $G_{NC} = \frac{S_G - S_D}{min(S_G, S_D)} \times 100\%$. Again, positive gains mean than GoT-WAVE is superior to DynaWAVE and negative gains mean that GoT-WAVE is inferior to DynaWAVE.

4.7.2.1 Real-world temporal networks

We analyze eight real networks (Table 4.4). Six of them are undirected, three of which are biological networks from the DynaWAVE study [VM17b], and three are social networks. We use two additional directed temporal networks.

Table 4.4: Real-world temporal networks used in our experiments.

	Network	Nodes	Events	Snapshots	Description
Undirected	**zebra** [RSF+15]	27	500	57	Zebra proximity network
	yeast [VCM17]	1,004	10,403	8	Yeast PPI
	aging [FM14]	6,300	76,666	38	Human aging PPI
	school [GBC14]	327	7,388	5	School proximity network
	gallery [ISB+11]	420	22,476	16	Gallery proximity network
	arxiv [LKF07]	2,504	138,495	7	Paper co-authorships
Dir.	**emails** [MPK11a]	167	8,771	9	E-mail communication
	tennis [ARS16]	876	103,938	42	Player dominance network

4.7.2.2 Performance on real undirected networks

In terms of the objective score (Figure 4.1), both DynaWAVE and GoT-WAVE show adequate behavior, i.e., the objective score decreases as we add more noise. When optimizing solely node conservation ($\alpha = 0$), we observe that: (i) for **gallery** and **zebra** networks, both methods closely match their ideal alignments over all noise levels; (ii) for **yeast** and **aging** networks, both methods closely match their ideal

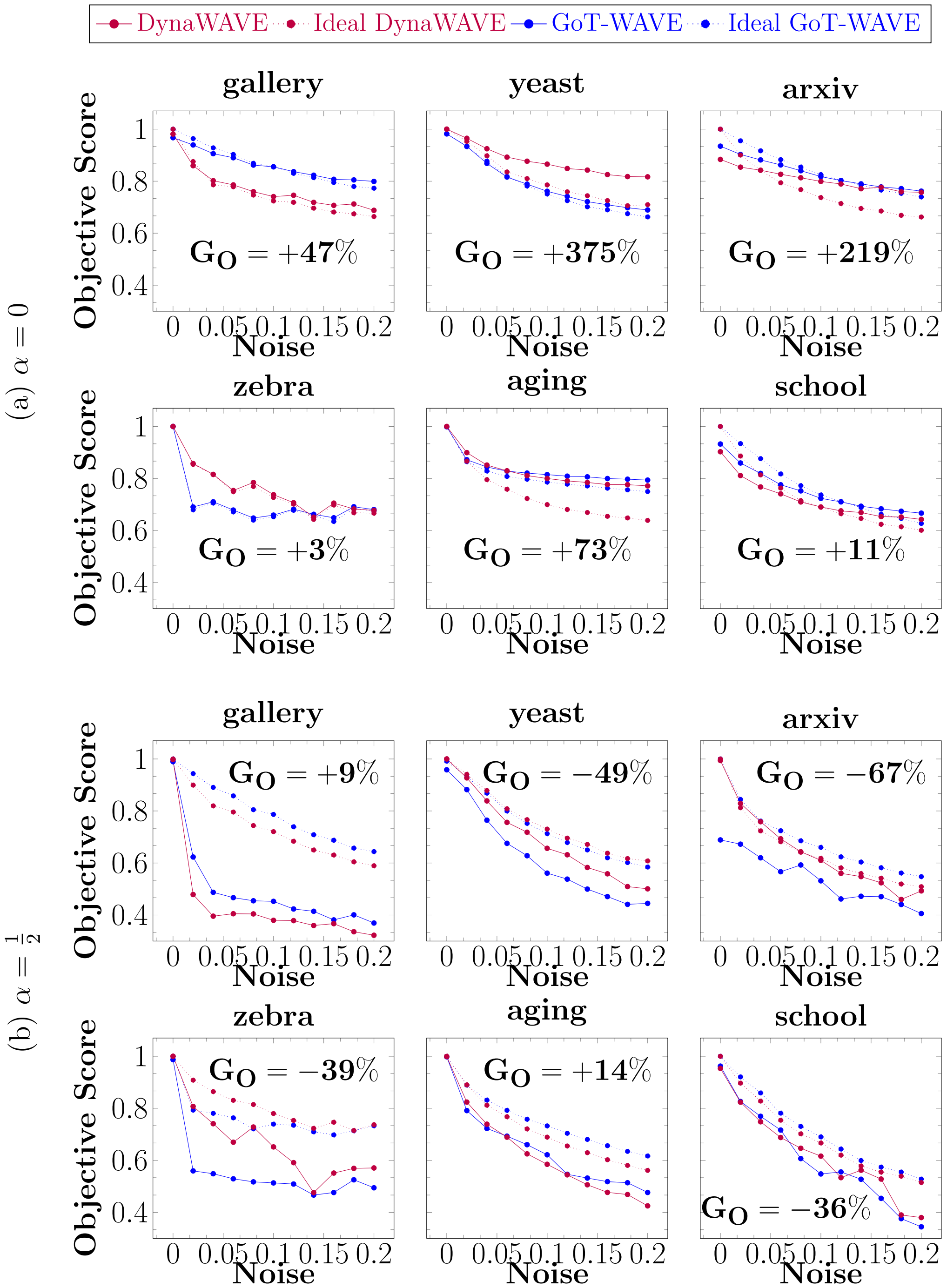


Figure 4.1: Comparison between GoT-WAVE and DynaWAVE on undirected networks in terms of how well their alignments' objective scores match the objective scores of ideal alignments, when (a) only node conservation is optimized ($\alpha = 0$) and (b) both node and edge conservation are optimized ($\alpha = \frac{1}{2}$). Recall that G_O is the relative gain of GoT-WAVE over DynaWAVE (positive: GoT-WAVE is superior; negative: DynaWAVE is superior).

alignments for low noise levels, but for high noises levels, DynaWAVE drifts away from its ideal alignments while GoT-WAVE still closely matches its ideal alignments; and (iii) for **arxiv** and **school** networks, both methods are far from their ideal alignments for low noise levels, but for high noise levels, GoT-WAVE closely matches its ideal alignments while DynaWAVE is still far from its ideal alignments. In other words, in terms of the total gain G_O, GoT-WAVE improves upon DynaWAVE, more closely matches its ideal alignments than DynaWAVE, for all six networks. When optimizing both node and edge conservation ($\alpha = \frac{1}{2}$), GoT-WAVE more closely matches its ideal alignments for two out of the six networks (**gallery** and **aging**). So, the two methods can be seen as complementary.

In terms of node correctness (Table 4.5 and Figure 4.2), for $\alpha = 0$, the two methods are again complementary - each is the best for three of the six networks. For $\alpha = \frac{1}{2}$, DynaWAVE's node correctness improves more substantially than GoT-WAVE's, which is why now DynaWAVE is superior for most (though not all) of the networks. We show an example of why that might be in Figure 4.3. In short, GoT-WAVE already captures some of the information that DWEC captures and thus does not benefit much from using it, while DynaWAVE captures different information from DWEC and thus benefits more from using it. Note that the superiority of one method over the other one is typically consistent over all noise levels, for both $\alpha = 0$ and $\alpha = \frac{1}{2}$.

In terms of running time (Table 4.6(a)), extracting GoT features is faster than extracting DGDV features for the sparser networks (**zebra**, **aging** and **school**) and slower for the denser networks (**aging**, **arxiv** and **gallery**). Denser networks induce more GoTs than dynamic graphlets (Table 4.6(b)), and thus, GoTs are computationally heavier. Just as for synthetic networks, because both methods use the same alignment strategy (WAVE), their alignment times are similar (Table 4.7).

Table 4.5: Node correctness when aligning an undirected real network to itself (noise = 0). In parentheses, we show relative improvement (positive gain) or degradation (negative gain) in performance of GoT-WAVE compared to DynaWAVE. In bold, we show the best result for each network.

	(a) $\alpha = 0$		(b) $\alpha = \frac{1}{2}$	
Network	DynaWAVE	GoT-WAVE	DynaWAVE	GoT-WAVE
zebra	**0.926** ± 0.05	0.578 ± 0.09 (-60%)	0.911 ± 0.04	0.615 ± 0.14 (-48%)
yeast	**0.966** ± 0.01	0.924 ± 0.01 (-5%)	**0.966** ± 0.01	0.919 ± 0.01 (-5%)
aging	0.912 ± 0.01	0.942 ± 0.01 (+3%)	**0.959** ± 0.01	0.955 ± 0.01 (-0.4%)
arxiv	0.340 ± 0.02	0.446 ± 0.02 (+31%)	**0.658** ± 0.01	0.602 ± 0.04 (-9%)
gallery	0.507 ± 0.03	0.485 ± 0.03 (-5%)	**0.557** ± 0.01	0.531 ± 0.01 (-5%)
school	0.735 ± 0.03	0.861 ± 0.03 (+17%)	**0.973** ± 0.01	0.971 +- 0.01 (-0.2%)

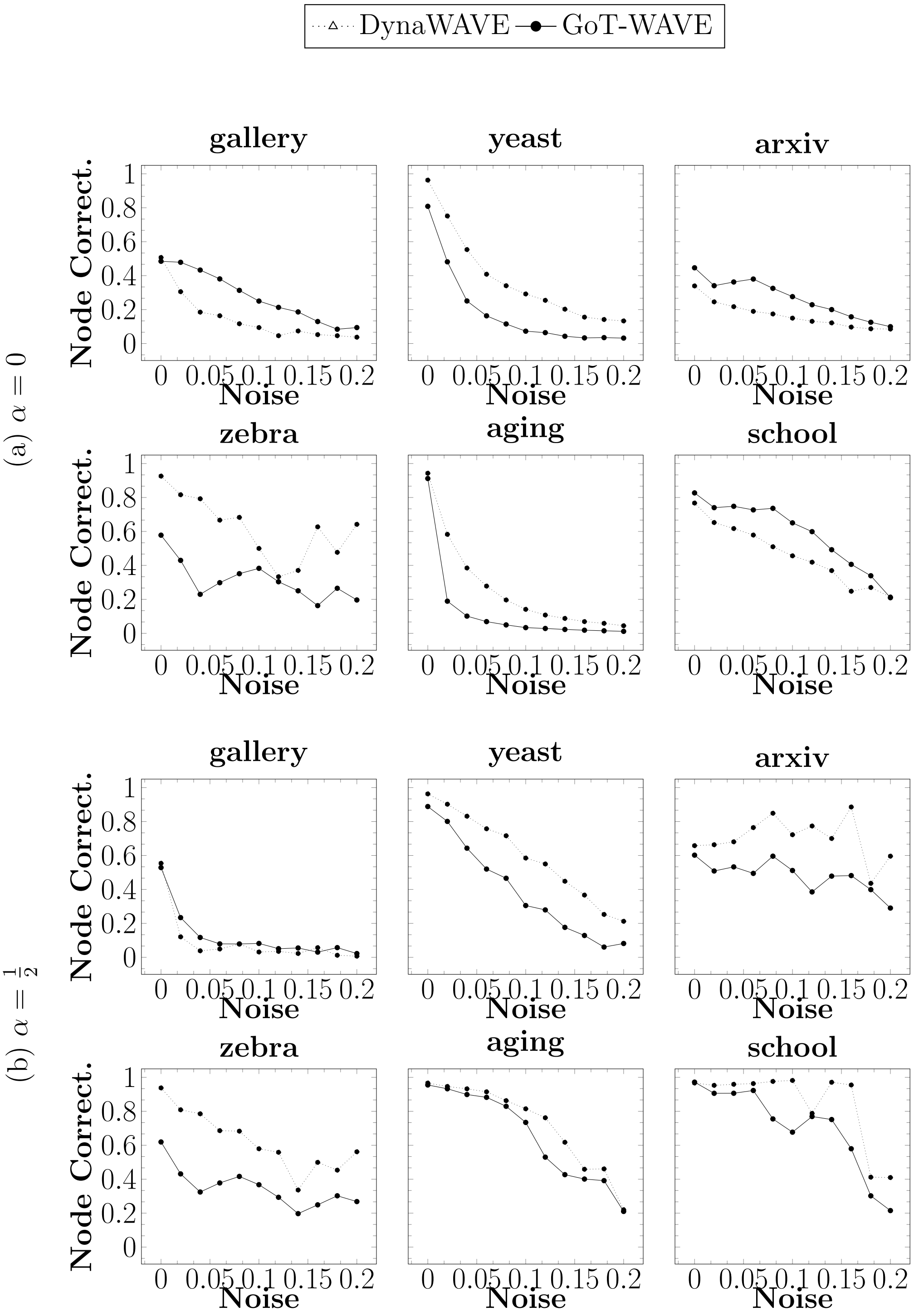


Figure 4.2: Comparison between GoT-WAVE and DynaWAVE on undirected networks in terms of node correctness, when (a) only node conservation is optimized ($\alpha = 0$) and (b) both node and edge conservation are optimized ($\alpha = \frac{1}{2}$). The higher the node correctness, the better the method.

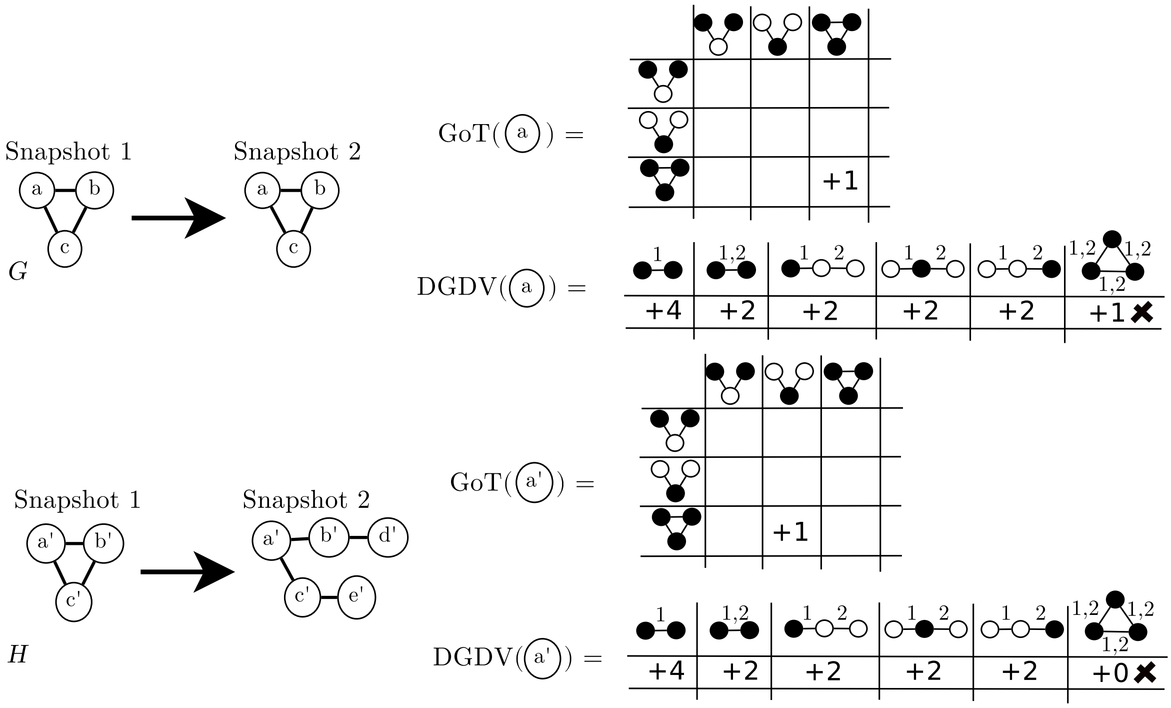


Figure 4.3: We observe that, when both node and edge conservation are considered ($\alpha = \frac{1}{2}$), in contrast to when only node conservation is considered ($\alpha = 0$), DynaWAVE has better results than GoT-WAVE consistently. Here we hypothesize why that might the case with an example. When we use GoTs, node a from network G and node a' from network H are correctly identified as different (i.e., node a and node a' have different GoTs). When we use DGDVs, node a and node a' are incorrectly identified as similar/equal (i.e., node a and node a' have the same DGDVs). For details on DGDVs we refer the reader to [HCM15]. DGDVs can not distinguish between nodes a and a' because a dynamic graphlet only allows for one event per time-step. For instance, the last graphlet in the DGDVs is not allowed (and not calculated). Thus, for this case, DGDVs can only distinguish nodes a and a' when edge conservation is also considered. Cases such as these show why DGDVs have bigger benefits in using DWEC than GoTs.

Table 4.6: Results on undirected real networks in terms of (a) feature extraction times and (b) number of subgraph occurrences (i.e., number of dynamic graphlets or GoTs found on the network). GoTs induce many more occurrences than DGDVs, and this is especially true for denser networks, such as **aging**, **arxiv** and **school**. Due to our fast enumeration algorithm based on g-tries [RS14b], GoTs' extraction is still faster for the sparser networks, namely **zebra**, **yeast** and **school**. In parentheses, we show relative improvement (positive gain) or degradation (negative gain) in performance of GoT-WAVE compared to DynaWAVE. We assume that more occurrences means degradation. Thus, GoT-WAVE shows a much higher degradation in terms of number of occurrences than in execution time (-1,886% versus -142%), showcasing our enumeration algorithm's efficiency.

	(a)		(b)	
Network	Execution time		#Occurrences	
	DGDVs	GoTs	DGDVs	GoTs
zebra	0.06s	**0.02s (+200%)**	9.676×10^3	$\mathbf{7.792 \times 10^3}$ **(+24%)**
yeast	86s	**65s (+32%)**	$\mathbf{7.160 \times 10^6}$	5.815×10^7 (-712%)
aging	**202s**	696s (-245%)	$\mathbf{1.532 \times 10^7}$	5.510×10^8 (-3,496%)
school	4s	**2s (+100%)**	$\mathbf{3.765 \times 10^5}$	2.352×10^6 (-525%)
arxiv	**586s**	1,360s (-132%)	$\mathbf{6.178 \times 10^7}$	1.067×10^9 (-1,627%)
gallery	**6s**	14s (-133%)	$\mathbf{5.864 \times 10^5}$	1.432×10^7 (-2,341%)
Total	884s	2,137s (-142%)	$\mathbf{8.523 \times 10^7}$	1.693×10^9 (-1,886%)

Table 4.7: Average execution time to align two networks using DGDVs or GoTs. We compute the time of aligning a network with each of its randomized versions. Since each network randomization has ten noise levels, each comprised of five networks, we consider a total of 50 alignments per network, and average out the times.

Network	DGDVs	GoTs
zebra	**4.50s**	5.36s (-19%)
yeast	**55.32s**	82.46s (-49%)
aging	2,264.92s	**1,209.58s (+87%)**
arxiv	416.40s	**249.98s (+67%)**
gallery	20.36s	**19.70s (+3%)**
school	20.46s	**14.72s (+39%)**
Total	2,781.95s	**1,586.79(75%)**

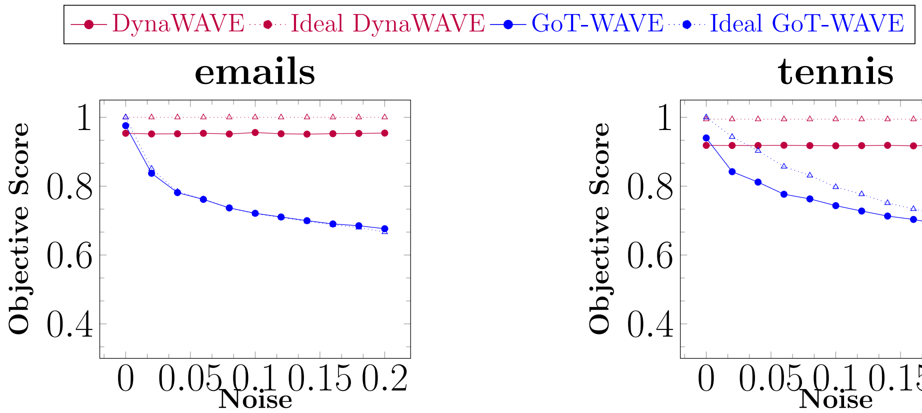


Figure 4.4: Comparison between GoT-WAVE and DynaWAVE on directed networks in terms of how well their alignments' objective scores match the objective scores of ideal alignments, when only node conservation is optimized ($\alpha = 0$). The noisy versions are generated using the pure directed randomization scheme (i.e., time stamps of two events are never swapped, only the edge direction is swapped). We observe, on one hand, as expected, that DynaWAVE does not distinguish between the original and its randomized versions because DGDVs do not take edge direction into account. On the other hand, GoT-WAVE clearly distinguishes between the original and its randomized versions because GoTs take edge direction into account.

4.7.2.3 Performance on real directed networks

Figure 4.4 shows objective score results for the pure directed randomization scheme. As expected, since this scheme only rewires edge direction, the original network and the noisy networks have identical topology when ignoring edge directions. Because of this, and because DGDVs are undirected, DynaWAVE can not differentiate between the networks, while GoT-WAVE can, since GoTs accounts for edge directions. The rest of this section focuses on the other, directed randomization scheme, where not only edge directions but also time stamps are rewired.

Unlike previous sections, we first address node correctness and only then objective score. We choose this organization because, on directed networks, we do experiments with different sets of GoTs (i.e., 4-node undirected GoTs, 3-node directed GoTs, and 4-node directed GoTs) in an effort to find the best set. For simplicity, we choose the best GoTs as those with the highest node correctness when aligning the original network to a noiseless version for $\alpha = 0$. We find that 3-node directed GoTs are the best for both of the directed networks (Table 4.8). Thus, henceforth, we use 3-node directed GoTs (for DGDVs, we still use four nodes and six events, as recommended by the DGDV authors).

Table 4.8: Node correctness when aligning a directed network to itself (noise = 0), for $\alpha = 0$. In parentheses, we show relative improvement (positive gain) or degradation (negative gain) in performance of GoT-WAVE compared to DynaWAVE. Node correctness results over all noise levels, using the best GoT-WAVE version (3-node directed GoTs), are shown in Figure 4.5 (a) and (b) for $\alpha = 0$ and $\alpha = \frac{1}{2}$, respectively.

Network	DynaWAVE	GoT-WAVE		
	Undirected-4	Undirected-4	Directed-3	Directed-4
emails	**0.85** ± 0.02	0.81 ± 0.03	0.83 ± 0.01 (-2%)	0.81 ± 0.02
tennis	0.74 ± 0.01	0.84 ± 0.03	**0.85 ± 0.02 (+15%)**	0.81 ± 0.02

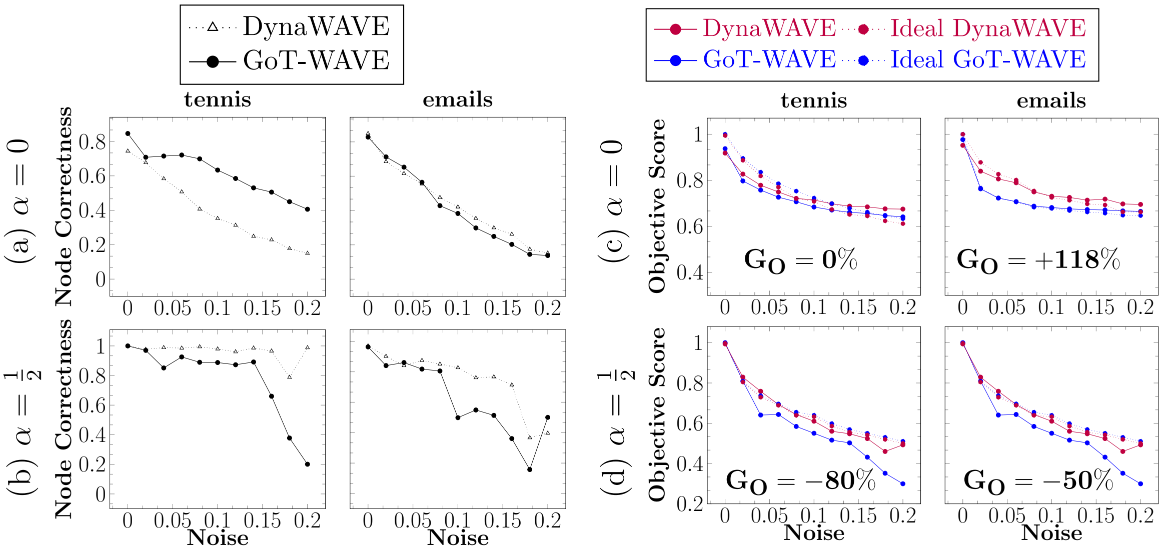


Figure 4.5: Comparison between GoT-WAVE and DynaWAVE on directed networks in terms of (a,b) node correctness and (c,d) how well their alignments' objective scores match the objective scores of ideal alignments, when (a,c) only node conservation is optimized ($\alpha = 0$) and (b,d) both node and edge conservation are optimized ($\alpha = \frac{1}{2}$). For panels (a,b), the higher the node correctness value, the better the method. For panels (c,d), recall that G_O is the relative gain of GoT-WAVE over DynaWAVE (positive: GoT-WAVE is superior; negative: DynaWAVE is superior).

In terms of node correctness, for $\alpha = 0$, we observe that GoT-WAVE has higher correctness than DynaWAVE for **tennis** over noise levels, and overall comparable node correctness for **emails**, depending on the noise level (Figure 4.5 (a)). We hypothesize that GoT-WAVE's performance depends on subgraph overlap between (consecutive) snapshots of the input network. Subgraph overlap is expected to be higher in the **tennis** network than in the **emails** network, because tennis players tend to have the same opponents every year, while one might not necessarily email the same people in different time periods. Indeed, these are exactly the trends that our two networks show. The same trend was already observed for an alternative email network [CM18].

Networks with low subgraph overlap such as our **emails** network have fewer transitions (i.e., lower GoTs frequencies), and thus provide less information to GoT-WAVE. For $\alpha = \frac{1}{2}$, DynaWAVE's node correctness is higher for both networks over most noise levels (Figure 4.5 (b)).

In terms of the objective score, for $\alpha = 0$, GoT-WAVE more closely matches its ideal alignments than DynaWAVE does for **emails**, and the two are comparable for **tennis** (Figure 4.5 (c)). For **tennis**, GoT-WAVE mismatches the ideal alignments at lower noise levels but matches them at higher noise levels, while DynaWAVE mismatches the ideal alignments at both lower and higher noise level. For $\alpha = \frac{1}{2}$, DynaWAVE's performance is again better for both networks (Figure 4.5 (d)).

In terms of running time, results are qualitatively similar to those for undirected networks (Table 4.9).

Table 4.9: Results on directed real networks in terms of feature extraction times when using DGDVs or GoTs. For DGDVs, we extract dynamic graphlets with up to four nodes and up to six events, as suggested in [HCM15]. For GoTs, we extract undirected GoTs with up to four nodes, directed GoTs with up to three nodes, and directed GoTs with up to four nodes. In parentheses, we show relative improvement (positive gain) or degradation (negative gain) in performance of GoT-WAVE compared to DynaWAVE. In bold, we show the best result for each network.

Network	DGDVs	GoTs		
	Undirected-4	Undirected-4	Directed-3	Directed-4
tennis	29.09s	113.07s (-289%)	**2.90s (+903%)**	128.43s (-341%)
emails	5.19s	5.65s (-9%)	**0.21s (+2,371%)**	7.84s (-51%)
Total	34.28s	118.72s (-246%)	**3.11s (+1,002%)**	136.27s (-297%)

4.8 Summary

We present GoT-WAVE as a new algorithm for temporal GPNA. Our results suggest that GoTs are an efficient measure of dynamic node conservation. While DynaWAVE benefits more from also optimizing dynamic edge conservation, only GoT-WAVE can support directed edges.

Future work on better incorporating dynamic edge conservation into GoT-WAVE may yield further improvements. Also, GoTs could be used under newer alignment strategies instead of WAVE. Further, on real networks, each of GoTs and DGDVs is superior half the time and the two dynamic node conservation measures are thus complementary. So, a deep understanding of each measure's (dis)advantages could perhaps guide development of a new, improved measure. As more temporal real data continue to become available, which is inevitable, dynamic network analyses, including temporal GPNA, will continue to gain importance.

Chapter 5

Node ranking

Finding important nodes in a network can be useful for different tasks, such as detecting potential points of failure or spreaders in social networks. Node centrality measures have been widely used for this purpose. However, finding dominant nodes is not necessarily the same as finding central nodes.

Our aim in this chapter is twofold.

First, we develop a measure of node dominance based on graphlets, which we name graphlet dominance (GD). Our aim is to showcase that GD is superior to node centrality measures when assessing node dominance. We test our hypothesis on a sports network and on an author citations network.

Second, instead of relying purely on network topology, we develop OTARIOS (OpTimizing Author Rankings using Insiders/Outsiders Subnetworks) which is a measure also based on topology but which uses additional features to better assess node importance.

In both cases, we measure the importance of nodes from the networks and produce node rankings. Our aim is to produce node rankings that are closer to ground-truth rankings that similar state-of-the-art algorithms.

5.1 Graphlet dominance (GD)

A simple way to measure node dominance is to compare the node's out-degree (i.e., its *dominant* edges) with its in-degree (i.e., its *dominated* edges). Using graphlet terminology: a node is dominant if it appears *many* more times in orbit 0 than in orbit 1 (Figure 3.3). For instance: (i) a high-ranked tennis player "beats" (dominates) more players than the ones that "beat" him (is dominated), or (ii) an important researcher "is referenced" (dominates) by more researchers than the ones he "references" (is dominated).

In most real-world cases, a node is not connected to all other nodes, thus, there is no direct dominance relation for all pairs of nodes. However, it is possible to extrapolate dominance to indirect relations. Consider graphlet G_2 from Figure 3.3: the node in orbit 2 dominates the node in orbit 3, and the node in orbit 3 dominates the node in orbit 4. For simplicity, henceforth we simply say that "orbit x dominates orbit y". While there is no direct connection from orbit 2 and orbit 4, there is a path from orbit 2 to orbit 4. Thus, orbit 2 indirectly dominates orbit 4. Since paths can have different lengths, we say that "orbit x i-dominates orbit y". In this case, orbit 2 2-dominates orbit 4.

Graphlets give more information about node dominance than just directed paths. Consider graphlets G_5 and G_6 from Figure 3.3: orbit 8 dominates orbit 9, and orbit 10 dominates orbits 11 and 12. Orbit 9 consists of two topologically equivalent nodes, while orbits 11 and 12 consist of a single node each. Thus, both orbit 8 and orbit 10 dominate two nodes. However, arguably, orbit 10 is a *more dominant orbit* than orbit 8 because orbit 10 dominates orbit 12 both directly and indirectly, i.e., orbit 10 dominates other dominant orbits, while orbit 8 only dominates two nodes directly. Similarly, orbit 12 is an dominated orbit to orbit 9.

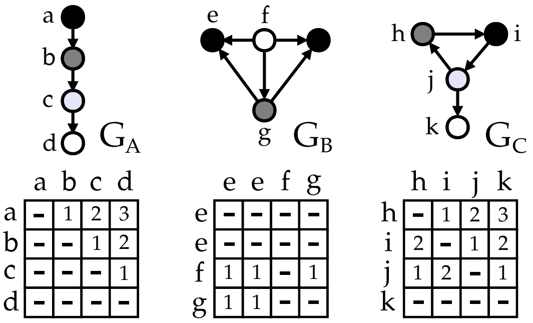
To be more general, our measure considers all k-node graphlets.

Computing k-graphlets does not guarantee that we find a dominance relation between every two nodes $x, y \in \mathcal{V}(G)$ since the minimum path length between x and y might be bigger than k. However, many real-world networks are small-world [WS98], thus even small graphlets, such as 3- or 4-node graphlets, find connected graphs for most pairs of nodes. Furthermore, subgraph counting for larger graphlets is very computationally expensive (Section 2.2). For these reasons, we use directed graphlets with at most four nodes.

5.1.1 Methodology

First, our graphlet-based ranking method receives as input a set of graphlets and calculates scores for their orbits. Then, during subgraph enumeration, we increase (or decrease) the nodes' scores according to the orbits that they appear at in the graphlet occurrences. Finally, we compare the scores and produce a node ranking.

We calculate orbit scores using the transitive closure of the graphlet, as shown in Figure 5.1, where distance $d(n_i, n_j)$ is the minimum path length between nodes n_i and n_j. Note that multiple nodes of a graphlet can be in the same orbit (e.g., orbit e from subgraph G_B), and those nodes have the same distance to all other nodes from different orbits (e.g., orbits f and g have distance 1 to both nodes in orbit e).



	a	b	c	d
a	-	1	2	3
b	-	-	1	2
c	-	-	-	1
d	-	-	-	-

	e	e	f	g
e	-	-	-	-
e	-	-	-	-
f	1	1	-	1
g	1	1	-	-

	h	i	j	k
h	-	1	2	3
i	2	-	1	2
j	1	2	-	1
k	-	-	-	-

Figure 5.1: Graph transitivity of 3 subgraphs. Nodes with the same shade are in the same orbit. Orbit scores are assessed using the transitivity matrix: row values are *positive* points while column values are *negative*. Higher cell values mean that the connection is less direct.

The three examples from Figure 5.1 show some aspects that our method captures and takes into account when scoring orbits:

- Orbit f from G_B is *dominant* since it has three out-edges and no in-edges, while orbit e is *dominated* since it has no out-edges and two in-edges. Orbit g stands somewhere in between since it dominates e but is dominated by f.
- Orbits a, b, c and d from G_A constitute a 4-node chain where the orbits at its start are more dominant than the ones at its end since they indirectly dominate more orbits (e.g., orbit a dominates b, 2-dominates c, and 3-dominates d).
- Orbits h, i, and j of G_C form a cycle and would, therefore, be considered equivalent if orbit k was not considered. However, orbit j dominates k directly while h and i dominate k indirectly. Also, orbit i dominates orbit k more directly than orbit h does.

$$S(o) = \left(\lambda \times \sum_{o_i \in \mathcal{I}(o)} \beta^{k-d(o,o_i)}\right) - \left((1-\lambda) \times \sum_{o_j \in \mathcal{S}(o)} \beta^{k-d(o_j,o)}\right) \tag{5.1}$$

We calculate orbit scores $S(o)$ for a given graphlet H as shown in Equation 5.1. For each orbit $o \in \mathcal{O}(H)$, we subtract the negative dominance points $\left(\sum_{(o_j \in \mathcal{S}(o))} \beta^{k-d(o_j,o)}\right)$ from the positive ones $\left(\sum_{(o_i \in \mathcal{I}(o))} \beta^{k-d(o,o_i)}\right)$. Set $\mathcal{I}(o) \in \mathcal{O}(H)$ is the set of orbits *dominated* to orbit o while $\mathcal{S}(o) \in \mathcal{O}(H)$ is the set of orbits *superior* to it. The *distance* between o_i and o_j is given by $d(o_i, o_j)$ and it can be at most $k-1$, where k is the size of the subgraph. Equation 5.1 gives higher relevance to *direct dominant connections* than to *indirect dominant connections* and, conversely, *direct dominated connections* take more points away than *indirect dominated connections.*

Parameter β controls the relative importance of the *directness*, i.e., a small β (closer to 1) gives roughly the same importance to direct and indirect dominances, while a high β means that direct dominances are more important. Notice that if β is too big the score becomes almost equivalent to the degree since distant dominances are almost irrelevant. Parameter $\lambda \in [0, 1]$ controls the relative importance of *dominating* versus *not being dominated*. Using $\lambda \approx 1$ means that a node is evaluated mostly by how many nodes he dominates (out-edges) while the amount of nodes that dominate him (in-edges) does not have a big impact in the rankings, and vice-versa when $\lambda \approx 0$, i.e., the node is ranked higher if he is dominated by few nodes. These parameters produce a flexible scoring mechanism.

$$S(v) = \sum_{o \in \mathcal{O}} Fr(v, o) \times S(o) \tag{5.2}$$

We calculate node scores $S(v)$ for all nodes $v \in \mathcal{V}(G)$ as shown in Equation 5.2. First, we perform subgraph counting on the network, listing the occurrences and the orbits where the nodes appear in each occurrence. Then, for each orbit o, $S(v)$ is increased by the score of the orbit $S(o)$ multiplied by the number of times v appeared in orbit o, represented by $Fr(v, o)$.

Finally, nodes are ordered from the lowest to the highest score to produce the ranking.

5.2 Comparison with node centrality measures

Here we propose a measure of node dominance. As far as we know, there is no directly comparable measure in the literature. However, node centrality measures, similarly to our node dominance measure, evaluate how important a node is in a network. Dozens of node centrality measures have been proposed [LCR+16, OFP+18], with different definitions of what makes a node important/central.

Some measures, such as the degree centrality (DC), relate the node's importance to its direct neighborhood, i.e., important nodes have many connections. Others, such as the closeness centrality (CC), betweenness centrality (BC), and subgraph centrality (SC), relate the node's importance to the paths that traverse it, i.e., important nodes have many paths traversing them. A node has (i) high CC if it is close to all other nodes in the network (i.e., it has a small average shortest path length), (ii) high BC if it is a shortcut between many pairs of nodes (i.e., many nodes have shortest paths to other nodes that traverse it), and (iii) high SC if it is part of many small groups (i.e., it is part of many small closed-loops, such as triangles or squares, e.g., G_4 and G_{31} from Figure 3.3). These four measures have a common aspect: they are computed individually for each node in the network. Iterative refinement measures, such as PageRank (PR), compute centrality scores on step s and update them on step $s + 1$. A node has high PR if it is important and also connected to important nodes (i.e., if it has many connections to nodes with many connections themselves). We focus on these five measures since they are some of the most widely used. [LCR+16] presents a complete survey on subgraph centrality algorithms.

Strong correlations have been found between node centrality measures [WS03, OFP+18]. Here we analyze how our own measure, graphlet dominance (GD), correlates with the five centrality measures (DC, BC, CC, SC, and PR) on a set of networks (Table 5.1), in an effort to show how our measure differs from the state-of-the-art.

Name	**Node**	**Edge**	$\lvert\mathcal{V}(\mathbf{G})\rvert$	$\lvert\mathcal{E}(\mathbf{G})\rvert$	**Source**
celegans	Neuron	Synapse	297	2,345	[New13]
tennis	Player	Beats	876	7,789	Our own
polblogs	Blog	Links	1,224	19,025	[New13]
goodreads	Writer	Influences	1,420	4,245	Our own

Table 5.1: Set of directed networks used to compare centrality measures.

Our GD measure has three parameters: λ, β, and k. We use two values for λ: 1 and $\frac{1}{2}$; $\lambda = 1$ only considers dominant edges and ignores dominated edges, and $\lambda = \frac{1}{2}$ gives equal importance to both. With this variation in λ we intend to show that considering dominated edges greatly affects the node scores and that other measures do not use this information. We use two values for β: 1 and $\frac{3}{2}$; $\beta = 1$ gives equal importance to direct and indirect dominance, and $\beta = \frac{3}{2}$ gives more importance to direct dominance. With this variation in β we intend to show that other measures do not account for direct/indirect dominance. Finally, we use three values for k: 2, 3, and 4. With this variation we intend to show that bigger graphlets capture different information that smaller ones, and that smaller graphlets are more similar to other node centrality measures than bigger graphlets. We create three major variants, $\text{GD}(\frac{1}{2}, \frac{3}{2}, k)$, $\text{GD}(\frac{1}{2}, 1, k)$, and $\text{GD}(1, \frac{3}{2}, k)$, and compare them with the five centrality measures previously discussed using Pearson correlations.

We first analyze the variant of GD that takes into account (i) both dominant and dominated connections and (ii) distinguishes between direct and indirect dominance, i.e., $\text{GD}(\frac{1}{2}, \frac{3}{2}, k)$. From Table 5.2 we observe that, for most cases, path-based measures (i.e., BC, SC, and CC) are weakly correlated with GD. This is expected since these measures give high scores to nodes with many paths traversing them, while GD lowers the score of nodes with many incoming edges. We consistently observe that the correlation between DC and GD decreases as k increases. Again, this is expected since $\text{GD}(\frac{1}{2}, \frac{3}{2}, 2)$ only considers 2-node subgraphs, like the degree. The correlation is not very high because of the way GD handles dominated connections. PR is, for most cases, the centrality measure more correlated with GD. Unlike other measures, and like GD, PR is more suited to find important nodes than central nodes, hence the higher correlation is also expected. Nevertheless, it is clear from these tests that GD captures different information from the centrality measures, and the differences increase for larger graphlets.

	celegans			**polblogs**			**tennis**			**goodreads**		
	k=2	k=3	k=4	k=2	k=3	k=4	k=2	k=3	k=4	k=2	k=3	k=4
DC	0.36	0.17	0.11	**0.31**	0.20	0.14	0.72	0.67	0.64	**0.70**	**0.75**	**0.70**
BC	0.03	0.04	0.03	-0.19	-0.26	-0.28	0.52	0.52	0.53	0.24	0.29	0.27
SC	0.33	0.18	0.12	0.21	0.13	0.08	0.59	0.61	0.63	0.18	0.16	0.14
CC	**0.40**	**0.23**	**0.17**	0.18	0.11	0.09	0.23	0.18	0.17	0.37	0.36	0.33
PR	**0.40**	0.19	0.13	0.29	**0.22**	**0.18**	**0.78**	**0.74**	**0.72**	0.41	0.42	0.40

Table 5.2: Comparison between $\text{GD}(\frac{1}{2}, \frac{3}{2}, k)$ and node centrality measures.

We now analyze how GD changes when it considers (i) only dominant connections, ignoring dominated connections, and (ii) distinguishes between direct and indirect dominance, i.e., $\text{GD}(1, \frac{3}{2}, k)$. From Table 5.3 we observe that, for most cases, the correlation between GD and the other measures increases. This happens because now GD, like the other measures, is also not considering the weight of dominated connections. The correlation of DC and GD with $k = 2$ now is almost 1, thus there is little gain of using this variant of GD over DC. As we increase k, GD becomes less correlated with the degree, particularly for larger networks.

	celegans			**polblogs**			**tennis**			**goodreads**		
	k=2	k=3	k=4	k=2	k=3	k=4	k=2	k=3	k=4	k=2	k=3	k=4
DC	**0.98**	**0.92**	**0.86**	**0.93**	**0.96**	**0.94**	0.94	0.89	0.87	**0.96**	**0.92**	**0.86**
BC	0.51	0.58	0.58	0.36	0.54	0.63	0.67	0.68	0.70	0.45	0.43	0.39
SC	0.57	0.54	0.50	0.62	0.63	0.57	0.67	0.71	0.74	0.13	0.11	0.09
CC	0.65	0.52	0.44	0.52	0.50	0.49	0.46	0.40	0.38	0.31	0.28	0.25
PR	0.65	0.55	0.47	0.64	0.71	0.73	**0.95**	**0.92**	**0.90**	0.45	0.41	0.38

Table 5.3: Comparison between GD$(1, \frac{3}{2}, k)$ and node centrality measures.

Finally, we analyze how GD changes when it considers (i) both dominant and dominated connections and (ii) does not distinguish between direct and indirect dominance, i.e., GD$(\frac{1}{2}, 1, k)$. From Table 5.4 we observe that this variant produces very similar results to GD$(\frac{1}{2}, \frac{3}{2}, k)$, but produces slightly higher correlations to other centrality measures. This indicates that direct/indirect dominance adds some information but so much as dominant/dominated connections.

	celegans			**polblogs**			**tennis**			**goodreads**		
	k=2	k=3	k=4	k=2	k=3	k=4	k=2	k=3	k=4	k=2	k=3	k=4
DC	0.36	0.19	0.14	**0.31**	0.20	0.15	0.72	0.69	0.67	**0.70**	**0.76**	**0.70**
BC	0.03	0.04	0.04	-0.19	-0.26	-0.27	0.52	0.53	0.55	0.24	0.31	0.27
SC	0.33	0.21	0.15	0.21	0.13	0.09	0.59	0.63	0.66	0.18	0.17	0.15
CC	**0.40**	**0.25**	**0.20**	0.18	0.12	0.10	0.23	0.20	0.19	0.37	0.38	0.36
PR	**0.40**	0.23	0.16	0.29	**0.23**	**0.20**	**0.78**	**0.76**	**0.75**	0.41	0.45	0.42

Table 5.4: Comparison between GD$(\frac{1}{2}, 1, k)$ and node centrality measures.

Table 5.5 complements the results previously discussed as is included for completeness: we correlate the fuller version of GD, GD$(\frac{1}{2}, \frac{3}{2}, 4)$, with other variants.

	k	**celegans**	**polblogs**	**tennis**	**goodreads**
GD$(\frac{1}{2}, \frac{3}{2}, k)$	2	0.816	0.851	0.944	0.826
	3	0.992	0.983	0.993	0.982
	4	1	1	1	1
GD$(\frac{1}{2}, 1, k)$	2	0.816	0.851	0.944	0.826
	3	0.986	0.978	0.990	0.983
	4	0.999	0.999	0.996	0.997
GD$(1, \frac{3}{2}, k)$	2	0.115	0.253	0.799	0.727
	3	0.079	0.180	0.880	0.829
	4	0.072	0.128	0.897	0.858

Table 5.5: Correlation between the complete GD variant with 4-node graphlets (i.e., GD$(\frac{1}{2}, \frac{3}{2}, 4)$) and simpler GD variants.

5.3 Tennis players ranking

In this section we apply our GD measure to rank tennis players. We analyze how GD produces different rankings by varying its parameters and interpret the results.

5.3.1 Motivation

Debating who is the best player (or team) is one of the most discussed topics in any competitive sport. Academia has proposed different ranking systems [Elo61, Mar02, Rad11], analyzed which variables are good indicators of form [SC88, Mel01, ML14], and tried to predict who will be successful [SH95, CGW99, VY09]. Here we focus on the first problem.

Objectively quantifying player achievements is not straightforward since multiple criteria can be used to compare players. Furthermore, the sports themselves evolve throughout the years, making it hard to compare players from different eras. Nevertheless, many competitive sports require a ranking system used for player seeding in tournaments.

Most official ranking systems by major sports organizations such as FIFA, ATP, etc., base their rankings on performances in tournaments (e.g., first-round elimination, Top-20, winner) over a certain time-frame (e.g., one year, five years) [Ste97]. We refer to these rankings as *tournament-based* since they also differentiate important tournaments (e.g., the FIFA World Cup, or Wimbledon) from minor events. Thus, in these rankings, a player (or team) is good if she/he wins many important tournaments.

Tournament-based approaches are problematic since (i) players' perception of tournaments evolve (e.g., the Australian Open was not considered a top tournament before the 1980s, thus many top players did not participate in it), (ii) the points given per round/tournament change, thus we can not compare points given in different seasons, and (iii) they give no weight to head-to-head results directly.

The approach that we follow is a *dominance-based* ranking system; here we are assuming that head-to-head results are more important than tournament performances. We do not look at head-to-head results solely because (i) not every pair of players has faced each other and (ii) analyzing the problem as a network gives more information about the intricate connections between the players.

The work by Radicchi et al. [Rad11] proposes a PageRank-like [PBMW99] ranking system for tennis players. Dingle et al [DKS12] also used Radicchi's ranking system to produce a more up-to-date ranking of both male and female tennis players. The network that Raddicchi et al. built is different from our own since a) their edges are *weighted* (w_{ij}: number of times that p_i beats p_j) while ours are *simple directed edges* reflecting win-percentages, b) they used match information from 1969 until 2010 whilst our networks are relative to matches from 1974 to 2015 and c) they only considered matches played on either Grand Slam tournaments or ATP Masters 1000 whereas we use information from all official ATP tournaments. Traditional PageRank does not decrease the node's rank with respect to its out-edges (in this case, meaning *loses against*) and is therefore not suitable to determine player dominance relations. The prestige score presented in [Rad11] lowers the w_{ij} according to p_j's out-degree (the number of times p_j loses against someone); therefore, dominating a *dominated* node gives less prestige than dominating a more *dominant* player. However, the prestige score is not decreased according to p_i's out-edges, which may result in *dominated* players having a high score as long as they dominate a few *dominant* players. Our scoring system increases the players' score in respect to the players that they dominate and, likewise, decreases their score when they are themselves dominated. Another approach was followed by Motegi and Masuda [MM12] where they use a dynamic win-ratio that takes into account temporal information and fluctuations in the ranking. They not only consider direct wins and losses but also indirect ones, namely those corresponding to directed paths of size 3. Our work differs because we use subgraphs of size 4, which encapsulate more information than paths of size 3. Furthermore, we consider global dominance relations to obtain an earned ranking, while their work focuses on obtaining a temporal snapshot for a particular point in time and use it for prediction purposes.

5.3.2 Network description

In order to construct the dominance network we first collected the names of all tennis players that have been ranked, at least once, in the Top-100 of the ATP year-end rankings from 1974 until 2015 and then extracted their match information from Tennis Abstract[1]. Players below the Top-100 only enter a few major tournaments, thus we did not consider them. A total of 856 tennis players have been in the Top-100 in the time-interval chosen and they have played $\approx 140,000$ matches between themselves.

We observe that the amount of matches played annually increased significantly in the

[1] www.tennisabstract.com

1990s but has dropped in recent years (Figure 5.2). This is possibly due to changes in the ranking system that encourage players to only participate in the most prestigious tournaments and also due to an increased awareness of the sport's physical demands. Nowadays, most tennis tournaments are contested on clay or hard courts, with only a few matches played on grass each year. Carpet was a popular surface until the mid-1990s but it was discontinued from the ATP Tour in the late 2000s. The surface characteristics affect the pace of the game, favoring different playing styles. Usually, grass is the *fastest* surface to play on, followed by carpet, hard and finally clay.

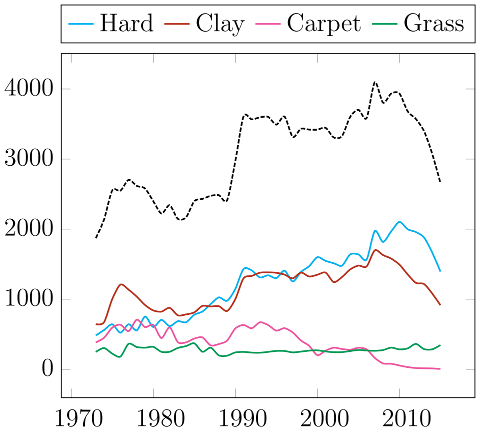


Figure 5.2: Matches played per year.

Table 5.6: Global statistics of the tennis networks, discriminated by surface.

Surface	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\frac{\lvert\mathcal{E}\rvert}{\lvert\mathcal{V}\rvert}$	$\frac{\lvert\mathbf{E}_{\Rightarrow}\rvert}{\lvert\mathbf{E}\rvert}$
Hard	301	868	2.88	0.64
Clay	289	793	2.74	0.65
Carpet	97	188	1.94	0.72
Grass	140	173	1.24	0.90
Overall	585	3279	5.61	0.68

After extracting match data, we create match tensors $M(s) = |\mathcal{P}| \times |\mathcal{P}| \times |\mathcal{Y}|$ and win-ratio tensors $W(s) = |\mathcal{P}| \times |\mathcal{P}| \times |\mathcal{Y}|$, one for each surface s and one for overall matches (thus, five match tensors and win-ratio tensors in total), where $\mathcal{P}$ is the set of player and $\mathcal{Y}$ is the set of years. Thus, $m(s)_{i,j,t}$ is the number of matches player p_i played against p_j on surface s in year t, and $w(s)_{i,j,t}$ is the win-ratio of player p_i against p_j on surface s in year t.

Then, we create *dominance networks* where nodes are players and the orientation of the edges between two players depends on their head-to-head win-ratio on a given surface: we create an edge (p_i, p_j) if p_i won at least $\delta\%$ of the matches against p_j on surface s in a given year t (Equation 5.3) and if they played a minimum ϕ matches on surface s during their careers (Equation 5.4).

$$w(s)_{i,j,t} \geq \delta\% \tag{5.3}$$

$$\left(\sum_{t=1974}^{2015} m(s)_{i,j,t} \right) \geq \phi \tag{5.4}$$

We use $\delta = \frac{2}{3}$, meaning that one player *dominates* another if he has defeated him in more than 66% of their matches, and $\phi = 3$ on grass courts and $\phi = 5$ for all other surfaces (since grass matches are fewer), meaning that we only create an edge if they played at least 3/5 matches.

We build *career dominance networks* by calculating dominances using the career win-percentage, instead of yearly results (Table 5.6). The number of "overall" edges is not simply the sum of the edges from all surfaces since an overall dominance is established by playing a minimum ϕ matches on any surface (i.e., one player can dominate another in overall matches without having ϕ encounters with him in any particular surface). Notice that only 585 of the original 856 players are represented in the network since the others did not play the required ϕ matches against any other Top-100 player, and consequently have no edges.

Since we require the win-ratio to be $> \delta\%$ for a dominance relation to be established, the networks contain bidirectional (or *reciprocal*) edges, meaning that two players met in at least ϕ matches but neither one dominates the other. The *dominant* (unidirectional) edges and *non-dominant* (bidirectional) edges are represented as $\mathcal{E}_{\Rightarrow}$ and $\mathcal{E}_{\Leftrightarrow}$, respectively, where $\mathcal{E} = (\mathcal{E}_{\Rightarrow} \cup \mathcal{E}_{\Leftrightarrow})$.

5.3.3 Network analysis

In regard to career dominance networks, Jimmy Connors dominates the most other players overall (63), followed by Roger Federer (60), and Ivan Lendl (59). On individual surfaces, Roger Federer leads with 46 out-edges on hard courts and 17 on carpet, Guillermo Vilas with 37 on clay, and John McEnroe with 23 on carpet.

We present two visual representations of our networks in Figure 5.3.

Figure 5.3 (a) shows the giant component of the career dominance network, where edge color represents surface: blue for hard, brown for clay, green for grass, pink for carpet and black for overall. Node size grows with the number of out-edges, e.g., Roger Federer corresponds to the largest node since he has the most out-edges (132).

Figure 5.3 (b) shows the relations between all 25 players that have been ranked as the ATP Top-1 player. Edges represent overall dominance and line thickness reflects how unbalanced the relation is. Boris Becker, Andre Agassi and Roger Federer dominate the highest number of ATP Top-1 players (6), and Juan Carlos Ferrero is dominated the most (5). Only Björn Borg, Gustavo Kuerten, and Rafael Nadal are not dominated by any ATP Top-1 players. If surfaces are taken into account, Gustavo Kuerten and Björn Borg remain as the only players not dominated by any Top-1 since Novak Djokovic and Roger Federer dominate Nadal on hard and grass courts, respectively. In fact, if $\phi = 5$, no ATP Top-100 player dominates Björn Borg regardless of δ since no one that played at least 5 matches against him has a positive winning-ratio.

We notice that Jimmy Connors, one of the players with most out-edges, does not dominate any Top-1 player. The fact that he faced the other Top-1 players when they were closer to their prime than himself might be the main reason for this. Comparing players from different eras might seem unfair if one inspects only individual relations but here we analyze the global scope of their careers, i.e., their head-to-head results against (i) players from their own era, (ii) players from the era preceding theirs, and (iii) players from the subsequent era. However, we note that this choice penalizes players at the ends of the time-interval considered.

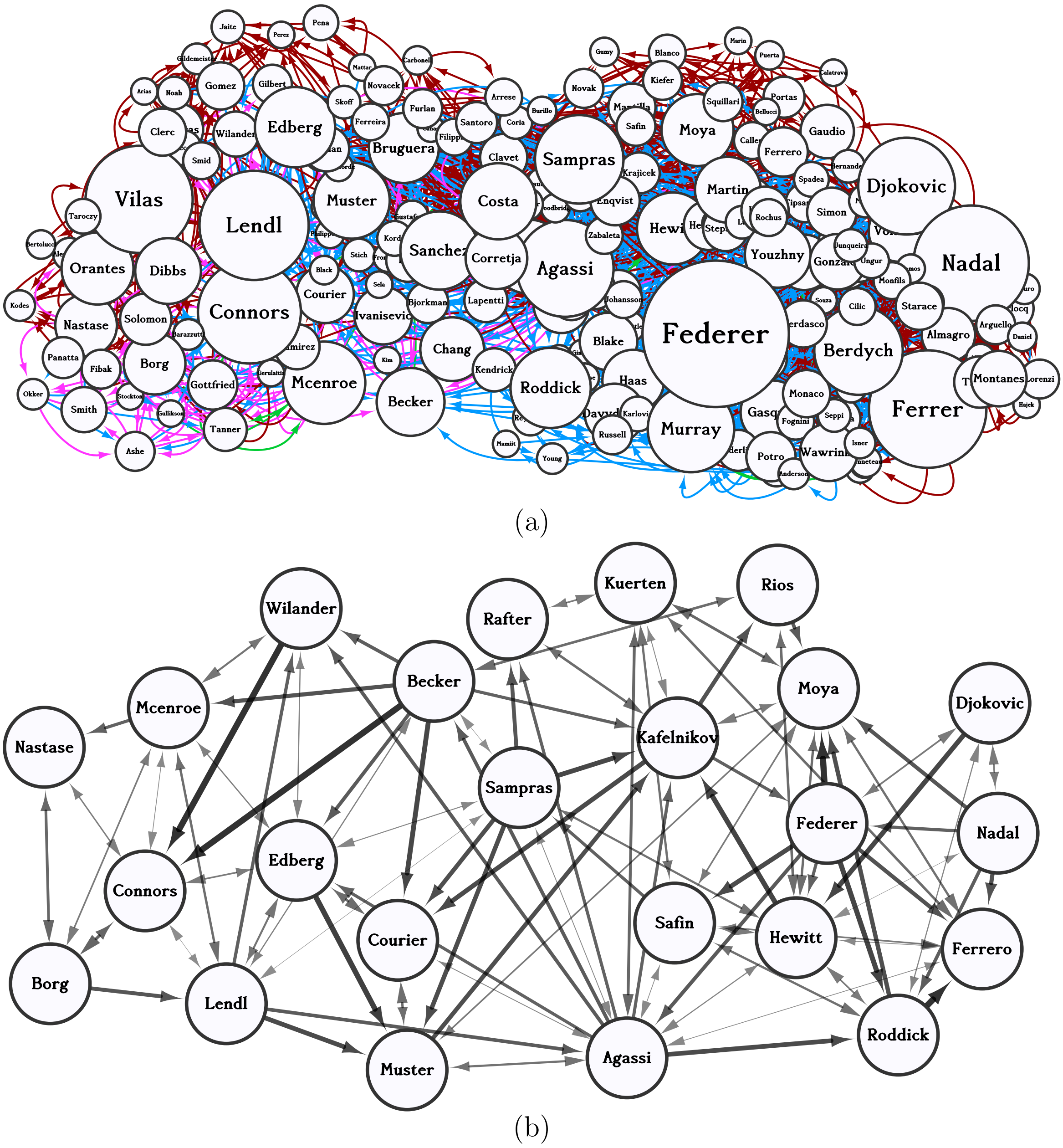
5.3.4 Results

We use our GD measure presented in Section 5.1 to rank tennis players and analyze the rankings obtained. Since the network is relatively small and it is easy to analyze the results, we are more focused on interpreting the different rankings produced by GD (depending on the parameters) than in evaluating our rankings against a ground-truth ranking (such as the ATP ranking). Nevertheless, in our analysis, we correlate our results with indicators of performance, such as the number of Grand Slam tournaments won. In all results, we use 4-node directed graphlets.

Table 5.7 presents the 15 players with highest scores depending on λ. In the middle column $\lambda = \frac{1}{2}$, meaning that *dominating* and *not being dominated* is equally important for the players' scores, and this value is used for comparison with the other λ.

When $\lambda < \frac{1}{2}$, our measure gives more importance to *not being dominated* that to *dominating* other players. Players such as Björn Borg and Gustavo Kuerten benefit from this parameter choice whereas Guillermo Vilas is penalized. For $\lambda \approx 0$, Rafael Nadal tops the ranking because very few players have a positive win-ratio against him. However, for very low λ, the GD ranking is not very meaningful since players that have only a few out-edges unrealistically climb in the rankings if they also have few in-edges.

When $\lambda > \frac{1}{2}$, our measure gives more importance to *dominating* that to *not being dominated* other players. Players such as Carlos Moya and Guillermo Vilas benefit from this parameter choice whereas Björn Borg and Novak Djokovic are penalized. Having $\lambda \approx 1$ still produces meaningful results since the ranking very highly play-



(a)

(b)

Figure 5.3: Tennis dominance networks: in (a) blue edges are drawn for dominances in hard courts, brown for clay, green for grass and pink for carpet. The nodes' size increases proportionally with their out-degree. (b) shows the relations between all ATP Top-1 players, disregarding surface.

ers that dominate many others is reasonable; however, it completely disregards the *dominated* edges.

We set $\lambda = \frac{1}{3}$ in our subsequent analysis, giving a slight edge to players that are not dominated by many others.

#	Player	⇳	Player	⇳	Player	Player	⇳	Player	⇳
1	I. Lendl	1 ▲	I. Lendl	1 ▲	R. Federer	R. Federer		R. Federer	
2	R. Federer	1 ▼	R. Federer	1 ▼	I. Lendl	I. Lendl		I. Lendl	
3	J. Connors		J. Connors		J. Connors	J. Connors		J. Connors	
4	R. Nadal	1 ▲	A. Agassi		A. Agassi	A. Agassi		A. Agassi	
5	N. Djokovic	1 ▲	R. Nadal		R. Nadal	R. Nadal		R. Nadal	
6	B. Becker		N. Djokovic	1 ▲	B. Becker	B. Becker		B. Becker	
7	A. Agassi	3 ▼	B. Becker	1 ▼	N. Djokovic	S. Edberg	1 ▲	G. Vilas	2 ▲
8	B. Borg	5 ▲	S. Edberg		S. Edberg	N. Djokovic	1 ▼	S. Edberg	
9	S. Edberg	1 ▼	J. McEnroe	1 ▲	G. Vilas	G. Vilas		N. Djokovic	2 ▼
10	P. Sampras	2 ▲	G. Vilas	1 ▼	J. McEnroe	J. McEnroe		J. McEnroe	
11	J. McEnroe	1 ▼	L. Hewitt		L. Hewitt	L. Hewitt		L. Hewitt	
12	A. Murray	4 ▲	P. Sampras		P. Sampras	P. Sampras		Y. Kafelnikov	2 ▲
13	L. Hewitt	2 ▼	B. Borg		B. Borg	Y. Kafelnikov	1 ▲	P. Sampras	1 ▼
14	G. Kuerten	7 ▲	A. Murray	2 ▲	Y. Kafelnikov	C. Moya	3 ▲	C. Moya	3 ▲
15	G. Vilas	6 ▼	A. Roddick		A. Roddick	B. Borg	2 ▼	D. Ferrer	2 ▲
	$\lambda = \frac{1}{6}$		$\lambda = \frac{1}{3}$		$\lambda = \frac{1}{2}$	$\lambda = \frac{2}{3}$		$\lambda = \frac{5}{6}$	

Table 5.7: Ranking obtained by varying λ: the relative weight between dominating (out-edges) and being dominated (in-edges).

We set $\beta = \frac{3}{2}$, giving more weight to direct dominances that to indirect ones. To illustrate the effect of β, consider graphlet G_A from Figure 5.1: if $\beta = 1$, $S(a) = 1^{(4-1)} + 1^{(4-2)} + 1^{(4-3)} = 3$, $S(b) = 1$, $S(c) = -1$ and $S(d) = -3$; if $\beta = 2$, $S(a) = 2^{(4-1)} + 2^{(4-2)} + 2^{(4-3)} = 14$, $S(b) = 4$, $S(c) = -4$ and $S(d) = -14$. For instance, this means that orbits a and b give similar scores when $\beta = 1$ but very different scores when $\beta = 2$. Low β does not distinguish direct from indirect dominances while high β penalizes indirect dominances too heavily, therefore we chose an intermediate value.

According to our ranking, Roger Federer is the most dominant player, followed by Jimmy Connors and Ivan Lendl (Table 5.8 (a)). We observe that the number of Grand Slam tournaments won by the player is correlated with higher positions in our ranking. From the Top-25 players only David Ferrer, Tim Henman, and Robin Soderling did not win any, but all three of them were Top-4 ATP players during their careers.

Table 5.8 (b) shows our rankings by surface of all 25 players that have been the Top-1 ATP player from 1974 until 2015. A dash (–) means that the player does not have any edge on the dominance network for that particular surface, i.e., he did not play the minimum ϕ matches against anyone. The position of the player is presented in bold-face if he is ranked among the Top-25. We observe that most (76%) ATP Top-1 players are also ranked as one of the Top-25 most dominant players by our measure; the exceptions are John Newcombe, Mats Wilander, Jim Courier, Marcelo Rios, Patrick Rafter, and Marat Safin. Patrick Rafter is a notable outlier since he is ranked at the

#	Player	#GS
1	R. Federer	17
2	J. Connors	8
3	I. Lendl	8
4	A. Agassi	8
5	R. Nadal	14
6	J. McEnroe	7
7	G. Vilas	4
8	N. Djokovic	10
9	B. Becker	6
10	P. Sampras	14
11	S. Edberg	6
12	A. Roddick	1
13	A. Murray	2
14	L. Hewitt	2
15	B. Borg	11
16	T. Muster	1
17	C. Moya	1
18	I. Năstase	2
19	D. Ferrer	0
20	G. Kuerten	3
21	Y. Kafelnikov	2
22	A. Ashe	3
23	JC. Ferrero	1
24	T. Henman	0
25	R. Soderling	0

(a)

Player	Overall	Hard	Clay	Grass	Carpet
I. Nătase	**18**	26	**9**	–	**18**
J. Newcombe	38	–	–	128	38
J. Connors	**2**	**11**	27	**2**	**4**
B. Borg	**15**	31	**7**	**12**	**7**
J. McEnroe	**6**	27	297	**4**	**1**
I. Lendl	**3**	**10**	**10**	90	**3**
M. Wilander	27	234	**8**	96	78
S. Edberg	**11**	**12**	118	**7**	70
B. Becker	**9**	192	55	**6**	**2**
J. Courier	41	**13**	37	32	73
P. Sampras	**10**	**7**	66	**11**	**6**
A. Agassi	**4**	**3**	63	28	41
T. Muster	**16**	178	**3**	–	–
M. Rios	33	252	**20**	–	–
C. Moya	**17**	190	149	–	–
Y. Kafelnikov	**21**	269	321	**22**	49
P. Rafter	381	243	193	**20**	–
M. Safin	46	298	31	167	–
G. Kuerten	**20**	**21**	**5**	–	–
L. Hewitt	**14**	**8**	177	496	–
JC. Ferrero	**23**	231	**16**	–	–
A. Roddick	**12**	**5**	76	63	–
R. Federer	**1**	**1**	**14**	**1**	–
R. Nadal	**5**	**6**	**2**	118	–
N. Djokovic	**8**	**2**	35	**8**	–

(b)

Table 5.8: GD ranking of tennis players with $\lambda = \frac{1}{3}$, $\beta = \frac{3}{2}$, and $k = 4$: (a) GD's Top-25 players and (c) GD's ranking by surface of all Top-1 ATP players (ordered chronologically). #GS is the number of Grand Slam tournaments won by the player.

bottom half (381th out of 585 players). We note that he was only ranked as the ATP Top-1 for one week, the shortest period by any player.

Our GD measure detects surface specialists (e.g., Wilander, Muster, Rios, Kuerten, and Ferrero on clay, Courier, Agassi and Hewitt on hard courts, and Newcombe and Rafter on grass), all-round players (such as Năstase, Connors, and Federer) and players with an Achilles-heel on a specific surface (such as Sampras and Djokovic on clay, Borg on hard, and Lendl and Nadal on grass). We should note that Nadal has a low ranking on grass despite having a win-ratio of $\approx 79\%$ in that surface and winning two Grand Slams tournaments played on grass. His low score comes primarily from the fact that he is dominated by Roger Federer on that surface and, because Federer is a dominant hub-like node in grass, Nadal appears in many different subgraphs with Federer and the other players that Federer dominates. Since Nadal occupies a negative orbit in those subgraphs his score is continuously decreased. This negative effect is primarily felt on small and sparse networks such as the grass network where even a single connection has a very high impact.

Table 5.9 shows our Top-10 by surface and also the number of Grand Slam tournaments won by the player on that surface. Roger Federer is the most dominant player both on hard courts and grass, Guillermo Vilas on clay and McEnroe on carpet. Again, the number of Grand Slam victories is correlated with the ranking (note that no grand slam tournament was ever contested on carpet). Nadal is not the most dominant player on clay despite winning nine Grand Slams on that surface. This is because Vilas was a very prolific player in clay and won many tournaments there, dominating most players of his time on that surface.

#	Player	#GS
1	Federer	9
2	Djokovic	7
3	Agassi	6
4	Murray	1
5	Roddick	1
6	Nadal	3
7	Sampras	7
8	Hewitt	1
9	Berdych	0
10	Lendl	5

Hard

#	Player	#GS
1	Vilas	2
2	Nadal	9
3	Muster	1
4	Bruguera	2
5	Kuerten	3
6	Orantes	1
7	Borg	6
8	Wilander	3
9	Nătase	1
10	Lendl	3

Clay

#	Player	#GS
1	Federer	7
2	Connors	4
3	Edmondson	1
4	McEnroe	3
5	Tanner	1
6	Becker	3
7	Edberg	4
8	Djokovic	3
9	Cash	1
10	Sampras	7

Grass

#	Player
1	McEnroe
2	Becker
3	Lendl
4	Connors
5	Ivanisevic
6	Sampras
7	Borg
8	Ashe
9	Rosewall
10	Walts

Carpet

Table 5.9: GD ranking of tennis players with $\lambda = \frac{1}{3}$, $\beta = \frac{3}{2}$, and $k = 4$ by surface. #GS is the number of Grand Slam tournaments won by the player on that surface.

Finally, we analyze yearly rankings instead of career rankings (Table 5.10).

We observe that Federer dominates the most different years (6), followed by Connors and Edberg (5), and Lendl and Djokovic (4). Federer and Edberg dominate the most consecutive years (5), however, their GD scores are very different: Federer averages $\approx$ 360 points over the five years while Edberg averages $\approx$ 140 points. This indicates that Edberg's era was more balanced that Federer's.

When we separate the time-interval into three different eras of 30 years each (e.g., $Y_1 = [1974, 1988[$, $Y_2 = [1988, 2002[$, and $Y_3 = [2002, 1987[)$, we observe that the most dominant players of each era (as computed by our GD measure) have won 15 Grand Slams in Y_1, 11 in Y_2, and 28 in Y_3, i.e., the number of Grand Slams won by the dominant players has greatly increased in the last 30 years. Currently, tennis players are judged by the number of Grand Slams, which was not the case in earlier decades. Our results suggests that using Grand Slams to judge players' dominance is

	1974	**1975**	**1976**	**1977**	**1978**	**1979**	**1980**
Player	Connors	Ashe	Connors	Vilas	Connors	Connors	McEnroe
GD	224.4	165.3	236.8	445.6	184.7	453.5	180.1
#GS	3	1	1	2	1	0	1
ATP	#1	#4	#1	#2	#1	#2	#2

	1981	**1982**	**1983**	**1984**	**1985**	**1986**	**1987**
Player	Connors	Lendl	Lendl	McEnroe	Lendl	Lendl	Edberg
GD	122.1	307.0	149.7	150.0	146.7	152.4	118.0
#GS	0	0	0	2	1	2	1
ATP	#2	#3	#2	#1	#1	#1	#2

	1988	**1989**	**1990**	**1991**	**1992**	**1993**	**1994**
Player	Edberg	Edberg	Edberg	Edberg	Becker	Sampras	Sampras
GD	61.1	127.8	185.6	213.0	86.0	186.8	289.2
#GS	1	0	1	1	0	2	2
ATP	#5	#3	#1	#1	#5	#1	#1

	1995	**1996**	**1997**	**1998**	**1999**	**2000**	**2001**
Player	Agassi	Chang	Chang	Agassi	Agassi	Hewitt	Kuerten
GD	183.3	138.3	63.9	166.2	92.2	95.1	115.3
#GS	1	0	0	0	2	0	1
ATP	#2	#2	#3	#6	#1	#7	#2

	2002	**2003**	**2004**	**2005**	**2006**	**2007**	**2008**
Player	Hewitt	Federer	Federer	Federer	Federer	Federer	Nadal
GD	98.7	121.3	379.7	542.6	445.0	316.0	349.0
#GS	1	1	3	2	3	3	2
ATP	#1	#2	#1	#1	#1	#1	#1

	2009	**2010**	**2011**	**2012**	**2013**	**2014**	**2015**
Player	Nadal	Nadal	Djokovic	Federer	Djokovic	Djokovic	Djokovic
GD	249.5	276.0	261.5	216.7	275.9	229.2	119.8
#GS	1	3	3	1	1	1	3
ATP	#2	#1	#1	#2	#2	#1	#1

Table 5.10: GD ranking of tennis players with $\lambda = \frac{1}{3}$, $\beta = \frac{3}{2}$, and $k = 4$ by year. GD is the score of the player computed by our graphlet dominance measure, #GS is the number of Grand Slam tournaments won by the player on that surface, and ATP is the year-end ranking of the player in the ATP official rankings.

not adequate for older eras.

Finally, our results suggest that the ATP ranking system might have a lag in determining who the most dominant player is. We observe multiple cases where the most dominant player in a given year only becomes the ATP Top-1 in the following year: Edberg was the most dominant player in 1989 and then the ATP Top-1 in 1990, Agassi in 1998, Hewitt in 2000, Federer in 2003, Nadal in 2009, and Djokovic in 2013.

5.4 Scientific authors ranking

This section is divided in two main parts.

First, we use GD to rank scientific authors. GD is a topology based approach, thus it only uses network structure to produce the rankings. Furthermore, since a citation network is not a clear dominance network (i.e., the notion of dominance is not clear in the edges), GD is not as well-suited for this task as it is when ranking nodes in dominance networks, such as ranking tennis players.

Then, we develop a feature enriched topology approach, named OTARIOS, which is less general than GD since it targets citations networks specifically, but produces better results since it uses relevant information from the authors' productivity and citation information that topology based approaches ignore.

5.4.1 Motivation

Deciding scientific committees, research grants, or faculty promotions is still done mostly by peer review. Nevertheless, bibliometrics have been proposed that assist the peer review process [VCG14]. Bibliometrics typically rely on the author's productivity (i.e., statistics of author's papers) and the author's impact (i.e., statistics of author's citations) [B.V18], e.g., one of the most widely used bibliometrics is the author's h-index [Hir05], which measures the impact only of the author's most relevant works.

However, traditional bibliometrics have the drawback of only assigning impact to authors' direct citations, thus they ignore indirect citations and fail to capture the nature of scientific development since they disregard the fact that a new discovery is not solely due to previous work directly referenced.

To address this limitation, graph algorithms have been developed for citation networks [Din09, RFMV09, DV12, WJD+13]. These algorithms are modifications of PageRank [PBM+98] applied to citation networks. One of PageRank's major ideas is that not all nodes are equal, i.e., it is good to be referenced by any webpage but it is better to be referenced by important webpages. This idea is general and applicable to citation networks, i.e., it is important to be cited by important authors.

Citation networks have information about the authors, venues, and papers that give important meta-information beyond the network structure. Traditional PageRank is a *topology-based* measure, meaning that it only uses information from the network structure. Algorithms extending PageRank have been proposed specifically for citation networks that take some feature information into account, such as the productivity of the author [RFMV09], the recency of the paper [HCKW10, DV12], and the venue prestige [HCKW10]; these algorithms are referred to as *feature enriched topology* measures.

Our goal is to improve the ranking of scientists using these two approaches: (a) topology-based and (b) feature enriched topology.

5.4.2 Network description

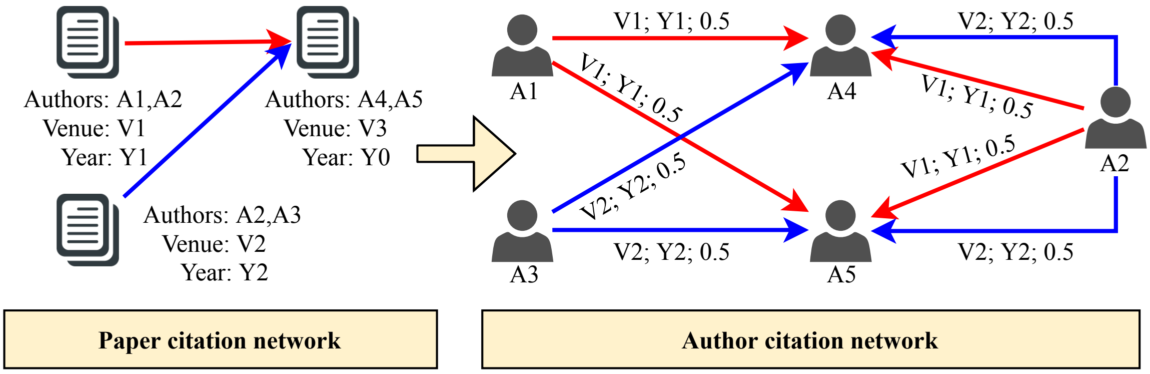


Figure 5.4: Creation of an author citation network.

Our data consists of paper citations of the form $(a_i, a_j, p_k, t_l, v_m)$, where authors $a_i, a_j \in \mathcal{A}$ and a_i cites a_j in paper $p_k \in \mathcal{P}$ published in year t_l in venue $v_m \in \mathcal{V}$.

There are two groups of author ranking methods: paper-level and author-level [WSC16]. On one hand, paper-level ranking uses the papers' citation network to diffuse scientific credit to cited papers, and then author credit is derived from the credit of his papers [HCKW10, DV12]. On the other hand, author-level ranking uses the authors' citation network to diffuse scientific credit to cited authors, thus the authors' credit is directly obtained [RFMV09, Din09, WJD+13]. In our case, we transform the paper citation network into an author citation network (Figure 5.4) and calculate all measures on the author citation network.

We create five networks, each consisting of publications in top-tier Computer Science conferences in different topics (Table 5.11).

Network	**Conferences**	$\lvert\mathcal{V}(\mathbf{G})\rvert$	$\lvert\mathcal{E}(\mathbf{G})\rvert$	$\frac{\mathcal{E}(G)}{\mathcal{V}(G)^2}$
CM	AAAI, IJCAI, ICML, ACL, ICCV, CVPR	41.5k	3.0M	0.002
TC	FOCS, SODA, STOC	5.6k	0.3M	0.010
NET	INFOCOM, NSDI, SIGCOMM, MOBICOM, SIGMETRICS	17.2k	1.5M	0.005
IS	KDD, CIKM, PODS, SIGMOD, VLDB, WWW, SIGIR	32.7k	2.8M	0.003
SE	PLDI, FSE, ICSE, OSDI, SOSP	12.4k	0.7M	0.005

Table 5.11: Set of networks used for experimental evaluation. Data was taken from [TZY+17, TZY+08]. The full DBLP dataset contains over 3M publications from 1936 to 2018. Each network is built using publications only from a set of conferences, e.g., network TC contains publications from FOCS, SODA and STOC.

To evaluate the methods, we need to compare their rankings with a ground-truth (more details in Section 2.5). For each network, we use a ground-truth based on peer-review using the best paper award information from every conference, i.e., each rewarded paper has a unit of prestige which is equally divided by its authors. Thus, each author has a certain ground-truth prestige that is the sum of the prestige of his awards. As a result, we are assuming that authors that have won more awards with fewer co-authors should be ranked higher.

In our experiments, we consider that methods that produce rankings more similar to the ground-truth ranking are better. For this purpose, for each network and for each method, we compare the method's predicted ranking with the network's ground truth using NDCG@5, NDCG@10, NDCG@20, NDCG@50, and NDCG@100.

5.4.3 Topology-based ranking

5.4.3.1 Results

Here we use GD (described in Section 5.1) to rank scientific authors and compare it with PageRank, since both are topology-based measures.

We set the damping factor of PageRank to its default value of 0.85 as described in [PBM+98].

Like we did for the tennis network in Section 5.3, we analyze how parameters λ, β, and k affect the rankings produced by GD. We performed parameter tuning on all five networks but only show the results for two of the networks for brevity (Tables 5.12, 5.13, and 5.14). Results of the best GD variant over all networks are shown at the end of the discussion (Table 5.15).

To analyze k, we set $\lambda = 1$ and $\beta = \frac{3}{2}$. Recall from Section 5.1 that this means that only dominant edges are considered (and not dominated edges) and that indirect dominances contribute less to the score of a node that direct dominances. We observe that 3-node graphlets have consistently and significantly better results (i.e., higher NDCG) that 2-node graphlets (Table 5.12). For most networks, 4-node graphlets did not have significantly better results and, since 4-node graphlets take exponentially longer to count, in the remaining tests we use 3-node graphlets.

GD(1, $\frac{3}{2}$, k)	**NDCG**						**NDCG**					
	@5	@10	@20	@50	@100	Mean	@5	@10	@20	@50	@100	Mean
$k = 2$	0,176	0,328	0,343	0,374	0,386	0,321	0,000	0,023	0,027	0,030	0,089	0,034
$k = 3$	**0,265**	0,346	**0,378**	0,394	0,437	0,364	**0,042**	**0,046**	**0,050**	**0,037**	**0,104**	**0,056**
$k = 4$	0,263	**0,376**	0,376	**0,413**	**0,440**	**0,374**	**0,042**	**0,046**	**0,050**	**0,037**	0,099	0,055
	(a) SE						(b) TC					

Table 5.12: Comparison of GD variants on networks (a) SE and (b) TC by varying k. For each GD variant we measure NDCG for the top-n ranked authors (@n), as well as NDCG's mean value. Bold highlights the highest score.

To analyze λ, we set $k = 3$ and $\beta = \frac{3}{2}$. Recall that when $\lambda = 0$, only dominated edges are considered. In the case of citation networks, this means that authors that cite many authors are heavily penalized. In practice this results in meaningless rankings (i.e., NCDG = 0) because dominant edges (i.e., received citations) are clearly important for an author's merit (Table 5.13). We observe that $\lambda = 1$ has higher NCDGs than $\lambda = \frac{1}{2}$, meaning that penalizing authors for citing too many other authors does not produce better rankings.

GD(λ, $\frac{3}{2}$, 3)	**NDCG**						**NDCG**					
	@5	@10	@20	@50	@100	Mean	@5	@10	@20	@50	@100	Mean
$\lambda = 0$	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
$\lambda = \frac{1}{2}$	0,176	0,314	0,308	0,345	0,319	0,292	**0,042**	0,033	0,049	**0,039**	0,101	0,053
$\lambda = 1$	**0,265**	**0,346**	**0,378**	**0,394**	**0,437**	**0,364**	**0,042**	**0,046**	**0,050**	0,037	**0,104**	**0,056**
	(a) SE						(b) TC					

Table 5.13: Comparison of GD variants on networks (a) SE and (b) TC by varying λ. For each GD variant we measure NDCG for the top-n ranked authors (@n), as well as NDCG's mean value. Bold highlights the highest score.

To analyze β, we set $k = 3$ and $\lambda = 1$. Recall that when $\beta = 1$, direct and indirect dominances are equally considered and, as β grows, indirect dominances are increasingly less considered. In our particular dataset, we do not observe significant differences for the values of β that we tested (Table 5.14).

GD(1, β, 3)	**NDCG** (a) SE						**NDCG** (b) TC					
	@5	@10	@20	@50	@100	Mean	@5	@10	@20	@50	@100	Mean
$\beta = 1$	**0,265**	0,343	0,375	**0,396**	0,434	0,363	**0,042**	**0,046**	**0,050**	0,037	**0,104**	0,056
$\beta = \frac{3}{2}$	**0,265**	**0,346**	**0,378**	**0,394**	**0,437**	**0,364**	**0,042**	**0,046**	**0,050**	0,037	**0,104**	0,056
$\beta = 2$	**0,265**	**0,346**	**0,378**	0,393	0,435	0,363	**0,042**	**0,046**	**0,050**	**0,043**	**0,104**	**0,057**

Table 5.14: Comparison of GD variants on networks (a) SE and (b) TC by varying β. For each GD variant we measure NDCG for the top-n ranked authors (@n), as well as NDCG's mean value. Bold highlights the highest score.

Finally, we compare the best variant of GD against PageRank (Table 5.15). GD outperforms PageRank in four of the five networks for most cases (i.e., predicting the Top-5, Top-10, etc.) and on average. PageRank outperforms GD significantly on the CM network. We hypothesize that this might be because CM is the sparsest network ($\frac{|\mathcal{E}(G)|}{|\mathcal{V}(G)|^2} < 0.002$), thus the number of graphlets found is relatively small, which does not negatively affect PageRank.

Network	**NDCG** (a) PageRank						**NDCG** (b) GD(1, $\frac{3}{2}$, 3)					
	@5	@10	@20	@50	@100	Mean	@5	@10	@20	@50	@100	Mean
CM	**0,119**	**0,153**	**0,129**	**0,253**	**0,250**	**0,181**	0,056	0,041	0,062	0,074	0,083	0,064
TC	0,000	0,000	0,013	**0,043**	**0,105**	0,032	**0,042**	**0,046**	**0,050**	0,037	0,104	**0,056**
NET	0,264	0,256	0,220	0,208	0,209	0,231	**0,364**	**0,315**	**0,295**	**0,245**	**0,254**	**0,294**
IS	0,085	0,159	**0,173**	0,225	0,240	0,177	**0,151**	**0,179**	0,170	**0,272**	**0,248**	**0,204**
SE	**0,291**	0,326	0,356	0,338	0,379	0,338	0,265	**0,346**	**0,378**	**0,394**	**0,437**	**0,364**

Table 5.15: Comparison of GD against PageRank on all five networks. For both GD and PageRank we measure NDCG for the top-n ranked authors (@n), as well as NDC's mean value. Bold highlights the highest score.

5.4.4 Feature enriched topology ranking

In this section we aim to improve topology based ranking by introducing additional features from the data. In the case of scientific author ranking, these features are related to the author's papers or the citations, such as the year and venue prestige.

In the previous section, using topology only, GD obtained better results than PageRank. However, in this section, we extend PageRank instead of GD to a feature enriched measure. We do this for several reasons: (i) the following tests (i.e., determining the best combination of features) are extensive and GD (namely subgraph counting) is much more computationally demanding than PageRank (see Section 2.2), (ii) this work preceded our analysis of GD in author ranking (where we see that GD is superior), and (iii) this is a joint work with Jorge Silva, who comes from the field of expert profiling and expert recommendation, and PageRank-based algorithms are state-of-the-art for author ranking, which is related to the aforementioned tasks.

Our PageRank-based method is named OTARIOS, from OpTimizing Author Ranking for Insiders/Outsiders Subnetworks. OTARIOS' efficiency comes from two aspects: (i) how it efficiently combines different features and (ii) how it divides the network into insider nodes and outsider nodes. For our purposes in this thesis, we are interested in analyzing only the first aspect; for details on how OTARIOS handles the second aspect, we refer the reader to OTARIOS original paper [SAS18].

We begin by introducing notation specific to the problem of ranking authors in a feature enriched network. Then, we present OTARIOS and discuss results.

5.4.4.1 Notation

For consistency, we denote sets by calligraphic letters (e.g., $\mathcal{S}$), elements of those sets (i.e., entities) by the respective capital letter with an index (e.g., $S_i \in \mathcal{S}$), attributes of entities (e.g., year, impact factor) as functions named in lower-case alphabetic or Greek letters (e.g., $a(S_i)$ or $\alpha(S_i)$), and constants as single Greek letters (e.g., τ). The cardinality of a given set S is denoted by $|\mathcal{S}|$.

Recency of a paper

$$\delta(P_j) = \left(\max_{P_{j'} \in \mathcal{P}} y(P_{j'}) \right) - y(P_j) \tag{5.5}$$

Recency of an author

$$\delta(A_i) = \min_{P_j \in \mathcal{P}_{A_i}} \delta(P_j) \tag{5.6}$$

Venue prestige

$$\lambda(V_k, y) = \frac{c(V_k, y)}{\displaystyle\sum_{x=1}^{3} p(V_k, y - x)} \tag{5.7}$$

Cited individuality

$$w(A_{i'} \to A_i, P_j) = \frac{1}{|\mathcal{A}_{P_j}|}, A_i \in \mathcal{A}_{P_j} \tag{5.8}$$

Citation recency

$$a(A_{i'} \to A_i, P_j) = e^{\frac{-\delta(P_j)}{\tau}}, A_{i'} \in \mathcal{A}_{P_j} \tag{5.9}$$

Citation prestige

$$v(A_{i'} \to A_i, P_j) = v(P_j), A_{i'} \in \mathcal{A}_{P_j} \tag{5.10}$$

Given a set of papers $\mathcal{P}$ published in a set of venues $\mathcal{V}$ by a set of authors $\mathcal{A}$, we want to find the n top-ranked authors. A paper $P_j \in \mathcal{P}$ is co-authored by authors $\mathcal{A}_{P_j} \subseteq \mathcal{A}$. Likewise, an author $A_i \in \mathcal{A}$ is (one of) the author(s) of papers $\mathcal{P}_{A_i} \subseteq \mathcal{P}$. We build citation networks $G = \{\mathcal{A}, \mathcal{E}\}$ where nodes represent authors and edges represent citations between authors, written as $A_{i'} \rightarrow A_i$.

Regarding node attributes, papers have metadata which we use as features, namely their publication year, venue prestige, and the number of references, represented by $y(P_j)$, $v(P_j)$ and $r_{out}(P_j)$, respectively. The *recency* of a paper, represented by $\delta(P_j)$, is given by Equation 5.5. Similarly, the *recency* of an author, represented by $\delta(A_i)$, is the recency of his most recent paper (Equation 5.6). The venue prestige of a paper P_j depends on the venue $V_k \in \mathcal{V}$ where it was published and the year when it was published, represented by $v(P_j) = \lambda(V_k, y(P_j))$. We estimate venue prestige with *CiteScore*[B.V18] (Equation 5.7), where $p(V_k, y)$ is the number of papers published in V_k in year y and $c(V_k, y)$ is the number of citations that all papers published in V_k in year y received. Thus, venues with many citations per paper have higher prestige.

Regarding edges, citation networks can be unweighted and simple, i.e., two papers (or authors) are connected by a single edge with weight equal to 1 [HCKW10, DV12], or weighted and multiple, i.e., two authors are connected by multiple edges with different weights. These multiple edges concern different edge attributes that depend on the publication P_j where author $A_{i'}$ cites author A_i. The recency of an edge, represented by $a(A_{i'} \rightarrow A_i, P_j)$, gives more importance to recent citations (Equation 5.9). As discussed in [HCKW10], which originally proposes this concept for author ranking algorithms, we set the decay factor $\tau = 4$. The venue prestige of an edge, represented by $v(A_{i'} \rightarrow A_i, P_j)$, gives more importance to citations in important venues (Equation 5.10). Finally, the individuality of an edge, represented by $w(A_{i'} \rightarrow A_i, P_j)$, gives more importance to citations received in papers where author A_i has few (or no) co-authors (Equation 5.8). The author's attribute total out-edge weight is obtained by summing all of its out-edges, as shown below.

Citations recency total weight

$$a_{out}(A_i) = \sum_{(A_i \rightarrow A_{i'}, P_j)} a(A_i \rightarrow A_{i'}, P_j) \tag{5.11}$$

Cited individuality total weight

$$w_{out}(A_i) = \sum_{(A_i \rightarrow A_{i'}, P_j)} w(A_i \rightarrow A_{i'}, P_j) \tag{5.12}$$

Citations prestige total weight

$$v_{out}(A_i) = \sum_{(A_i \rightarrow A_{i'}, P_j)} v(A_i \rightarrow A_{i'}, P_j) \tag{5.13}$$

5.4.4.2 OTARIOS

Here we propose a new feature enriched topology algorithm for citation networks, named OTARIOS. OTARIOS divides the citation network in two subnetworks, *insiders* (i.e., authors for which we have all their citations) and *outsiders* (i.e., authors that cite insiders but for whom not all citations are known). Then, only insiders are ranked while outsiders influence the ranks of insiders.

OTARIOS efficiently combines different publication/citation attributes in a multi-edge weighted network (instead of a simple weighted network used by traditional PageRank). Furthermore, OTARIOS is a flexible algorithm, allowing users to personalize which publication/citation attributes are used to rank researchers (e.g., value venue prestige highly or lowly).

Like traditional PageRank, OTARIOS consists of two steps: score initialization and score diffusion.

During score initialization, OTARIOS computes an initial score for each author, represented by $R(A_i)$. OTARIOS calculates $R(A_i)$ by taking into account multiple features that favor different author characteristics (Table 5.16). We divide the features into two categories: productivity and outsiders influence. Productivity measures the value of the author's publications, while outsider influence measures the value of the author's citations coming from outsiders. Regarding productivity, OTARIOS takes three factors into account: volume, recency and venues. Regarding outsiders influence, OTARIOS takes another three factors into account: individuality, recency and venues. We compute the author's initial score $R(A_i)$ as the sum of the two products of the factors in each group (i.e., productivity ($volume \times recency \times venues$) + outsiders influence ($individuality \times recency \times venues$)).

During score diffusion, OTARIOS improves author scores in an iterative process. Outsiders are removed from the network since their presence degrades the score diffusion step. In each iteration, OTARIOS updates an author's score $S(A_i)$ as $ST(A_i) + RR(A_i) + DN(A_i)$. We compute $RR(A_i)$ and $DN(A_i)$ in function of the initial rank of each author (discussed in Table 5.16), and compute $ST(A_i)$ in function of the author's citations coming from other insiders. OTARIOS considers three different features to assess score term $ST(A_i)$: individuality, recency and venues (Table 5.17). The $ST(A_i)$ at each iteration is the product of every features. (i.e., score term ($individuality \times recency \times venues$)). Like PageRank, OTARIOS stops when it reaches low variation in the node scores.

	Criteria	Initialization: $\boldsymbol{R(A_i)}$	Description
Productivity	Volume (P)	$\frac{\sum_{(P_j \in \mathcal{P}_{A_i})} \frac{1}{\lvert\mathcal{A}_{P_j}\rvert}}{\sum_{(A_{i'} \in \mathcal{A})} \sum_{(P_j \in \mathcal{P}_{A_{i'}})} \frac{1}{\lvert\mathcal{A}_{P_j}\rvert}}$	Favors publishing many papers with few co-authors.
	Recency (A)	$e^{\frac{-\delta(A_i)}{\tau}}$	Favors publishing recently.
	Venues (V)	$\left(\sum_{(P_j \in \mathcal{P}_{A_i})} v(P_j)\right) \times \lvert\mathcal{P}_{A_i}\rvert^{-1}$	Favors publishing in prestigious venues.
Outsiders Influence	Individuality (W)	$\sum_{(A_{i'} \to A_i, P_j)} \frac{\lambda(A_{i'}) \times w(A_{i'} \to A_i, P_j)}{w_{out}(A_{i'})}, A_{i'} \in \mathcal{O}$	Favors being cited by outsiders that cite few authors.
	Recency (A)	$\sum_{(A_{i'} \to A_i, P_j)} \frac{\lambda(A_{i'}) \times a(A_{i'} \to A_i, P_j)}{a_{out}(A_{i'})}, A_{i'} \in \mathcal{O}$	Favors being cited by outsiders more recently.
	Venues (V)	$\sum_{(A_{i'} \to A_i, P_j)} \frac{\lambda(A_{i'}) \times v(A_{i'} \to A_i, P_j)}{v_{out}(A_{i'})}, A_{i'} \in \mathcal{O}$	Favors being cited by outsiders in prestigious venues.

Table 5.16: List of features used for OTARIOS' author rank initialization: $R(A_i)$. OTARIOS considers both the authors' productivity and the direct influence of outsiders on the authors. We create different variants of these features, e.g., $PV + V$ uses volume (P) and venue prestige (V) to measure author productivity, and uses venue prestige (V) to measure the direct influence of outsiders.

Criteria	Score term: $\boldsymbol{ST(A_i)}$	Description
Individuality (W)	$\sum_{(A_{i'} \to A_i, P_j)} \frac{S(A_{i'}) \times w(A_{i'} \to A_i, P_j)}{w_{out}(A_{i'})}, A_{i'} \in \mathcal{I}$	Favors being cited by insiders that cite few authors.
Recency (A)	$\sum_{(A_{i'} \to A_i, P_j)} \frac{S(A_{i'}) \times a(A_{i'} \to A_i, P_j)}{a_{out}(A_{i'})}, A_{i'} \in \mathcal{I}$	Favors being cited by insiders more recently.
Venues (V)	$\sum_{(A_{i'} \to A_i, P_j)} \frac{S(A_{i'}) \times v(A_{i'} \to A_i, P_j)}{v_{out}(A_{i'})}, A_{i'} \in \mathcal{I}$	Favors being cited by insiders in prestigious venues.

Table 5.17: List of features used for OTARIOS' author score term calculation: $ST(A_i)$. Combined with author initialization (Table 5.16), we create different variants, e.g., PV+V+A combines initialization PV+V with score term A, i.e., using citation recency. All variants use $RR(N_i) = q \times R(N_i)$ and $DN(N_i) = (1 - q) \times R(N_i)$, thus we omit them from the table.

OTARIOS			**NDCG**						**MRR**			
variant	@5	@10	@20	@50	@100	Mean	@5	@10	@20	@50	@100	Mean
∅ + A + ∅	0.269	0.233	0.207	0.186	0.174	0.214	443	1125	903	1526	2066	1213
∅ + V + ∅	0.269	0.233	0.207	0.186	0.185	0.216	412	1108	916	1522	2096	1211
∅ + AV + ∅	0.269	0.233	0.207	0.186	0.177	0.215	419	1109	902	1511	2074	1203
AP + A + ∅	0.288	0.246	0.259	0.218	0.241	0.250	350	500	440	1121	1502	783
AP + V + ∅	0.288	0.246	0.258	0.218	0.239	0.250	344	489	439	1134	1527	787
AP + AV + ∅	0.288	0.246	0.259	0.218	0.240	0.250	345	494	439	1143	1523	789
AP + A + A	0.380	0.297	0.283	0.282	0.280	0.304	385	647	472	1111	1416	806
AP + A + AV	**0.407**	0.345	0.291	0.291	0.274	0.322	242	614	473	1116	1455	780
AP + A + AW	0.381	**0.369**	**0.313**	**0.302**	**0.288**	**0.330**	**219**	**386**	**328**	**879**	**1219**	**606**

Table 5.18: Comparison of OTARIOS variants on network NET. For each OTARIOS variant, we measure NDCG and MRR for the top-n ranked authors (@n), as well as the metric's mean value. Bold highlights the highest score for each metric. The best OTARIOS variant is colored in blue.

5.4.4.3 Results

In our analysis we do not assume that combining all publication/citation information (i.e., features) necessarily leads to the best results. Instead, we start with simple OTARIOS variants and progressively add new features. We illustrate this process for network NET (Table 5.18). We begin by comparing OTARIOS variants that only consider outsiders influence (e.g., ∅+A+∅). For the best ones, we add productivity features (e.g., AP+A+∅). In general, we see that results improve when merging outsiders influence with productivity. Finally, we add score term calculation to OTARIOS (e.g., AP+A+A). For the NET network, we see that AP+A+A is the variant that obtains the best results, with a mean NDCG of 0.330 and a mean MRR of 606.

We compare OTARIOS against PageRank and CountRank (CR), a baseline bibliometric. CR counts the total citations received by each author. We use CR because our DBLP dataset does not contain the authors' h-index and collecting this information for all 300k authors is not feasible. We create three CR variants: uniform, individuality and position. For each citation an author receives, uniform assigns the same merit to each co-author, individuality equally divides the merit by all co-authors (i.e., more co-authors means that each one receives less credit), and position gives more credit to authors whose name appears first in the publication (i.e., the first author gets double the merit of the second author, triple of the third author, etc.).

We show that OTARIOS achieves significantly better results than topology-based PageRank (by $> 30\%$) and CR (by $> 20\%$) (Table 5.19). We show results for the five best OTARIOS variants, but we should note that 21 OTARIOS variants, of the total 53, obtain better mean MRR and NDCG than the best competitor. The best

	NDGC						MRR					
Method	CM	TC	NET	IS	SE	Mean	CM	TC	NET	IS	SE	Mean
$CR_{position}$	0.097	0.049	0.189	0.176	0.261	0.154	1427	463	1009	892	324	823
$CR_{uniform}$	0.138	0.045	0.278	0.189	0.222	0.174	1659	516	1066	1067	387	939
PageRank	**0.180**	0.032	0.231	0.176	0.338	0.191	1203	508	817	720	356	721
$CR_{individuality}$	0.129	0.043	0.247	0.211	0.372	0.200	1171	438	878	744	289	704
∅ + AVW + AW	0.143	0.081	0.323	0.213	0.315	0.215	1161	324	664	707	289	629
∅ + V + AW	0.148	0.080	0.321	0.214	0.314	0.215	1169	325	671	709	294	634
AP + VW + AW	0.150	**0.087**	0.330	**0.268**	**0.383**	0.244	**1070**	273	**604**	680	**207**	**567**
AV + VW + AW	0.143	0.085	**0.356**	0.264	**0.383**	**0.246**	1333	285	618	**676**	215	626
AP + A + AW	0.152	**0.087**	0.330	0.273	**0.383**	0.245 (+22%)	1079	**272**	606	688	**207**	570 (+24%)

Table 5.19: Comparison of competing methods against OTARIOS over all networks. The value of each cell is the metric's mean value for that network (e.g., the mean NDCG and MRR of AP+A+AW for network NET is highlighted in Table 5.18). In bold we highlight the highest score for each metric. The best competing method is colored in red and the best OTARIOS variant is colored in blue. Inside parentheses we show the gain of OTARIOS versus $CR_{individuality}$.

OTARIOS variant obtains a mean NDCG of 0.246 and a mean MRR of 567 over all five networks. Our best OTARIOS variant overall (i.e., considering a combination of NDCG and MRR) is AP+A+AW, which considers (i) the author's publication volume and publication recency, (ii) how recently his work is being cited by outsiders, and (iii) how recently his work is being cited by insiders and how individual he his.

We analyze which features are typically more important by visualizing which features the top OTARIOS variants use (Table 5.20). The best OTARIOS variants (i.e., all top-9 variants) combine productivity, outsiders influence and score term, showing the importance of considering multiple aspects of both publications and citations. Recency (A) seems to be more relevant to evaluate productivity and insiders score term than outsiders influence. On the other hand, venue prestige (V) seems to more relevant to evaluate outsiders influence than author productivity and insiders score term. This is expected because insiders tend to publish in the same venues, while outsiders might cite insiders in any venues, thus the venue prestige of outsider citations varies greatly.

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Prod.	A	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟥	🟥	🟦	🟥	🟥	🟦	🟦	🟦	🟥	🟦	🟥
	P	🟥	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟥	🟥	🟦	🟥	🟥	🟦	🟦	🟥	🟥	🟥	🟥
	V	🟦	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
OI	A	🟥	🟦	🟥	🟦	🟦	🟥	🟥	🟥	🟦	🟦	🟥	🟦	🟦	🟦	🟦	🟥	🟦	🟥	🟥	🟥
	V	🟦	🟥	🟦	🟦	🟦	🟥	🟦	🟥	🟥	🟥	🟦	🟥	🟦	🟦	🟦	🟦	🟥	🟥	🟥	🟥
	W	🟦	🟥	🟥	🟥	🟦	🟦	🟦	🟦	🟥	🟥	🟥	🟥	🟥	🟦	🟥	🟥	🟥	🟥	🟦	🟦
ST	A	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
	V	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟦	🟦	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
	W	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟥	🟥	🟦	🟦	🟥	🟦	🟦	🟥	🟥	🟥	🟥	🟥	🟦

Table 5.20: Features considered by the 20 best OTARIOS variants (by mean NDCG). Rows represent different features (related to productivity (Prod.), outsiders influence (OI) and score term (ST)) and columns represent the OTARIOS variant ranked at position n. Blue means that the feature is considered and red that it is not.

5.5 Summary

Here we presented a novel measure for node dominance, named GD. We demonstrated that GD captures different information from node centrality measures and is better suited than those to find dominant nodes in a network. We also analyzed the importance of (a) giving different weights to dominant (good) and dominated (bad) edges, (b) direct and indirect dominances, and (c) the size of the graphlets being enumerated.

Our ranking mechanism based on GD considers not only the subgraphs themselves but also the position (or orbit) of the players in the subgraphs. GD differs PageRank, since PageRank only considers one of the two possible edge directions, giving importance to wins and almost disregarding losses, or vice-versa.

We use our GD measure to two applications: (a) ranking tennis players and (b) ranking scientific authors. In (a) there is a clear notion of dominance while in (b) there is not.

Our results on the tennis networks showed that, even without any kind of meta-information (e.g., tournament or round information), GD is able to produce consistent and meaningful results using only the topology of the network. Our ranking system produces results that agree with the ATP ranking while at same time offering a different perspective since the intricate relations between players are also captured. This approach gives a better idea of actual player dominance which is valuable when trying to assess who are the best tennis players.

Our results on the citation networks showed that GD is superior to PageRank when ranking scientists and comparing with a ground-truth of best paper awards. GD and PageRank are topology-based approaches, which only use the topology of the network to rank nodes.

We also presented a preliminary feature enriched topology ranked, named OTARIOS. OTARIOS is based on PageRank and capable of combining publication and citation information. Our tests showed that OTARIOS is $\approx 20\%$ more efficient than bibliometric approaches and $\approx 30\%$ more efficient than PageRank. How to adapt GD to consider these features is left for future work.

Chapter 6

Conclusions and future work

Networks are widely used to model systems. Extracting their topological features, namely subgraphs, offers rich structural information that boosts our understating of network function. Depending on the task, practitioners might want to study the whole system (i.e., the network) or its agents (i.e., the nodes). Network comparison and node ranking are thus fundamental tasks in network science. The purpose of this work was to provide efficient methods for both network comparison and node ranking.

This chapter points out the main contributions, discusses limitations and proposes directions for future research, and presents concluding remarks.

6.1 Main contributions

This work described in this thesis consisted of the design, implementation, and evaluation of network comparison and node ranking methods. We proposed new methods for network comparison and node ranking. Our methods were shown to be consistently faster and more accurate than similar (i.e., graphlet-based measures) state-of-the-art solutions when we tested them both on synthetic and real-world networks. Our tools are available online for practitioners. Next, we give a more detailed description of our contributions.

Directed graphlets. We extended undirected graphlets to take into account the edge direction of the subgraphs. We showed that, in the case of directed graphlets, edge direction captures relevant information than undirected graphlets ignored. We tested our hypothesis on (a) a set of synthetic network pertaining to different directed graph models and on (b) real-world data corresponding to directed biological networks. We verified that directed graphlets achieved higher accuracy than undirected graphlets when classifying both types of networks.

Graphlet-tries. Related to directed graphlets, we extended the concept of g-tries to take into account the orbits of the subgraphs stored in the g-trie. Graphlet-tries are an efficient structure to store and enumerate graphlets. In our experiments, we verified that our method consisting of directed graphlets and graphlet-tries, named GT-Scanner, outperforms state-of-the-art algorithms in terms of running time. The tool is made available online[1].

Graphlet-orbit Transitions (GoTs). We proposed GoTs, which are graphlet-based features of temporal networks. Most previously existing graphlet-based features were static or only allowed for one new event per graphlet transition. We showed that GoTs capture complex subgraph transitions that other graphlet-based measures do not. We showed that GoTs improve upon state-of-the-art both in terms of accuracy and running time when classifying synthetic and real temporal networks.

GoT-WAVE. We proposed GoT-WAVE, a method for temporal network alignment (NA). We combined WAVE [SCTM15], a fast algorithm for static NA, with GoTs, the set of temporal graphlet-based features previously discussed. GoT-WAVE outperformed state-of-the-art temporal NA algorithms for most tests that we performed on synthetic and real data, in terms of accuracy and running time. The tool is made available online[2].

Graphlet-based node ranking. We proposed GD a measure of node ranking, based on the notion of node dominance, that takes into account the graphlets that the nodes appear at. GD can also be regarded as a new node centrality measure. We compared GD with other node centrality measures and applied it to real-world test cases: (a) player dominance in a sports network and (b) author impact in citations networks. While we focused on these two test-cases, our method is applicable to any dominance network.

OTARIOS. We proposed a PageRank-based measure for node ranking, named OTAR-

[1] `http://www.dcc.fc.up.pt/~daparicio/software`

[2] `http://www.dcc.fc.up.pt/got-wave`

IOS. OTARIOS is specific to author citations networks and uses features beyond the network. This contrasts with GD, which only uses the topology of the network. OTARIOS takes into consideration multiple factors (e.g., venue prestige, year) concerning the authors' productivity (i.e., their publications) and the authors' impact (i.e., their citations). We showed that OTARIOS outperformed competing methods in terms of how well its ranking matches the ground-truth ranking on several real networks, each comprised of citations in conferences on a given topic.

6.2 Future work

We hope that the work presented in this thesis can lead to future research in the area. We put forward methods for three different tasks, namely network classification, network alignment, and node ranking, and each of these areas is very broad and with many research opportunities. Most of the work presented in this thesis relied on efficient subgraph counting, and that is another area where research is active and with many possible directions for future work. We give some pointers of future work next.

Extend graphlets and g-tries to multilayer networks. In our work, we extended both g-tries and graphlets to directed and temporal networks. We did not address multilayer networks. Work on multilayer layers is ever increasing, and extending the techniques we presented to them might be an interesting research direction.

Use GoTs in other alignment algorithms. We used GoTs as node conservation features in WAVE [SCTM15], resulting in GoT-WAVE. We also ran experiments for MAGNA++ [VSM15], which is an alignment method inferior to WAVE. Recently, SANA [MH17] has been proposed has been shown to have good results. Extending SANA to temporal networks and using GoTs as node conservation features might lead to a method superior to GoT-WAVE.

Better edge conservation with GoTs. In our experiments we observed that GoT-WAVE's performance was not augmented as much as DynaWAVE's when edge conservation was combined with node conservation. A possible direction is to find edge conservation measures, different from DS^3 and DWEC, that improve GoT-WAVE's performance.

Combine GoTs with DGDV. We performed preliminary experiments where we combined GoTs with DGDV as node conservation features for temporal NA. We did not observe a positive growth in accuracy; however, further research on how to combine the two measure efficiently could lead to better results than when using them separately.

GD for feature enriched topology ranking. We extended PageRank from a topology based approach to a feature enriched topology approach. Due to time limitations, we did not follow through with an extension of GD that also uses features from the network besides its topology. In our experiments, we observed that GD was superior to PageRank, thus a feature enriched version of GD might have better results than OTARIOS.

CPU-GPU version of g-tries. We have some preliminary work on adapting g-tries to the GPU [Apa14]. At the time, our results did not improve upon the CPU version. However, combining CPU with GPU could have good results by following a strategy similar to [RZ16]: many simple tasks are handed over to the GPU while the CPU solves just a few complex tasks (to avoid branching problems in the GPU). Since the focus of this thesis was not on efficient parallel solutions to subgraph counting, we left this problem open for future research.

6.3 Closing remarks

When this dissertation started more than four years ago, its theme was the very broad ”Analysis of Complex Networks”. Throughout the years we managed to focus on just two problems, which are almost as broad: network comparison and node ranking. Subgraphs ended up being the common thread through (most) of this work and, at the end, we feel that everything seems cohesive.

Researching these tasks was a very interesting undergoing, where we studied both algorithmic aspects and applied our methods to real data. We hope that our tools are of use to the scientific community and to practitioners. One of the most rewarding aspects during this long process was to receive e-mails from strangers, who are also studying the same problems, asking for clarifications or with help running our code.

Finally, we also hope that (at least some of) the opportunities for research are followed through by other people.

Appendix A

Graphlet-tries

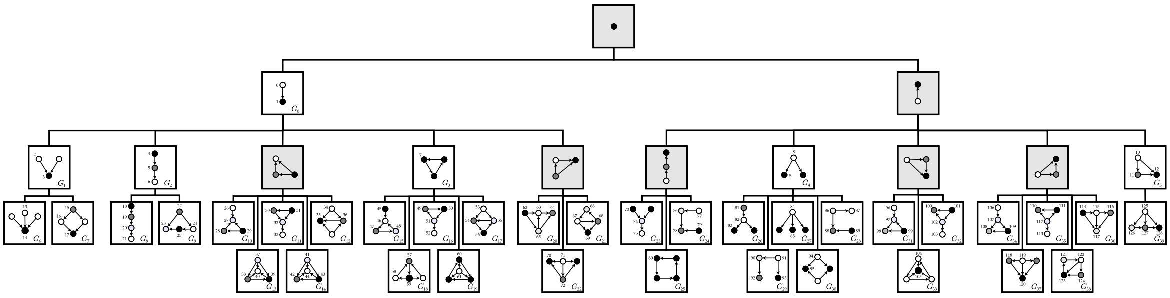


Figure A.1: A graphlet-trie containing the 39 non-bidirectional directed graphlets of sizes 2, 3 and 4. The orbit numbers are generated automatically. While it is guaranteed that the g-trie only has non-isomorphic graphs on the leafs (bottom-level nodes), isomorphic graphs may appear in some of the top nodes (represented in grey). In these cases only one of the graphs is considered for orbit counting while the others are only used to efficiently traverse the search space, using symmetry breaking conditions to guarantee that each occurrence is only counted once.

Appendix B

Temporal network randomization

B.1 Undirected randomization

This randomization scheme was proposed in [Hol15] and used in [VCM17, VM17b]. Given the original undirected dynamic network $\mathcal{S}_N$ a randomization percentage p, one randomly picks edge e_1 to be rewired. We then pick another random edge e_2 and, with probability p, we rewire the two events. That is, given $e_1 = (u, v, S_i, S_f)$ and $e_2 = (u', v', S_i', S_f')$ (where S_i and S_i' are the starting snapshots, and S_f and S_f' are the ending snapshots) we do one of the following transformations with 50% probability:

- $e_1 = (u, v, S_i, S_f) \rightarrow e_1 = (u, v', S_i, S_f)$ and
  $e_2 = (u', v', S_i', S_f') \rightarrow e_2 = (u', v, S_i, S_f)$, or

- $e_1 = (u, v, S_i, S_f) \rightarrow e_1 = (u, u', S_i, S_f)$ and
  $e_2 = (u', v', S_i', S_f') \rightarrow e_2 = (v, v', S_i, S_f)$, or

If the transformation was performed, e_1 and e_2 are both taken out of the list of edges to be rewired. Otherwise, only e_1 is taken out. The process is followed until no edges are to be rewired.

B.2 Directed randomization

This randomization scheme is adapted from [Hol15] to directed networks. Given the original directed dynamic network $\mathcal{S}_N$ a randomization percentage p, one randomly picks edge e_1 to be rewired. We then pick another random edge e_2 and, with probability p, we rewire the two events. That is, given $e_1 = (u, v, S_i, S_f)$ and $e_2 = (u', v', S'_i, S'_f)$ (where S_i and S'_i are the starting snapshots, and S_f and S'_f are the ending snapshots) we do one of the following transformations with 50% probability:

- $e_1 = (u, v, S_i, S_f) \rightarrow e_1 = (u, v', S_i, S_f)$ and
  $e_2 = (u', v', S'_i, S'_f) \rightarrow e_2 = (u', v, S_i, S_f)$, or
- $e_1 = (u, v, S_i, S_f) \rightarrow e_1 = (u, u', S_i, S_f)$ and
  $e_2 = (u', v', S'_i, S'_f) \rightarrow e_2 = (v, v', S_i, S_f)$, or

If the transformation was performed, an additional parameter γ controls edge reversal. So, with probability γ, one performs the transformation for each edge:

- $e_k = (x, y, S_n, S_m) \rightarrow e_k = (y, x, S_n, S_m)$

Then, e_1 and e_2 are both taken out of the list of edges to be rewired. Otherwise, only e_1 is taken out. The process is followed until no edges are to be rewired.

B.3 Pure directed randomization

Given the original dynamic directed network $\mathcal{S}_N$, one randomly picks edge e_1 to be rewired. That is, given $e_1 = (u, v, S_i, S_f)$ (where S_i is the starting snapshot, and S_f is the ending snapshot), and a randomization percentage p, we the following transformations with p% probability:

- $e_1 = (u, v, S_i, S_f) \rightarrow e_1 = (v, u, S_i, S_f)$

Otherwise, e_1 is kept as it was. e_1 is taken out of the list of edges to be rewired and the process is followed until no edges are to be rewired.

References

[AA04] István Albert and Réka Albert. Conserved network motifs allow protein–protein interaction prediction. *Bioinformatics*, 20(18):3346–3352, 2004.

[ABG⁺16] Maryam Aliakbarpour, Amartya Shankha Biswas, Themistoklis Gouleakis, John Peebles, Ronitt Rubinfeld, and Anak Yodpinyanee. Sublinear-time algorithms for counting star subgraphs with applications to join selectivity estimation. *arXiv preprint arXiv:1601.04233*, 2016.

[Agg15] Charu C Aggarwal. *Data mining: the textbook*. Springer, 2015.

[ÁMDGL⁺14] M Álvarez-Moreno, Cornelis De Graaf, Nuria Lopez, Feliu Maseras, Josep M Poblet, and Carles Bo. Managing the computational chemistry big data problem: the iochem-bd platform. *Journal of chemical information and modeling*, 55(1):95–103, 2014.

[ANR⁺17] Nesreen K Ahmed, Jennifer Neville, Ryan A Rossi, Nick G Duffield, and Theodore L Willke. Graphlet decomposition: Framework, algorithms, and applications. *Knowledge and Information Systems*, 50(3):689–722, 2017.

[Apa14] David Oliveira Aparício. Pattern discovery in complex networks using parallelism. 2014.

[ARdS14] 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.

REFERENCES

[ARMS18] David Aparício, Pedro Ribeiro, Tijana Milenković, and Fernando Silva. Got-wave: Temporal network alignment using graphlet-orbit transitions. *arXiv preprint arXiv:1808.08195*, 2018.

[ARMS19] David Aparício, Pedro Ribeiro, Tijana Milenković, and Fernando Silva. Temporal network alignment via GoT-WAVE. *Bioinformatics*, 2019.

[ARS+07] Yassen Assenov, Fidel Ramírez, Sven-Eric Schelhorn, Thomas Lengauer, and Mario Albrecht. Computing topological parameters of biological networks. *Bioinformatics*, 24(2):282–284, 2007.

[ARS15] David Aparício, Pedro Ribeiro, and Fernando Silva. Network comparison using directed graphlets. *arXiv preprint arXiv:1511.01964*, 2015.

[ARS16] David Aparício, Pedro Ribeiro, and Fernando Silva. A subgraph-based ranking system for professional tennis players. In *Complex Networks VII*, pages 159–171. Springer, 2016.

[ARS17] David Aparício, Pedro Ribeiro, and Fernando Silva. Extending the applicability of graphlets to directed networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 14(6):1302–1315, 2017.

[ARS18] David Aparício, Pedro Ribeiro, and Fernando Silva. Graphlet-orbit transitions (got): A fingerprint for temporal network comparison. *PloS one*, 13(10):e0205497, 2018.

[AZBP17] Bijaya Adhikari, Yao Zhang, Aditya Bharadwaj, and B Aditya Prakash. Condensing temporal networks using propagation. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 417–425. SIAM, 2017.

[BA99] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[BB02] Christian Borgelt and Michael R Berthold. Mining molecular fragments: Finding relevant substructures of molecules. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 51–58. IEEE, 2002.

[BBY09] Dan Braha and Yaneer Bar-Yam. Time-dependent complex networks: Dynamic centrality, dynamic motifs, and cycles of social interactions. In *Adaptive Networks*, pages 39–50. Springer, 2009.

[BC18] Alain Barrat and Ciro Cattuto. Sociopatterns. `http://www.sociopatterns.org/datasets/`, 2018. Accessed: 2018-08-11.

[BCF$^+$10] Luc Brun, Donatello Conte, Pasquale Foggia, Mario Vento, and Didier Villemin. Symbolic learning vs. graph kernels: An experimental comparison in a chemical application. In *ADBIS (Local Proceedings)*, pages 31–40, 2010.

[BFL$^+$06] Luciana S Buriol, Gereon Frahling, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Christian Sohler. Counting triangles in data streams. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 253–262. ACM, 2006.

[BM06] Vladimir Batagelj and Andrej Mrvar. Pajek datasets. `http://vlado.fmf.uni-lj.si/pub/networks/data/bio/foodweb/foodweb.htm`, 2006. [Online; accessed 20-September-2018].

[BRAH12] Mansurul A Bhuiyan, Mahmudur Rahman, and M Al Hasan. Guise: Uniform sampling of graphlets for large graph analysis. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 91–100. IEEE, 2012.

[B.V18] Elvesier B.V. *Research Metrics Guidebook.* Elvesier, 2018.

[CC00] Trevor F Cox and Michael AA Cox. *Multidimensional scaling.* Chapman and hall/CRC, 2000.

[CGW99] Edward M Condon, Bruce L Golden, and Edward A Wasil. Predicting the success of nations at the summer olympics using neural networks. *Computers & Operations Research*, 26(13):1243–1265, 1999.

[Chi18] Mike Chirico. Phildelphia crime data. https://www.kaggle.com/mchirico/philadelphiacrimedata, 2018. Accessed: 2018-03-02.

[CM18] Joseph Crawford and Tijana Milenković. Cluenet: Clustering a temporal network based on topological similarity rather than denseness. *PloS one*, 13(5):e0195993, 2018.

[CMJ+07] Qinghua Cui, Yun Ma, Maria Jaramillo, Hamza Bari, Arif Awan, Song Yang, Simo Zhang, Lixue Liu, Meng Lu, Maureen O'Connor-McCourt, and Others. A map of human cancer signaling. *Molecular systems biology*, 3(1):152, 2007.

[COJT+11] Luciano da Fontoura Costa, Osvaldo N Oliveira Jr, Gonzalo Travieso, Francisco Aparecido Rodrigues, Paulino Ribeiro Villas Boas, Lucas Antiqueira, Matheus Palhares Viana, and Luis Enrique Correa Rocha. Analyzing and modeling real-world phenomena with complex networks: a survey of applications. *Advances in Physics*, 60(3):329–412, 2011.

[CPS+16] Cesar H Comin, Thomas K Peron, Filipi N Silva, Diego R Amancio, Francisco A Rodrigues, and Luciano da F Costa. Complex systems: features, similarity and connectivity. *arXiv preprint arXiv:1606.05400*, 2016.

[CRBS12] Sarvenaz Choobdar, Pedro Ribeiro, Sylwia Bugla, and Fernando Silva. Comparison of co-authorship networks across scientific fields using motifs. In *Advances in Social Networks Analysis and Mining (ASONAM), 2012 IEEE/ACM International Conference on*, pages 147–152. IEEE, 2012.

[CSM15] Joseph Crawford, Yihan Sun, and Tijana Milenković. Fair evaluation of global network aligners. *Algorithms for Molecular Biology*, 10(1):19, 2015.

[CZ14] CL Philip Chen and Chun-Yang Zhang. Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences*, 275:314–347, 2014.

[DBWF11] Mina Doroud, Prantik Bhattacharyya, S Felix Wu, and Diane Felmlee. The evolution of ego-centric triads: A microscopic approach toward predicting macroscopic network properties. pages 172–179, 2011.

[DCLS+10] Munmun De Choudhury, Yu-Ru Lin, Hari Sundaram, K Selcuk Candan, Lexing Xie, Aisling Kelliher, et al. How does the data sampling strategy impact the discovery of information diffusion in social media? *Icwsm*, 10:34–41, 2010.

[DG08] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[Din09] Ying Ding. Applying weighted pagerank to author citation networks. *Journal of the American Society for Information Science and Technology*, 62(2):236–245, 2009.

[DKS12] Nicholas Dingle, William Knottenbelt, and Demetris Spanias. On the (page) ranking of professional tennis players. In *European Workshop on Performance Engineering*, pages 237–247. Springer, 2012.

[DV12] Marcel Dunaiski and Willem Visser. Comparing paper ranking algorithms. In *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*, pages 21–30. ACM, 2012.

[ECK16] Ahed Elmsallati, Connor Clark, and Jugal Kalita. Global alignment of protein-protein interaction networks: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 13(4):689–705, 2016.

[Elo61] AE Elo. New uscf rating system. *Chess Life*, 16:160–161, 1961.

[ER60] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[ERS18] Talya Eden, Dana Ron, and C Seshadhri. On approximating the number of k-cliques in sublinear time. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 722–734. ACM, 2018.

[ESDS16] Frank Emmert-Streib, Matthias Dehmer, and Yongtang Shi. Fifty years of graph matching, network alignment and network comparison. *Information Sciences*, 346:180–197, 2016.

[FFF14] Irene Finocchi, Marco Finocchi, and Emanuele G Fusco. Counting small cliques in mapreduce, 2014.

[FFF15] Irene Finocchi, Marco Finocchi, and Emanuele G Fusco. Clique counting in mapreduce: algorithms and experiments. *Journal of Experimental Algorithmics (JEA)*, 20:1–7, 2015.

[FLL+08] Wenbin Fang, Ka Keung Lau, Mian Lu, Xiangye Xiao, Chi K Lam, Philip Yang Yang, Bingsheng He, Qiong Luo, Pedro V Sander, and Ke Yang. Parallel data mining on graphics processors. *Hong Kong Univ. Sci. and Technology, Hong Kong, China, Tech. Rep. HKUST-CS08-07*, 2008.

[FM14] Fazle E Faisal and Tijana Milenković. Dynamic networks reveal key players in aging. *Bioinformatics*, 30(12):1721–1729, 2014.

[FMCM15] Fazle E Faisal, Lei Meng, Joseph Crawford, and Tijana Milenković. The post-genomic era of biological network alignment. *EURASIP Journal on Bioinformatics and Systems Biology*, 2015(1):3, 2015.

[FND+09] Jason Flannick, Antal Novak, Chuong B Do, Balaji S Srinivasan, and Serafim Batzoglou. Automatic parameter learning for multiple local network alignment. *Journal of computational biology*, 16(8):1001–1022, 2009.

[FZM15] Fazle Elahi Faisal, Han Zhao, and Tijana Milenković. Global network alignment in the context of aging. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 12(1):40–52, 2015.

[GBC14] Valerio Gemmetto, Alain Barrat, and Ciro Cattuto. Mitigation of infectious disease at school: targeted class closure vs school closure. *BMC infectious diseases*, 14(1):695, 2014.

[GGG+11] Wei Gao, John H Gilmore, Kelly S Giovanello, Jeffery Keith Smith, Dinggang Shen, Hongtu Zhu, and Weili Lin. Temporal and spatial evolution of brain network topology during the first two years of life. *PloS one*, 6(9):e25278, 2011.

[GK07] Joshua A Grochow and Manolis Kellis. Network motif discovery using subgraph enumeration and symmetry-breaking. In *Annual International Conference on Research in Computational Molecular Biology*, pages 92–106. Springer, 2007.

[GKK01] K-I Goh, B Kahng, and D Kim. Universal behavior of load distribution in scale-free networks. *Physical Review Letters*, 87(27):278701, 2001.

[GL04] Diego Garlaschelli and Maria I Loffredo. Patterns of link reciprocity in directed networks. *Physical review letters*, 93(26):268701, 2004.

[GL16] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[GM17] Pietro Hiram Guzzi and Tijana Milenković. Survey of local and global biological network alignment: the need to reconcile the two sides of the same coin. *Briefings in bioinformatics*, 19(3):472–481, 2017.

[GM18] Shawn Gu and Tijana Milenković. Graphlets versus node2vec and struc2vec in the task of network alignment. *arXiv preprint arXiv:1805.04222*, 2018.

[GRS11] Mira Gonen, Dana Ron, and Yuval Shavitt. Counting stars and other small subgraphs in sublinear-time. *SIAM Journal on Discrete Mathematics*, 25(3):1365–1411, 2011.

[GRSJRMK13] Ali Gholami Rudi, Saeed Shahrivari, Saeed Jalili, and Zahra Razaghi Moghadam Kashani. Rangi: a fast list-colored graph motif finding algorithm. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 10(2):504–513, 2013.

[GS04] Shantanu Godbole and Sunita Sarawagi. Discriminative methods for multi-labeled classification. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 22–30. Springer, 2004.

[GS13] Sylvain Guillemot and Florian Sikora. Finding and counting vertex-colored subtrees. *Algorithmica*, 65(4):828–844, 2013.

[GVF$^+$15] Mathieu Génois, Christian L Vestergaard, Julie Fournet, André Panisson, Isabelle Bonmarin, and Alain Barrat. Data on face-to-face contacts in an office building suggest a low-cost vaccination strategy based on community linkers. *Network Science*, 3(3):326–347, 2015.

[HBEJF14] Benjamin T Hazen, Christopher A Boone, Jeremy D Ezell, and L Allison Jones-Farmer. Data quality for data science, predictive analytics, and big data in supply chain management: An introduction to the problem and suggestions for research and applications. *International Journal of Production Economics*, 154:72–80, 2014.

[HCKW10] Won-Seok Hwang, Soo-Min Chae, Sang-Wook Kim, and Gyun Woo. Yet another paper ranking algorithm advocating recent publications.

In *Proceedings of the 19th international conference on World wide web*, pages 1117–1118. ACM, 2010.

[HCM15] Yuriy Hulovatyy, Huili Chen, and T Milenković. Exploring the structure and function of temporal networks with dynamic graphlets. *Bioinformatics*, 31(12):i171–i180, 2015.

[HD14] Tomaž Hočevar and Janez Demšar. A combinatorial approach to graphlet counting. *Bioinformatics*, 30(4):559–565, 2014.

[HD15] Tomaž Hočevar and Janez Demšar. Orca. `http://www.biolab.si/supp/orca/`, 2015. [Online; accessed 20-September-2018].

[HD17] Tomaž Hočevar and Janez Demšar. Combinatorial algorithm for counting small induced graphs and orbits. *PloS one*, 12(2):e0171428, 2017.

[Hir05] Jorge E Hirsch. An index to quantify an individual's scientific research output. *Proceedings of the National academy of Sciences*, 102(46):16569–16572, 2005.

[Hol15] Petter Holme. Modern temporal network theory: a colloquium. *The European Physical Journal B*, 88(9):234, 2015.

[HOO11] Sungpack Hong, Tayo Oguntebi, and Kunle Olukotun. Efficient parallel graph exploration on multi-core cpu and gpu. In *Parallel Architectures and Compilation Techniques (PACT), 2011 International Conference on*, pages 78–88. IEEE, 2011.

[HS12] Petter Holme and Jari Saramäki. Temporal networks. *Physics reports*, 519(3):97–125, 2012.

[HW09] Haibo Hu and Xiaofan Wang. Evolution of a large online social network. *Physics Letters A*, 373(12):1105–1110, 2009.

[HWP03] Jun Huan, Wei Wang, and Jan Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. In *null*, page 549. IEEE, 2003.

[ISB$^+$11] Lorenzo Isella, Juliette Stehlé, Alain Barrat, Ciro Cattuto, Jean-François Pinton, and Wouter Van den Broeck. What's in a crowd? analysis of face-to-face behavioral networks. *Journal of theoretical biology*, 271(1):166–180, 2011.

[JCZ13] Chuntao Jiang, Frans Coenen, and Michele Zito. A survey of frequent subgraph mining algorithms. *The Knowledge Engineering Review*, 28(1):75–105, 2013.

[JHK12] Jeannette Janssen, Matt Hurshman, and Nauzer Kalyaniwalla. Model selection for social networks using graphlets. *Internet Mathematics*, 8(4):338–363, 2012.

[JK02] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[JS17] Shweta Jain and C Seshadhri. A fast and provable method for estimating clique counts using turán's theorem. In *Proceedings of the 26th International Conference on World Wide Web*, pages 441–449. International World Wide Web Conferences Steering Committee, 2017.

[JSP15] Madhav Jha, C Seshadhri, and Ali Pinar. Path sampling: A fast and provable method for estimating 4-vertex subgraph counts. In *Proceedings of the 24th International Conference on World Wide Web*, pages 495–505. International World Wide Web Conferences Steering Committee, 2015.

[JTA$^+$00] Hawoong Jeong, Bálint Tombor, Réka Albert, Zoltan N Oltvai, and A-L Barabási. The large-scale organization of metabolic networks. *Nature*, 407(6804):651–654, 2000.

[JYW10] Ning Jin, Calvin Young, and Wei Wang. Gaia: graph classification using evolutionary computation. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 879–890. ACM, 2010.

[KAB$^+$14] 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.

[KAE$^+$09] Zahra RM Kashani, Hayedeh Ahrabian, Elahe Elahi, Abbas Nowzari-Dalini, Elnaz S Ansari, Sahar Asadi, Shahin Mohammadi, Falk Schreiber, and Ali Masoudi-Nejad. Kavosh: a new algorithm for finding network motifs. *BMC bioinformatics*, 10(1):318, 2009.

[Kaz16] Ehsan Kazemi. Network alignment: Theory, algorithms, and applications. 2016.

[KIMA04] Nadav Kashtan, Shalev Itzkovitz, Ron Milo, and Uri Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11):1746–1758, 2004.

[KKK+11] Lauri Kovanen, Márton Karsai, Kimmo Kaski, János Kertész, and Jari Saramäki. Temporal motifs in time-dependent networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2011(11):P11005, 2011.

[KKK+12] Man-Sun Kim, Jeong-Rae Kim, Dongsan Kim, Arthur D Lander, and Kwang-Hyun Cho. Spatiotemporal network motif reveals the biological traits of developmental gene regulatory networks in drosophila melanogaster. *BMC systems biology*, 6(1):31, 2012.

[Kla09] Gunnar W Klau. A new graph-based method for pairwise global network alignment. *BMC bioinformatics*, 10(1):S59, 2009.

[KMM+10] Oleksii Kuchaiev, Tijana Milenković, Vesna Memišević, Wayne Hayes, and Nataša Pržulj. Topological network alignment uncovers biological function and phylogeny. *Journal of the Royal Society Interface*, page rsif20100063, 2010.

[KP09] Oleksii Kuchaiev and NATAŠA PRŽULJ. Learning the structure of protein-protein interaction networks. In *Biocomputing 2009*, pages 39–50. World Scientific, 2009.

[KS18] Ema Kušen and Mark Strembeck. On message exchange motifs emerging during human/bot interactions in multilayer networks: The case of two riot events. 2018.

[KSD+13] 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.

[KSHP14] Oleksii Kuchaiev, Aleksandar Stevanovic, Wayne Hayes, and Natasa Przulj. GraphCrunch 2: Software tool for network modeling, alignment and clustering. `http://www0.cs.ucl.ac.uk/staff/natasa/graphcrunch2/`, 2014. [Online; accessed 20-September-2018].

[KSK$^+$03] Brian P Kelley, Roded Sharan, Richard M Karp, Taylor Sittler, David E Root, Brent R Stockwell, and Trey Ideker. Conserved pathways within bacteria and yeast as revealed by global protein network alignment. *Proceedings of the National Academy of Sciences*, 100(20):11394–11399, 2003.

[KWG13] Avita Katal, Mohammad Wazid, and RH Goudar. Big data: issues, challenges, tools and good practices. In *Contemporary Computing (IC3), 2013 Sixth International Conference on*, pages 404–409. IEEE, 2013.

[KWNP09] Oleksii Kuchaiev, Po T Wang, Zoran Nenadic, and Natasa Przulj. Structure of brain functional networks. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 4166–4170. IEEE, 2009.

[LBKT08] Jure Leskovec, Lars Backstrom, Ravi Kumar, and Andrew Tomkins. Microscopic evolution of social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 462–470. ACM, 2008.

[LCL$^+$13] Wan Li, Lina Chen, Xia Li, Xu Jia, Chenchen Feng, Liangcai Zhang, Weiming He, Junjie Lv, Yuehan He, Weiguo Li, et al. Cancer-related marketing centrality motifs acting as pivot units in the human signaling network and mediating cross-talk between biological pathways. *Molecular BioSystems*, 9(12):3026–3035, 2013.

[LCR$^+$16] Linyuan Lü, Duanbing Chen, Xiao-Long Ren, Qian-Ming Zhang, Yi-Cheng Zhang, and Tao Zhou. Vital nodes identification in complex networks. *Physics Reports*, 650:1–63, 2016.

[LJC$^+$09] Yang Liu, Xiaohong Jiang, Huajun Chen, Jun Ma, and Xiangyu Zhang. Mapreduce-based pattern finding algorithm applied in motif detection for prescription compatibility network. In *International Workshop on Advanced Parallel Processing Technologies*, pages 341–355. Springer, 2009.

[LKF05] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international* *conference on Knowledge discovery in data mining*, pages 177–187. ACM, 2005.

[LKF07] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2, 2007.

[LLB$^+$09] Chung-Shou Liao, Kanghao Lu, Michael Baym, Rohit Singh, and Bonnie Berger. Isorankn: spectral methods for global alignment of multiple protein networks. *Bioinformatics*, 25(12):i253–i258, 2009.

[LLDM09] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[LXXL17] Wenqing Lin, Xiaokui Xiao, Xing Xie, and Xiao-Li Li. Network motif discovery: A gpu approach. *IEEE Transactions on Knowledge and Data Engineering*, 29(3):513–528, 2017.

[MA03] Shmoolik Mangan and Uri Alon. Structure and function of the feed-forward loop network motif. *Proceedings of the National Academy of Sciences*, 100(21):11980–11985, 2003.

[Mar02] Joseph Martinich. College football rankings: Do the computers know best? *Interfaces*, 32(5):85–94, 2002.

[Mar13] Vivien Marx. Biology: The big challenges of big data, 2013.

[MDCR$^+$16] Alberto JM Martin, Calixto Dominguez, Sebastián Contreras-Riquelme, David S Holmes, and Tomas Perez-Acle. Graphlet based metrics for the comparison of gene regulatory networks. *PloS one*, 11(10):e0163497, 2016.

[MDF$^+$12] Sam Mavandadi, Stoyan Dimitrov, Steve Feng, Frank Yu, Richard Yu, Uzair Sikora, and Aydogan Ozcan. Crowd-sourced biogames: managing the big data problem for next-generation lab-on-a-chip platforms. *Lab on a chip*, 12(20):4102–4106, 2012.

[MDP14] Noël Malod-Dognin and Nataša Pržulj. Gr-align: fast and flexible alignment of protein 3d structures using graphlet degree similarity. *Bioinformatics*, 30(9):1259–1265, 2014.

[Mel01] Merrill J Melnick. Relationship between team assists and win-loss record in the national basketball association. *Perceptual and Motor Skills*, 92(2):595–602, 2001.

[Mel06] Guy Melancon. Just how dense are dense graphs in the real world?: a methodological note. In *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*, pages 1–7. ACM, 2006.

[MGG12] Duane Merrill, Michael Garland, and Andrew Grimshaw. Scalable gpu graph traversal. In *ACM SIGPLAN Notices*, volume 47, pages 117–128. ACM, 2012.

[MH17] Nil Mamano and Wayne B Hayes. Sana: simulated annealing far outperforms many other search algorithms for biological network alignment. *Bioinformatics*, 33(14):2156–2164, 2017.

[MIK$^+$04] 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.

[MJW$^+$09] Avi Ma'ayan, Sherry L Jenkins, Ryan L Webb, Seth I Berger, Sudarshan P Purushothaman, Noura S Abul-Husn, Jeremy M Posner, Tony Flores, and Ravi Iyengar. Snavi: Desktop application for analysis and visualization of large-scale signaling networks. *BMC systems biology*, 3(1):10, 2009.

[ML14] Plácido Moreno and Sebastián Lozano. A network dea assessment of team efficiency in the nba. *Annals of Operations Research*, 214(1):99–124, 2014.

[MLP08] Tijana Milenković, Jason Lai, and Nataša Pržulj. Graphcrunch: a tool for large network analyses. *BMC bioinformatics*, 9(1):70, 2008.

[MM12] Shun Motegi and Naoki Masuda. A network-based dynamical ranking system for competitive sports. *Scientific reports*, 2:904, 2012.

[MMFDC14] 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.

[MNHP10] Tijana Milenković, Weng Ng, Wayne Hayes, and Nataša Pržulj. Optimal network alignment with graphlet degree vectors. *Cancer informatics*, 9:121, 2010.

[MNSK12] Ali Masoudi-Nejad, Falk Schreiber, and Zahra Razaghi Moghadam Kashani. Building blocks of biological networks: a review on major network motif discovery algorithms. *IET systems biology*, 6(5):164–174, 2012.

[Moh14] Shahin Mohammadi. Kavosh: a new algorithm for finding network motifs. `https://github.com/shmohammadi86/Kavosh`, 2014. [Online; accessed 20-September-2018].

[MP08] Tijana Milenković and Nataša Pržulj. Uncovering biological network function via graphlet degree signatures. *Cancer informatics*, 6:CIN–S680, 2008.

[MP12] Vesna Memišević and Nataša Pržulj. C-graal: Common-neighbors-based global graph alignment of biological networks. *Integrative Biology*, 4(7):734–743, 2012.

[MP14] Brendan D McKay and Adolfo Piperno. Practical graph isomorphism, ii. *Journal of Symbolic Computation*, 60:94–112, 2014.

[MPK11a] Radosław Michalski, Sebastian Palus, and Przemysław Kazienko. Matching organizational structure and social network extracted from email communication. In *International Conference on Business Information Systems*, pages 197–206. Springer, 2011.

[MPK11b] Radosław Michalski, Sebastian Palus, and Przemysław Kazienko. Matching organizational structure and social network extracted from email communication. In *Lecture Notes in Business Information Processing*, volume 87, pages 197–206. Springer Berlin Heidelberg, 2011.

[MS12] Dror Marcus and Yuval Shavitt. Rage–a rapid graphlet enumerator for large networks. *Computer Networks*, 56(2):810–819, 2012.

[MSM16] Lei Meng, Aaron Striegel, and Tijana Milenković. Local versus global biological network alignment. *Bioinformatics*, 32(20):3155–3164, 2016.

[MSOI⁺02] 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.

[New13] Mark Newman. Mark newman's network data. `http://www-personal.umich.edu/~mejn/netdata`, 2013. Accessed: 2018-08-11.

[NK05] Siegfried Nijssen and Joost N Kok. The gaston tool for frequent subgraph mining. *Electronic Notes in Theoretical Computer Science*, 127(1):77–87, 2005.

[NTM⁺13] Vincenzo Nicosia, John Tang, Cecilia Mascolo, Mirco Musolesi, Giovanni Russo, and Vito Latora. Graph metrics for temporal networks. In *Temporal networks*, pages 15–40. Springer, 2013.

[OB16] Mark Ortmann and Ulrik Brandes. Quad census computation: Simple, efficient, and orbit-aware. In *International Conference and School on Network Science*, pages 1–13. Springer, 2016.

[OBEJ08] Jens M Olesen, Jordi Bascompte, Heidi Elberling, and Pedro Jordano. Temporal dynamics in a pollination network. *Ecology*, 89(6):1573–1582, 2008.

[OFP⁺18] Stuart Oldham, Ben Fulcher, Linden Parkes, Aurina Arnatkeviciute, Chao Suo, and Alex Fornito. Consistency and differences between centrality metrics across distinct classes of networks. *arXiv preprint arXiv:1805.02375*, 2018.

[OSMN09] Saeed Omidi, Falk Schreiber, and Ali Masoudi-Nejad. Moda: an efficient algorithm for network motif discovery in biological networks. *Genes & genetic systems*, 84(5):385–395, 2009.

[PBL17] Ashwin Paranjape, Austin R Benson, and Jure Leskovec. Motifs in temporal networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 601–610. ACM, 2017.

[PBM⁺98] Lawrence Page, Sergey Brin, Rajeev Motwani, Terry Winograd, et al. The pagerank citation ranking: Bringing order to the web. 1998.

[PBMW99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[PCJ06] N Pržulj, Derek G Corneil, and Igor Jurisica. Efficient estimation of graphlet frequency distributions in protein–protein interaction networks. *Bioinformatics*, 22(8):974–980, 2006.

[Pet18] Jordan B Peterson. *12 Rules for Life: An Antidote to Chaos*. Random House Canada, 2018.

[PHHT10] Christopher Y Park, David C Hess, Curtis Huttenhower, and Olga G Troyanskaya. Simultaneous genome-wide inference of physical, genetic, regulatory, and functional pathway components. *PLoS computational biology*, 6(11):e1001009, 2010.

[PK12] Rob Patro and Carl Kingsford. Global network alignment using multiscale spectral signatures. *Bioinformatics*, 28(23):3105–3114, 2012.

[PKSH10] Natasa Pržulj, Oleksii Kuchaiev, Aleksandar Stevanovic, and Wayne Hayes. Geometric evolutionary dynamics of protein interaction networks. In *Pacific Symposium on Biocomputing*, volume 2009, pages 178–189, 2010.

[PR13] 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.

[Prž07a] Nataša Pržulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, 2007.

[Prz07b] Nataša Przulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23:177–183, 2007.

[PSV17] Ali Pinar, C Seshadhri, and Vaidyanathan Vishal. Escape: Efficiently counting all 5-vertex subgraphs. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1431–1440. International World Wide Web Conferences Steering Committee, 2017.

[PTTW13] Aduri Pavan, Kanat Tangwongsan, Srikanta Tirthapura, and Kun-Lung Wu. Counting and sampling triangles from a graph stream. *Proceedings of the VLDB Endowment*, 6(14):1870–1881, 2013.

[RA18] Ryan Rossi and Nesreen Ahmed. Dynamic networks — network repository. `http://networkrepository.com/dynamic.php`, 2018. Accessed: 2018-08-11.

[Rad11] Filippo Radicchi. Who is the best player ever? a complex network analysis of the history of professional tennis. *PloS one*, 6(2):e17249, 2011.

[RFMV09] Filippo Radicchi, Santo Fortunato, Benjamin Markines, and Alessandro Vespignani. Diffusion of scientific credits and the ranking of scientists. *Physical Review E*, 80(5):056103, 2009.

[Rib11] Pedro Ribeiro. *Efficient and Scalable Algorithms for Network Motifs Discovery*. PhD thesis, Faculty of Science of the University of Porto, June 2011.

[Ris18] Megan Risdal. Minneapolis incidents & crime. https://www.kaggle.com/mrisdal/minneapolis-incidents-crime, 2018. Accessed: 2018-03-02.

[RLH10] Luis EC Rocha, Fredrik Liljeros, and Petter Holme. Information dynamics shape the sexual networks of internet-mediated prostitution. *Proceedings of the National Academy of Sciences*, 107(13):5706–5711, 2010.

[RS10a] Pedro Ribeiro and Fernando Silva. Efficient subgraph frequency estimation with g-tries. In *International Workshop on Algorithms in Bioinformatics*, pages 238–249. Springer, 2010.

[RS10b] Pedro Ribeiro and Fernando Silva. G-tries: an efficient data structure for discovering network motifs. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1559–1566. ACM, 2010.

[RS14a] Pedro Ribeiro and Fernando Silva. Discovering colored network motifs. In *Complex Networks V*, pages 107–118. Springer, 2014.

[RS14b] 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.

[RS14c] 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.

[RSF$^+$15] Daniel I Rubenstein, Siva R Sundaresan, Ilya R Fischhoff, Chayant Tantipathananandh, and Tanya Y Berger-Wolf. Similar but different: dynamic social network analysis highlights fundamental differences between the fission-fusion societies of two equid species, the onager and grevy's zebra. *PloS one*, 10(10):e0138645, 2015.

[RSF17] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 385–394. ACM, 2017.

[RSK09] Pedro Ribeiro, Fernando Silva, and Marcus Kaiser. Strategies for network motifs discovery. In *e-Science, 2009. e-Science'09. Fifth IEEE International Conference on*, pages 80–87. IEEE, 2009.

[RSL10] Pedro Manuel Pinto Ribeiro, Fernando MA Silva, and Luís MB Lopes. Parallel calculation of subgraph census in biological networks. In *BIOINFORMATICS*, pages 56–65, 2010.

[RZ16] Ryan A Rossi and Rong Zhou. Leveraging multiple gpus and cpus for graphlet counting in large networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1783–1792. ACM, 2016.

[SAK$^+$09] Carl F Schaefer, Kira Anthony, Shiva Krupa, Jeffrey Buchoff, Matthew Day, Timo Hannay, and Kenneth H Buetow. Pid: the pathway interaction database. *Nucleic acids research*, 37(suppl 1):D674–D679, 2009.

[SAS18] Jorge Silva, David Aparício, and Fernando Silva. OTARIOS: Op-Timizing Author Ranking with Insiders/Outsiders Subnetworks. In *International Workshop on Complex Networks and their Applications*. Springer, 2018.

[SBS13] Erin R Shellman, Charles F Burant, and Santiago Schnell. Network motifs provide signatures that characterize metabolism. *Molecular BioSystems*, 9(3):352–360, 2013.

[SC88] Richard Schulz and Christine Curnow. Peak performance and age among superathletes: track and field, swimming, baseball, tennis, and golf. *Journal of Gerontology*, 43(5):P113–P120, 1988.

[SCKH04] Olaf Sporns, Dante R Chialvo, Marcus Kaiser, and Claus C Hilgetag. Organization, development and function of complex brain networks. *Trends in cognitive sciences*, 8(9):418–425, 2004.

[SCTM15] Yihan Sun, Joseph Crawford, Jie Tang, and Tijana Milenković. Simultaneous optimization of both node and edge conservation in network alignment via wave. In *International Workshop on Algorithms in Bioinformatics*, pages 16–39. Springer, 2015.

[SdV13] Volker S Schmid and Han de Vries. Finding a dominance order most consistent with a linear hierarchy: an improved algorithm for the i&si method. *Animal Behaviour*, 86(5):1097–1105, 2013.

[SGLP14] Kai Sun, Joana P Gonçalves, Chris Larminie, and Nataša Pržulj. Predicting disease associations via biological network analysis. *BMC bioinformatics*, 15(1):304, 2014.

[SH95] Barry M Staw and Ha Hoang. Sunk costs in the nba: Why draft order affects playing time and survival in professional basketball. *Administrative Science Quarterly*, pages 474–494, 1995.

[SKZ+15] Neil Shah, Danai Koutra, Tianmin Zou, Brian Gallagher, and Christos Faloutsos. Timecrunch: Interpretable dynamic graph summarization. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1055–1064. ACM, 2015.

[SM14] Vikram Saraph and Tijana Milenković. Magna: maximizing accuracy in global network alignment. *Bioinformatics*, 30(20):2931–2940, 2014.

[SMDYP16] Anida Sarajlić, Noël Malod-Dognin, Ömer Nebil Yaveroğlu, and Nataša Pržulj. Graphlet-based characterization of directed networks. *Scientific reports*, 6:35098, 2016.

[SOMMA02] Shai S Shen-Orr, Ron Milo, Shmoolik Mangan, and Uri Alon. Network motifs in the transcriptional regulation network of escherichia coli. *Nature genetics*, 31(1):64, 2002.

[SPK13] Comandur Seshadhri, Ali Pinar, and Tamara G Kolda. Triadic measures on graphs: The power of wedge sampling. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 10–18. SIAM, 2013.

[SS04] Falk Schreiber and Henning Schwobbermeyer. Towards motif detection in networks: Frequency concepts and flexible search. *Proc. Intl. Wsh. Network Tools and Applications in Biology (NETTAB'04)*, pages 91–102, 2004.

[SSDB95] Mark Schena, Dari Shalon, Ronald W Davis, and Patrick O Brown. Quantitative monitoring of gene expression patterns with a complementary dna microarray. *Science*, 270(5235):467–470, 1995.

[Ste97] Raymond T Stefani. Survey of the major world sports rating systems. *Journal of Applied Statistics*, 24(6):635–646, 1997.

[SVB$^+$11] Juliette Stehlé, Nicolas Voirin, Alain Barrat, Ciro Cattuto, Lorenzo Isella, Jean-François Pinton, Marco Quaggiotto, Wouter Van den Broeck, Corinne Régis, Bruno Lina, et al. High-resolution measurements of face-to-face contact patterns in a primary school. *PloS one*, 6(8):e23176, 2011.

[SW05] Thomas Schank and Dorothea Wagner. Finding, counting and listing all triangles in large graphs, an experimental study. In *International Workshop on Experimental and Efficient Algorithms*, pages 606–609. Springer, 2005.

[SXB08] Rohit Singh, Jinbo Xu, and Bonnie Berger. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proceedings of the National Academy of Sciences*, 105(35):12763–12768, 2008.

[TZY$^+$08] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: Extraction and mining of academic social networks. In *KDD'08*, pages 990–998, 2008.

[TZY$^+$17] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Citation Network Dataset. `https://aminer.org/citation`, 2017. [Online; accessed 14-September-2018].

[Ull76] Julian R Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM (JACM)*, 23(1):31–42, 1976.

[VCG14] Elizabeth S Vieira, José AS Cabral, and José ANF Gomes. How good is a model based on bibliometric indicators in predicting the final decisions made by peers? *Journal of Informetrics*, 8(2):390–405, 2014.

[VCM17] Vipin Vijayan, Dominic Critchlow, and T Milenković. Alignment of dynamic networks. *Bioinformatics*, 33(14):i180–i189, 2017.

[vdHSKP09] Martijn P van den Heuvel, Cornelis J Stam, René S Kahn, and Hilleke E Hulshoff Pol. Efficiency of functional brain networks and intellectual performance. *Journal of Neuroscience*, 29(23):7619–7624, 2009.

[VFMV03] Alexei Vázquez, Alessandro Flammini, Amos Maritan, and Alessandro Vespignani. Modeling of protein interaction networks. *Complexus*, 1(1):38–44, 2003.

[VKMM17] Vipin Vijayan, Eric Krebs, Lei Meng, and Tijana Milenković. Pairwise versus multiple network alignment. *arXiv preprint arXiv:1709.04564*, 2017.

[VM17a] V Vijayan and T Milenković. Multiple network alignment via multimagna+. *IEEE/ACM transactions on computational biology and bioinformatics*, 2017.

[VM17b] Vipin Vijayan and T Milenković. Aligning dynamic networks with dynawave. *Bioinformatics*, page btx841, 2017.

[VMCG09] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. On the evolution of user interaction in Facebook. In *Proc. Workshop on Online Social Networks*, pages 37–42, 2009.

[Vri98] Han de Vries. Finding a dominance order most consistent with a linear hierarchy: a new procedure and review. *Animal Behaviour*, 55(4):827–843, 1998.

[VSM15] Vipin Vijayan, Vikram Saraph, and T Milenković. Magna++: Maximizing accuracy in global network alignment via both node and edge conservation. *Bioinformatics*, 31(14):2409–2411, 2015.

[VY09] Nico W Van Yperen. Why some make it and others do not: Identifying psychological factors that predict career success in professional adult soccer. *The Sport Psychologist*, 23(3):317–329, 2009.

[Wer05] Sebastian Wernicke. A faster algorithm for detecting network motifs. In *International Workshop on Algorithms in Bioinformatics*, pages 165–177. Springer, 2005.

[Wer06] Sebastian Wernicke. Efficient detection of network motifs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4), 2006.

[WF94] Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.

[WHC12] Guangyu Wu, Martin Harrigan, and Pádraig Cunningham. Classifying wikipedia articles using network motif counts and ratios. In *Proceedings of the Eighth Annual International Symposium on Wikis and Open Collaboration*, page 12. ACM, 2012.

[WJD$^+$13] Jevin D West, Michael C Jensen, Ralph J Dandrea, Gregory J Gordon, and Carl T Bergstrom. Author-level eigenfactor metrics: Evaluating the influence of authors, institutions, and countries within the social science research network community. *Journal of the American Society for Information Science and Technology*, 64(4):787–801, 2013.

[WLR$^+$14] Pinghui Wang, John Lui, Bruno Ribeiro, Don Towsley, Junzhou Zhao, and Xiaohong Guan. Efficiently estimating motif statistics of large networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 9(2):8, 2014.

[WLY14] Pei Wang, Jinhu Lü, and Xinghuo Yu. Identification of important nodes in directed biological networks: A network motif approach. *PloS one*, 9(8):e106132, 2014.

[WR06a] Sebastian Wernicke and Florian Rasche. Fanmod: a tool for fast network motif detection. *Bioinformatics*, 22(9):1152–1153, 2006.

[WR06b] Sebastian Wernicke and Florian Rasche. FANMOD: a tool for fast network motif detection. `http://theinf1.informatik.uni-jena.de/motifs/`, 2006. [Online; accessed 20-September-2018].

[WS98] Duncan J Watts and Steven H Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, 1998.

[WS03] Stefan Wuchty and Peter F Stadler. Centers of complex networks. *Journal of Theoretical Biology*, 223(1):45–53, 2003.

[WSC16] Hao Wang, Hua-Wei Shen, and Xue-Qi Cheng. Scientific credit diffusion: Researcher level or paper level? *Scientometrics*, 109(2):827–837, 2016.

[WTZ+05] Tie Wang, Jeffrey W Touchman, Weiyi Zhang, Edward B Suh, and Guoliang Xue. A parallel algorithm for extracting transcriptional regulatory network motifs. In *Bioinformatics and Bioengineering, 2005. BIBE 2005. Fifth IEEE Symposium on*, pages 193–200. IEEE, 2005.

[WZZ+18] Pinghui Wang, Junzhou Zhao, Xiangliang Zhang, Zhenguo Li, Jiefeng Cheng, John CS Lui, Don Towsley, Jing Tao, and Xiaohong Guan. Moss-5: A fast method of approximating counts of 5-node graphlets in large graphs. *IEEE Transactions on Knowledge and Data Engineering*, 30(1):73–86, 2018.

[YAW17] Wenchao Yu, Charu C Aggarwal, and Wei Wang. Temporally factorized network modeling for evolutionary network analysis. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 455–464. ACM, 2017.

[YFCM15] Boyoung Yoo, Fazle Faisal, Huili Chen, and Tijana Milenković. Improving identification of key players in aging via network de-noising and core inference. *IEEE/ACM transactions on computational biology and bioinformatics*, 2015.

[YFCM17] Boyoung Yoo, Fazle Elahi Faisal, Huili Chen, and Tijana Milenković. Improving identification of key players in aging via network de-noising and core inference. *IEEE/ACM transactions on computational biology and bioinformatics*, 14(5):1056–1069, 2017.

[YH02] Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 721–724. IEEE, 2002.

[YMDD^{+}14] Ömer Nebil Yaveroğlu, Noël Malod-Dognin, Darren Davis, Zoran Levnajic, Vuk Janjic, Rasa Karapandza, Aleksandar Stojmirovic, and Nataša Pržulj. Revealing the hidden language of complex networks. *Scientific reports*, 4:4547, 2014.

[YMP15] Ömer Nebil Yaveroğlu, Tijana Milenković, and Nataša Pržulj. Proper evaluation of alignment-free network comparison methods. *Bioinformatics*, 31(16):2697–2704, 2015.

[ZHY^{+}13] Luming Zhang, Yahong Han, Yi Yang, Mingli Song, Shuicheng Yan, and Qi Tian. Discovering discriminative graphlets for aerial image categories recognition. *IEEE Transactions on Image Processing*, 22(12):5071–5084, 2013.

[ZQ05] Dongxiao Zhu and Zhaohui S Qin. Structural comparison of metabolic networks in selected single cell organisms. *BMC bioinformatics*, 6(1):1, 2005.

[ZTH^{+}10] Qiankun Zhao, Yuan Tian, Qi He, Nuria Oliver, Ruoming Jin, and Wang-Chien Lee. Communication motifs: a tool to characterize social communications. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1645–1648. ACM, 2010.

[ZYHW13] Lin Zhu, Zhu-Hong You, De-Shuang Huang, and Bing Wang. t-lse: a novel robust geometric approach for modeling protein-protein interaction networks. *PLoS One*, 8(4):e58368, 2013.