Network Alignment: Theory, Algorithms, and Applications

THÈSE N^{O} 7279 (2016)

PRÉSENTÉE LE 2 NOVEMBRE 2016
À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE POUR LES COMMUNICATIONS INFORMATIQUES ET LEURS APPLICATIONS 4
PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Ehsan KAZEMI

acceptée sur proposition du jury:

Prof. B. Faltings, président du jury
Prof. M. Grossglauser, directeur de thèse
Prof. M. Crovella, rapporteur
Prof. N. Kiyavash, rapporteuse
Prof. P. Thiran, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2016

”و من با تو سخن می گویم
نامت را به من بگو
دستت را به من بده
حرفت را به من بگو
قلبت را به من بده.

من ریشه های تو را دریافته ام
با لبانت برای همه ی لب ها سخن گفته ام
و دست هایت با دستان من آشناست.
در خلوت روشن با تو گریسته ام
برای خاطر زندگان
و در گورستان تاریک با تو خوانده ام
زیباترین سرودها را
زیرا که مردگان این سال
عاشق ترین زندگان بوده اند...“

احمد شاملو

To the memory of Naser who died for freedom!

به یاد ناصر که جان در راه آزادی داد!

Acknowledgements

I would never have been able to finish my dissertation alone. I owe my gratitude to all of the people who have made this journey possible; to the people whose names appear hereafter and to all those who are not mentioned in this acknowledgement.

First and foremost, I would like to express my sincerest thanks to my thesis advisor, Prof. Matthias Grossglauser. Matthias is a brilliant scientist, a great teacher and a smart person. I am always inspired by his broad vision, depth of knowledge and motivation. I am grateful to him for hiring me in LCA4. I owe him a lot for giving me the opportunity to investigate many research directions, and for pushing me always to explore further. Without his knowledge, advice, constant support and encouragement during all these years, finishing this thesis would have not been possible. I am deeply indebted to Matthias for all he has done for me!

I would like to thank my committee members for having accepted to assess my thesis and taking the time to read it through: Prof. Mark E. Crovella, Prof. Negar Kiyavash and Prof. Patrick Thiran. I would also like to thank Prof. Boi Faltings for being the president of the jury. It was a true honor for me to present my work to them.

It has been a great pleasure working at EPFL. I was very lucky to collaborate with many brilliant researchers. Working with them was a great experience and without their contribution, help and knowledge, this thesis would not exist in its present form. I am grateful to Prof. Patrick Thiran and Hamed Hassani for their great tips on the writing and presentation skills. I was fortunate to collaborate with them on several projects. Also, I have been very lucky to have the opportunity to work with Lyudmila Yartseva, Reza Shokri, Prof. Jean-Pierre Hubaux, Hassan Pezeshgi Modarres, Mohamed Kafsi, Vincent Etter and Lucas Maystre.

I would like to thank our lab secretaries Angela, Patricia and Holly for all their supports and for always being kind and helpful. I should especially thank Holly for “hollifying” our manuscripts with patience and a smile. I learned a lot from her. I am also grateful to ICSIL1 team, Marc-André, Stéphane and Yves, for the IT support.

I would like to extend my appreciation to my former and new colleagues and friends in LCA who supported me throughout these years. Special thanks go to: Brunella Spinelli, Christina Vlachou, Elisa Celis, Emtiyaz Khan, Farnood Salehi, Farid Naini, Italo Dacosta, Julien Herzen, Lucas Maystre, Lyudmila Yartseva, Mohamed Kafsi, Pedram Pedarsani, Sébastien Henri, Victor Kristof, Vincent Etter, William Trouleau and Young-Jun KO. Wish you all the best.

I have been very fortunate to be surrounded by amazing friends through these years. We shared many good memories outside my Ph.D. life. My friends were my family and they helped me in several occasions. I would like to thank my dearest friends in Lausanne. In particular,

many thanks go to Ali Tehranchi, Amin Shoaie, Arezoo Ghiassaleh, Ashkan Norouzi Fard, Azade Yazdan Yar, Farhnag Nabi, Farid Naini, Farnood Salehi, Farzan Jazaeri, Golzar Mesbah, Hamed Hassani, Hassan Pezeshgi Modarres, Hesam Setareh, Iman Honarvar Gheysary, Mahdi Aminian, Mahdi Khoramshahi, Majid Bastankhah, Mani Bastani Parizi, Maryam Ghasemi, Mehrdad Azizi Ghanad, Mohammad Karzand, Mohammad Parhikar, Mokhtar Bozorg, Nakisa Mousavi, Nooshin Mirzadeh, Payam Pakarha, Pedram Pad, Saghar Nabavi, Samira Asgari, Samira Kouchali and Vahid Babaei. I will miss all our parties, hikes, walks and runs.

Finally, I would like to say a heartfelt thank you to my mother, Paricheher, and my father, Ghasem, for their infinite love and support in the past thirty years. You are all I have and I will be forever thankful. Also, I am deeply grateful to my beloved sister, Setareh, and my lovely Brother, Naser, for all their love and encouragement. Love you so much!

This thesis is dedicated to the memory of my uncle, Naser, who fought and died for freedom.

Shiraz, Iran, 16 October 2016 E. K.

Abstract

Networks, as abstractions for representing complex relationships among entities, are central in the modeling and analysis of many large-scale human and technical systems, and they have applications in diverse fields such as computer science, biology, social sciences, and economics. Recently, network mining, i.e., statistical models and computational methods applicable specifically to network data, has been an active area of research. In this thesis, we study several related network-mining problems, from three different perspectives: the modeling and theory perspective, the computational perspective, and the application perspective. In the bulk of this thesis, we focus on network alignment, where the data provides two (or more) partial views of the network, and where the node labels are sometimes ambiguous. Network alignment has applications in social-network reconciliation and de-anonymization, protein-network alignment in biology, and computer vision.

In the first part of this thesis, we investigate the feasibility of network alignment with a random-graph model. This random-graph model generates two (or several) correlated networks, and lets the two networks to overlap only partially. Indeed, this model is parameterized by the expected node overlap t^2 and by the expected edge overlap s^2 of the two networks. For a particular alignment, we define a cost function for structural mismatch. We show that, if the average node-degrees of the random graphs grow as $s^{-2}t^{-1}\left(\log(n)+\omega(1)\right)$, the minimization of the proposed cost function (assuming that we have access to infinite computational power), with high probability, results in an alignment that recovers the set of shared nodes between the two networks, and that also recovers the true matching between the shared nodes. Our result shows that network alignment is fundamentally robust to partial edge-overlaps and node-overlaps, and this motivates us to look for network-alignment algorithms with low computational and memory complexity.

The most scalable network-alignment approaches use ideas from percolation theory, where a matched node-couple infects its neighboring couples that are additional potential matches. In the second part of this thesis, we propose a new percolation-based network-alignment algorithm that can match large networks by using only the network structure and a handful of initially pre-matched node-couples called seed set. We characterize a phase transition in matching performance as a function of the seed-set size. We also show the excellent performance of our algorithm over several real large-scale social networks.

In the third part of this thesis, we consider two important application areas of network mining in biology and public health. The first application area is percolation-based network alignment of protein-protein interaction (PPI) networks in biology. The alignment of biological networks has many uses, such as the detection of conserved biological network motifs, the prediction of protein interactions, and the reconstruction of phylogenetic trees. Network alignment can be used to transfer biological knowledge between species. We introduce a new global pairwise-network alignment algorithm for PPI networks, called PROPER. The PROPER algorithm shows higher accuracy and speed compared to other global network-alignment methods. We also extend PROPER to the global multiple-network alignment problem. We introduce a new algorithm, called MPROPER, for matching multiple networks, and we show that MPROPER outperforms the other state-of-the-art algorithms. Finally, we explore IsoRank, one of the first and most referenced global pairwise-network alignment algorithms. We develop an approximation algorithm that outperforms IsoRank by several orders of magnitude in time and memory complexity, despite only a negligible loss in precision.

Our second application area is the control of epidemic processes. We develop and model strategies for mitigating an epidemic in a large-scale dynamic contact network. More precisely, we study epidemics of infectious diseases by (i) modeling the spread of epidemics on a network by using many pieces of information about the mobility and behavior of a population, such as mobile call-data records; and by (ii) designing personalized behavioral recommendations for individuals, in order to mitigate the effect of epidemics on that network. We evaluate the effectiveness of our suggested recommendations over the Orange D4D dataset and show their benefits.

Key words: Network mining, network alignment, graph matching, random graph, percolation, protein-protein interaction, epidemic modeling

Résumé

Les réseaux, en tant qu'abstraction pour représenter des relations complexes entre entités, sont au cœur de la modélisation et de l'analyse de nombreux systèmes humains et techniques à grande échelle. Leurs applications sont très répandues dans divers domaines tels que l'informatique, la biologie, les sciences sociales et de l'économie. En conséquence, théorie des réseaux, c'est à dire, les modèles statistiques et les méthodes de calcul applicables spécifiquement au réseaux de données, est un domaine de recherche actif actuellement. Dans cette thèse, nous étudions plusieurs problèmes associés à l'extraction de réseaux à partir des trois points de vue suivants : celui de la modélisation et de la théorie, celui du calcul et celui de l'application. La majorité de cette thèse se concentre sur l'alignement de réseaux, où les données fournissent deux (ou plsueiurs) vues partielles de ceux-ci et où les étiquettes de nœuds peuvent être ambiguës. L'alignement de réseaux a des applications dans la réconciliation et la désanonymisation de réseaux sociaux, l'alignement de réseaux de protéines en biologie et la vision par ordinateur.

Dans la première partie de cette thèse, nous étudions la faisabilité de l'alignement de réseau selon un modèle de graphes aléatoires. Celui-ci génère deux (ou plusieurs) réseaux corrélés et leur permet de ne se chevaucher que partiellement. En effet, ce modèle est paramétré par le chevauchement prévu des nœuds t^2 et par le chevauchement prévu d'arêtes s^2 des deux réseaux. Pour un alignement particulier, nous définissons une fonction de coût pour l'inadéquation structurelle. Nous démontrons que la minimisation de celle-ci (en supposant que nous avons accès à une puissance de calcul infini), si la moyenne des degrés de nœuds des graphes aléatoires croît comme $s^{-2}t^{-1}\left(\log(n)+\omega(1)\right)$, résulte en un alignement qui récupère l'ensemble des nœuds partagés entre les deux réseaux avec une forte probabilité et qui couvre, également, la véritable correspondance entre ces nœuds. Notre résultat montre que l'alignement de réseaux est fondamentalement robuste aux arêtes partielles et aux chevauchements de nœuds. Cela motive la recherche d'algorithmes d'alignement de réseaux avec une faible complexité de calcul et de mémoire. Les approches les plus extensibles d'alignement de réseaux utilisent des idées de la théorie de la percolation, où un nœud-couple apparié infecte ses couples avoisinants comme des adéquations potentielles supplémentaires.

Dans la deuxième partie de cette thèse, nous proposons un nouvel algorithme d'alignement de réseaux basé sur la percolation, qui peut correspondre à de grands réseaux en utilisant uniquement leur structure, ainsi qu'une poignée de nœud-couples initialement pré-appariés, appelés graines. Nous caractérisons une transition de phase aux performances de couplage en fonction de la taille de l'ensemble des graines, sur le modèle de graphe aléatoire introduit précédemment. Nous montrons aussi l'excellente performance de notre algorithme sur plusieurs réseaux sociaux réels à grande échelle.

Dans la troisième partie de cette thèse, nous considérons deux domaines d'application importants de l'extraction de réseaux en biologie et en santé publique. Le premier domaine d'application est l'alignement en biologie de l'interaction protéine-protéine (PPI) des réseaux basé sur la percolation. L'alignement des réseaux biologiques a de nombreuses utilisations, telles que la détection de motifs conservés dans les réseaux biologiques, la prédiction d'interactions entre protéines, ainsi que la reconstruction d'arbres phylogénétiques. L'alignement de réseaux peut aussi être utilisé pour transférer des connaissances biologiques entre espèces. Nous introduisons un nouvel algorithme d'alignement global de réseaux par paires pour les réseaux PPI, appelés PROPER. L'algorithme PROPER permet une meilleure précision et une plus grande rapidité d'exécution par rapport aux autres méthodes d'alignement global de réseaux. Nous appliquons également PROPER au problème d'alignement global de réseaux multiples. Nous introduisons un nouvel algorithme pour coupler plusieurs réseaux, appelé MPROPER, et montrons que MPROPER surpasse les autres algorithmes de pointe sur les réseaux biologiques réels. Enfin, nous explorons IsoRank, l'un des premiers algorithmes, et l'un des plus référencés, d'alignement global de réseaux appariés. Nous développons un algorithme d'approximation qui surpasse IsoRank de plusieurs ordres de grandeur en temps et en mémoire, en dépit seulement d'une perte négligeable de précision.

Notre deuxième domaine d'application est le contrôle des processus épidémiques. Nous développons des stratégies pour atténuer une épidémie dans un réseau de contacts dynamiques à grande échelle. Plus précisément, nous étudions les épidémies de maladies infectieuses par : (i) La modélisation de la propagation d'épidémies sur un réseau en utilisant de nombreux éléments d'information sur la mobilité et le comportement d'une population, tels que les données d'appel téléphoniques ; and (ii) la conception de recommandations comportementales personnalisées aux particuliers, afin d'atténuer l'impact des épidémies sur ce réseau, tout en minimisant l'effet sur le cours normal de la vie quotidienne. Nous évaluons l'efficacité de nos recommandations sur le jeu de données d'Orange D4D et nous montrons leurs avantages.

Mots clefs : Théorie des réseaux, l'alignement de réseaux, interaction protéine-protéine, graphe aléatoire, percolation, épidémie

Contents

Acknowledgements i

Abstract (English/Français) iii

1 Introduction 1

- 1.1 Motivation . . . 1
- 1.2 The Modeling and Theory Perspective . . . 4
- 1.3 The Computational Perspective . . . 6
- 1.4 The Application Perspective . . . 7
  - 1.4.1 Network Alignment and PPI Networks . . . 7
  - 1.4.2 Modeling and Mitigating Epidemics . . . 9
- 1.5 Contributions . . . 10
  - 1.5.1 The Modeling and Theory Perspective . . . 10
  - 1.5.2 The Computational Perspective . . . 11
  - 1.5.3 The Application Perspective . . . 11

I The modeling and theory perspective **15**

2 A Model and Achievability Result for Network Alignment **17**

- 2.1 Model and Conditions for Perfect Matching . . . 19
  - 2.1.1 Problem Definition . . . 19
  - 2.1.2 Random Bigraph Model . . . 20
  - 2.1.3 Perfect Matchability and Structural Mismatch . . . 21
- 2.2 Proof of Theorem 3 . . . 23
  - 2.2.1 Node Partition . . . 25
  - 2.2.2 Edge Partition . . . 25
  - 2.2.3 Indicator Terms and Expected Values . . . 27
  - 2.2.4 Correlation Structure . . . 29
  - 2.2.5 Concentration . . . 31
- 2.3 Summary . . . 33

Appendix **35**

- 2.A Graph Matching and Edge Mismatch: Proof of Lemma 4 . . . 35
- 2.B Partition of Node Pairs into Chains and Cycles . . . 36

2.C Labeling the Indicator Terms 38
2.D Chernoff-Hoeffding Lemma 41

II The computational perspective 43

3 Percolation Graph Matching 45
3.1 Algorithms 48
3.1.1 Notation 48
3.1.2 NoisySeeds Algorithm 49
3.1.3 ExpandOnce Algorithm 52
3.2 Performance of Matching with Noisy Seeds 52
3.2.1 Sketch of the Proof 54
3.2.2 Proof of Theorem 16 54
3.3 ExpandWhenStuck Heuristic 62
3.4 Simulation Results 64
3.4.1 Experimental Results with Random Graphs 65
3.4.2 Experimental Results with Real Graphs 71
3.4.3 MapReduce implementation 77
3.5 Summary 78

Appendix 80
3.A Table of Notations 80

III The application perspective 81

4 Global Pairwise-Network Alignment 83
4.1 The PROPER Algorithm: Two Steps 84
4.1.1 The First Step: `SeedGeneration` 85
4.1.2 The Second Step: `MapPercolation` 86
4.2 Performance Measures 87
4.2.1 Structural and Functional Similarity Measures 88
4.2.2 Pathway Comparison Measures 90
4.3 Experimental Results 91
4.3.1 Structural and Functional Based Comparisons 93
4.3.2 The MapPercolation Algorithm and r 97
4.3.3 Aligning Biological Pathways 100
4.3.4 Execution Time 103
4.4 Discussion 104
4.4.1 Why the PROPER Algorithm? 105
4.5 Summary 107

Appendix **108**
4.A IsoBase: Experimental Results 108
4.B Pathways: Experimental Results 114

5 Global Multiple-Network Alignment **121**
5.1 Problem Definition 122
5.2 The MPROPER Algorithm 123
5.2.1 First Step: SeedGeneration 123
5.2.2 Second Step: MultiplePercolation 125
5.3 Performance Measures 127
5.4 Experimental Results 132
5.4.1 Comparisons 132
5.4.2 Computational Complexity 139
5.5 Why MPROPER? 140
5.5.1 Why SeedGeneration? 140
5.5.2 Why MultiplePercolation? 143
5.6 Summary 147

Appendix **148**
5.A GO Annotation: Statistics 148

6 IsoRank Node Similarities **153**
6.1 Problem Definition 153
6.2 Structural IsoRank Depends Only on Degrees 154
6.3 Fast Approximate IsoRank 157
6.4 Simulation Result 158
6.5 Summary 159

7 Modeling and Mitigating Epidemics **161**
7.1 Mobility Model 162
7.1.1 The Features of Mobility Model 162
7.1.2 Learning and Evaluating Mobility Models 163
7.2 Epidemic Model 165
7.2.1 Local Epidemic Dynamics 166
7.2.2 Implementation 167
7.3 Mobile Micro-measures 168
7.3.1 Concrete Micro-Measures 169
7.4 Empirical Evaluation 170
7.4.1 Model Parameters and Evaluation Metrics 170
7.4.2 Results 172
7.5 Summary 177

8 Conclusion and Perspectives 179
8.1 Open Questions 181
8.2 Future Research Directions 182

Bibliography 198

Curriculum Vitae 199

1 Introduction

1.1 Motivation

Human societies with billions of people, technological and economic systems, connected mobile devices, interacting genes and proteins in living organisms, and collections of activities of neurons in human brains are examples of the many complex systems that make up our daily life. Networks[1], as abstractions for representing complex relationships among entities, are at the heart of these complex systems. In a network, an entity or object is represented by a node, where some interacting or related pairs of nodes are connected by links. Any network can be modeled by a graph $G(V, E)$, where the set of vertices V represents the entities of the network and the set of edges E represents the links. Social networks such as Facebook, Google and Twitter, the network of interactions between proteins, genes and transcripts, the networks of connections between neurons in the brain, the power-grid network of generators, consumers and transmission lines, human-made technological networks such as the World-Wide Web, Internet and ad-hoc wireless networks, road networks, and trade networks are all samples of real-world networks that are the backbones of different complex systems.

We will never understand complex systems unless we develop a deep understanding of the complex networks underlying them. Studying complex networks, referred to as network science[2], has recently been an active area of research. The set of developed mathematical, computational, and statistical tools, which are applicable specifically to network data are the main building blocks of network science. Studying complex networks directly affects different fields such as computer science, biology, social sciences and economics. It has applications from personalized drug design to metabolic engineering. It can improve our security by fighting terrorism networks. It helps businesses to improve their marketing strategies and their influence on costumers. In this thesis, we study several related network-mining problems, from three different perspectives: the modeling and theory perspective, the computational perspective, and the application perspective.

---

[1] Also known as complex networks.

[2] Network mining

Despite the diversity of seemingly unrelated complex networks, many of them share common characteristics that are frequently observed in their experimental evaluations: heavy-tailed degree distribution, small diameter, high clustering, transitivity, community structure, homophily, node centrality, and small-world effect [11, 20, 33, 55, 142, 143, 192]. It seems that the structure and the dynamic of these networks follow a common set of fundamental laws and principles. These common topological and dynamical features motivate the research community to look for models to explain and to predict the universal properties of complex networks. Models enable the formulation of conjectures and provide the necessary explanations for different experimental observations. For example, many network models are presented by researchers to characterize properties of complex networks, to mimic the evolution and growth of these networks, and to reproduce many of their structural properties [32]. Real-world networks are modeled with random graphs models [42, 58], small-world models [198] and preferential attachment models [19]. To model networks with community structure, we can use the stochastic block model, a generative random model to produce graphs with known communities [78].

In order to predict the performance of network-mining algorithms, to compare them, to provide guarantees for the correctness of algorithms, and to understand the theoretical basis of these algorithms, we need to analyze them. Robust and rigorous models, in addition to identifying and explaining the unifying properties that are at the basis of real networks, help us to develop a fundamental understanding of network-mining algorithms. For this reason, one important modeling approach is to design mathematical models that capture key aspects of the input. With these models, we can analysis different types of algorithms. For example, network-sampling models create a small, but representative sample out of large complex networks in order to efficiently compute the graph measures such as shortest path, centrality, betweenness, clustering coefficient, assortativity and degree distribution [115, 197]. In the literature, graph sampling and graph sparsification models cover a wide range of approaches, from edge sampling and node sampling to graph traversal techniques such as random walks [23, 125, 179, 180]. Also, graph sampling methods are used to model structurally correlated networks, as it is convenient to assume these similar networks are sampled from an underlying hidden network [49, 109, 150]. Furthermore, it is beneficial to study, under mathematical models, information cascades [39, 206], network dynamics and epidemics [37, 48]. In this thesis, the first perspective from which we study complex networks is the modeling and theory perspective.

Network mining is more and more challenging with the rapid growth of many of the networks of practical interest. Indeed, real-world complex networks are large and some grow in size with time. Efficient algorithms have an important role in the understanding of static and dynamic properties of large complex networks. The algorithmic efficiency with respect to time (and memory) is an old and an essential question in the history of science. It is well emphasized by a famous quote from Ada Byron, *Memoir on the Analytic Engine*, dated back to 1843: “In almost every computation a great variety of arrangements for the succession of the processes is possible, and various considerations must influence the selections amongst them for the purposes of a calculating engine. One essential object is to choose that arrangement which shall tend to reduce to a minimum the time necessary for completing the calculation.”

The rise of interconnected multi-core processors and distributed systems has increased available computational power substantially and has brought researchers many new opportunities to process very large networks. Several distributed graph-parallel frameworks, such as Graphx [201], PowerGraph [68] and Pregel [127], have been successful in implementing large-scale data-intensive applications in past few years. Although there are numerous examples of the success of these platforms in providing scalable solutions for real-world problems, for many algorithms, distributed solutions induce substantial overhead and cause computations to be slower [132]. Generally, it is true that for an efficient algorithmic design “you can have a second computer once you’ve shown you know how to use the first one.”[3] In the second part of this thesis, we consider network-mining problems from a computational perspective. By borrowing tools from graph theory, algorithms, statistics and probability theory, we design effective, simple, parallelizable and robust computational methods with provable guarantees, based on sound mathematical foundations.

The main distinguishing feature of network science from graph theory is its application nature. We are interested in developing tools with excellent performances over real data, i.e., a good model and algorithm should provide deep insight about real complex systems. Network science has many direct applications in different disciplines. Understanding networks helps in the improvement of public health and public policy. Network inference problems play a major role in computational biology. Modern systems-biology approaches, which represent genes and proteins and their interactions as networks, provide new opportunities for studying and improving diagnoses and treatments of disease phenotypes and genetic variations associated with psychiatric diseases such as Autism Spectrum Disorders (ASD), Schizophrenia (SCZ) and Intellectual Disability (ID) [67, 73, 121, 147, 193]. Analyzing and predicting how complex networks function have many other important applications, such as finding connectivity patterns of neuronal firings in the brain [96, 122], modeling, detecting and mitigating the spread of epidemics in human networks [93, 164, 195], characterizing the diffusion of information in social networks [8, 160], sampling hidden and hard-to-reach populations [48], and influencing maximization for viral marketing and rumor control [38, 105, 117]. In the third part of this thesis, we study network-science problems from an application perspective. We use the developed models and algorithms (from the first two parts) to make inferences about real networks, mainly for biological networks.

Each of the modeling and theory perspective, the computational perspective and the application perspective is associated with a specific important aspect in studying complex networks. Indeed, the combination of these perspectives offers a collection of powerful models and tools for solving network-mining problems. We can also check the quality of our solutions in carefully designed experimental settings.

---

[3] A quote by Paul Barham.

In this thesis, we study how to merge information from different sources in order to make better inferences about a network and its properties from the modeling and theory perspective, the computational perspective and the application perspective. In many data analysis applications, information from different sources has to be merged into an integrated data model. This is notoriously difficult, because entity names or features from different sources are often unreliable and/or incompatible. When merging network data, one remedy is to rely on structural information, rather than on explicit vertex labels or vertex features to match two (or several) networks. More specifically, in the bulk of this thesis, we focus on network alignment[4], where the data provides two (or more) partial views of the network, and where the node labels are sometimes ambiguous. Network alignment has applications in social-network reconciliation and de-anonymization, protein-interaction network alignment in biology, and computer vision.

We investigate the network-alignment problem from three perspectives. In Chapter 2, we explore the feasibility of network alignment by using a random graph model. In Chapter 3, we give a new network-alignment algorithm. In Chapters 4, 5 and 6, we study network alignment from the application perspective. Also, in Chapter 7, by merging many pieces of information about the mobility and behavior of a population, we model the spread of an epidemic in a large-scale dynamic contact network.

In the rest of this chapter, we first discuss each of the modeling, computational and application perspectives briefly. Then, we summarize the main contributions of this thesis.

1.2 The Modeling and Theory Perspective

The network-alignment problem has received significant attention recently. It shows that social networks can be aligned by structural information [40, 41, 63, 99, 109, 140, 150, 202]. From a privacy point of view, research on network alignment provides many examples of networks that are vulnerable to structure-based de-anonymization attacks; these attacks are real threats to users' privacy [18, 89, 90, 138, 140, 151, 181, 199]. The main idea in these attacks is that the structural characteristics of users are uniquely identifiable across different networks. Protein-interaction network alignment enables us to find proteins with common biological functions in different species [103, 107, 176]. Also, network alignment has many applications in pattern recognition and machine vision [47], e.g., finding similar images in a database by matching segment-adjacency graphs [56, 108, 189].

Formally speaking, the network-alignment problem can be stated as follows: We are given two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$, where some couples of vertices $[i, j] \in V_1 \times V_2$ correspond to some unique underlying entity (e.g., a person). In general, not all vertices in $V_{1,2}$ have a counterpart in the other graph. The purpose of graph matching is to find the corresponding vertex couples in $V_1 \times V_2$, based on the topologies of the two networks and node features.

---

[4] Network alignment is also known as graph matching or network reconciliation in the literature.

For example, consider G_1 to be the network of users in Twitter, and G_2 the network formed by the contact relationships of Flicker users. The sets V_1 and V_2 only partially overlap in general, because some users have an account on one but not two services. The goal is to find the bijection between those users who have accounts on both Twitter and Flicker (users in $V_0 = V_1 \cap V_2$), based on the structural similarities of the two networks [140].

Network alignment can be viewed as a generalization of the classic graph-isomorphism problem, where we look for the *correct alignment* between the nodes of two structurally similar networks, without relying on node identifiers. While finding even the exact graph isomorphism can be complex in the worst case[5], in most of the real world scenarios the problem is much more difficult as the two graphs are subject to noise and uncertainties, and are not exactly isomorphic [47].

Recent works on network alignment have taken a modeling and information-theoretic angle, and shown conditions on the parameters of a random-graph model when perfect matching is possible [49, 88, 89, 91, 150]. The fundamental scaling results show the regions where network alignment is theoretically feasible. The feasibility of network alignment depends on two main assumptions: the structure of the two graphs and the side information that can be presented in different forms. A good random-graph model will serve as a basis for the structure of networks in information-theoretic results. For example, Pedarsani and Grossglauser [150] model the observed graphs as samples of a fixed underlying graph. They assume edges of each network are sampled from this hidden underlying network with a fixed probability s. The parameter s controls the structural similarity or the correlation of the two networks, e.g., for $s = 1$ the two networks are isomorphic and network alignment problem is equivalent to graph isomorphism. When $s < 1$, with high probability the two graphs are not isomorphic. Pedarsani and Grossglauser [150] find regions such that the correct alignment can still be identified, where unlimited computational power is assumed.

In this thesis, we study the feasibility of network alignment by using a new random-graph model. This model generates two (or several) correlated networks and permits the two networks to overlap only partially. More specifically, in Chapter 2, we introduce a simple parsimonious graph-sampling model called $G(n, p; t, s)$: Assuming we have a fixed hidden underlying network $G(n, p)$, nodes and edges of the two observed networks $G_{1,2}$ are sampled from G through independent node-sampling and edge-sampling processes with fixed probabilities t and s, respectively. This model is inspired by the model from [150]. By using this stochastic model, we find sufficient conditions for the identifiability of the true partial matching between the node sets of the two graphs.

---

[5] The class of graphs that appear the most challenging is thought to be the strongly regular graphs [178].

1.3 The Computational Perspective

In the first part of this thesis, we study network alignment from a modeling and theory perspective. Our results in Chapter 2, along with the recent results from [49, 88, 89, 91, 150], show that network alignment is fundamentally robust to partial edge and node overlaps. Although this feasibility result is true for the information-theoretic setting, where unlimited computational power is assumed, it motivates us to look for network-alignment algorithms with low computational and memory complexity.

Several heuristics have been proposed for network alignment [47]. A major class of the network-alignment algorithms could be formulated from the optimization point of view, based on a notion of graph edit distance [47, 61]. For example, it is possible to model network alignment as a quadratic assignment problem (QAP) [123], which is a well-known NP-complete problem [64], and try to find a linear programming (LP) relaxations for the QAP [24, 107]. Spectral methods are used as another approach for network alignment in the literature [35, 114, 176]. The main issue with the QAP and spectral approaches is that they are not scalable, and it is not possible to apply them over graphs with millions of nodes and edges. Another class of graph-matching algorithms uses semantic information (e.g., name, location and image of users) for the de-anonymization of social networks [128, 145]. Melnik et al. [133] introduce a similarity-flooding algorithm that matches nodes based on the spread of similarities in the network. Several machine-learning models are developed to match graphs by using the collected features about the nodes [7, 56, 145]. In general, performance guarantees and a characterization of feasible classes of the graphs to be matched by all these heuristics have been elusive in the literature.

It has been shown that structural similarity is the most important feature in the graph-matching process [75], and structure-based algorithms are more accurate and scalable [18, 109, 202]. Currently, the most scalable structure-base methods use ideas from percolation theory, where a matched node-couple infects its neighboring couples as additional potential matches. This line of work begins with the assumption that there is side information in the form of a *seed set* of “pre-matched" node couples, i.e., it assumes that a (small) subset of nodes across the two graphs are identified a priori. The matching is generated incrementally, starting from the seed couples and percolating to other node couples; for this reason, we refer to this class of algorithms as percolation graph-matching (PGM) methods.

The pioneering work by Narayanan and Shmatikov [140] is based on a seed-based heuristic PGM algorithm, which succeeded in de-anonymizing social networks with millions of nodes. They empirically observed a strong sensitivity of their algorithm to the seed-set size: If the seed set was too small, the percolation did not occur; when the seed-set size was increased, there was an abrupt change to a supercritical regime, where the algorithm succeeded in de-anonymizing a large fraction of the network. Yartseva and Grossglauser [202], for a random-bigraph model, prove the existence of such a phase transition in the seed-set size. A similar model is analyzed in [109] and is extended to scale-free graphs, under the assumption that seeds are dense (i.e., a constant fraction of nodes are seeds).

These PGM approaches have a basic feature in common: they incrementally build the matching between nodes of the two graphs. In every step, the set of node couples matched so far are used as evidence to match an additional node couple, if possible. The evidence for deciding which couple to match can take different forms, but it is obtained locally within the two graphs. For example, in [202], the rule is extremely simple: (i) every seed couple is considered matched; (ii) a node couple is matched if it has at least r *already matched* neighbours[6] and i, j are not already part of another matched couple. The recursive application of rule (ii) can, under some conditions, match all the nodes.

The analysis of iterative matching algorithms on large networks, by using tools from percolation theory and random graphs, has a rich history in the literature. For example, there is an important body of work on the design and analysis of gossip algorithms, whose purpose is to deliver a message to the whole network as efficiently as possible [97, 170, 196].

In PGM algorithms, initial seeds play an important role. The seed couples can be obtained in several ways, depending on the scenario: For example, some users of two different social networks might elect to make their identities public, which provides a set of known matches. Alternatively, methods have been proposed in the literature to identify plausible seed couples, based on structural graph features [18, 151] or manually through visual inspection [140].

In the second part of this thesis, we consider the network-alignment problem from a computational perspective. In Chapter 3, we give a new PGM algorithm with a dramatic reduction in the required size of the seed set. This algorithm can operate in a regime that needs far fewer starting seeds than previous approaches. By using ideas from bootstrap percolation theory, we rigorously characterize the phase transition in the seed-set size. We also show the excellent performance of our algorithm in matching several real social networks with over a million nodes, by using only a handful of seeds.

1.4 The Application Perspective

In the third part of this thesis, we consider two important applications of network mining in biology and public health. The first application area is percolation-based network alignment of protein-interaction networks in biology. The second application area is the control of epidemic processes.

1.4.1 Network Alignment and PPI Networks

Proteins are large biomolecules that carry out vital functions in living cells. Proteins rarely conduct their functions alone. Their interactions with the other biomolecules, especially other

---

[6] Two couples $[i, j]$ and $[i', j']$ are called neighbours if there is an edge (i, i') in E_1, and an edge (j, j') in E_2.

proteins, enables their diverse functionality [207]. Proteins conduct numerous functions, such as forming signaling networks and metabolic pathways, and regulating enzymatic activities, all via protein-protein interactions [207]. In this context, the term protein-protein interaction (PPI) stands for the mutual interactions between pairs of proteins.

PPI data are obtained by high-throughput experimental techniques such as yeast 2-hybrid [85], synthetic lethality [188] and co-immunoprecipitation coupled mass spectrometry [9]. The data are deposited in more than 100 PPI databases [146] such as BioGRID [36], the Molecular Interaction Database (MINT) [120], the Human Protein Reference Database (HPRD) [152], and IntAct [76]. Despite the large amount of PPI data, the detection of the protein pathways and protein complexes is challenging because many of the PPIs are noisy and non-reproducible.

PPI networks are a valuable source of information for understanding the evolution of protein interactions and system-level cellular processes. A comparative analysis of PPI networks provides insight into species evolution and information about evolutionarily-conserved biological interactions, such as pathways across multiple species [103, 172, 183, 207]. Network-alignment algorithms were introduced to compare PPI networks between two or more species.

The comparison of PPI networks, by network alignment, shows that there are identical interaction patterns between proteins with high sequence-similarity across different species [203]. For example, there are many common protein interactions between proteins in yeast networks and their corresponding protein orthologs in PPI networks of worms [131]. Because functional interactions are conserved across species and false positives are unlikely to occur in multiple species, network alignment can increase the confidence level of an observed interaction in a database [171].

PPI-network alignment has many applications in areas such as the detection of new pathways and of conserved motifs, the prediction of the functions of proteins, orthology detection, drug design, protein-protein interaction prediction and phylogenetic tree reconstruction [111, 175].

Generally, PPI-network alignment methods assume that two functional ortholog proteins on two different PPI networks are likely to interact with proteins in the corresponding networks that are functionally orthologs themselves [157, 177, 207]. Following this line of thought, local network-alignment (LNA) and global network-alignment (GNA) methods are the main approaches for aligning PPI networks [57, 59, 207]. The LNA algorithms search for small but highly conserved subnetworks (e.g., homologous regions of biological pathways or protein complexes) between species, whereas GNA algorithms try to align all (or most of) the proteins to find large subgraphs that are functionally and structurally conserved over all the nodes in the two networks [57, 59, 207].

There are two main classes of GNA algorithms: (i) pairwise-network alignment, and (ii) multiple-network alignment. The multiple-network alignment methods produce alignments consisting of aligned clusters (or tuples) with nodes from several networks [57, 59]. Also, multiple-network alignment algorithms are classified into two categories of one-to-one and many-to-many algorithms. In the first category, each node from a network can be aligned with at most one node from another network. In the many-to-many category, one or several nodes from a network can be aligned with one or several nodes from another network.

Recently, Meng et al. [135] compared the performances of several LNA and GNA algorithms in predicting new functionalities of proteins. Their result indicates that these two general classes of algorithms produce very different predictions, but they are still complementary to each other. This highlights the need for both LNA and GNA algorithms that produce high-quality alignments.

In this thesis, we investigate the GNA problem. In Chapter 4, we introduce a new percolation-based pairwise-network algorithm for PPI networks; it is called PROPER. In Chapter 5, we introduce a new algorithm, called MPROPER, for aligning multiple networks. In Chapter 6, we explore IsoRank [119, 175, 176], one of the first and most referenced global network-alignment algorithms, and develop an approximation algorithm for it.

1.4.2 Modeling and Mitigating Epidemics

Epidemics of infectious diseases are among the largest threats to the quality of life and to the economic and social well-being of developing countries. In several occasions throughout human history, outbreaks of diseases have had disastrous effects on societies: the outbreak of bubonic plague killed between 30% to 50% of Europe's population in the 1300s [70]; the epidemics, caused by the arrival of Europeans, had harmful consequences for the native Americans' civilizations [53]; in 1918, the Spanish flu pandemic caused an estimated 50 million deaths worldwide [186]; more recently, the 2002–2003 SARS pandemic that originated in Hong-Kong and spread worldwide caused the death of 774 [200].

Modeling and effectively mitigating the spread of infectious diseases is one of the high priorities of global public-health policies and has been a long-standing goal. Epidemic modeling enables scientists to predict epidemic outbreaks and to find strategies for decreasing mortality rates, along with the costs to the economy [60, 65, 81, 83]. In modeling epidemics, biological issues mix with social ones and make it more challenging. As a classic example of epidemic modeling, Kermack and McKendrick [106] in their seminal work introduce a SIR model with three distinct classes of populations: susceptible (S), infective (I) and recovered (R). This simple yet powerful model is very popular for modeling the evolution of epidemics in populations. Hethcote [77] reviews different extensions of this model such as SIS, SI and SEIS, as well as threshold theorems involving measures such as the reproduction number.[7]

In a large-scale dynamic contact network, the mobility of individuals plays a crucial role and is the source of disease spread among different geographical areas [16, 27, 166, 190]. Therefore, in order to improve the realism of epidemic models, we need to build an accurate and data-

---

[7] Reproduction number is the average number of secondary infections caused by an infected individual when in contact with a susceptible population.

driven mobility model. This mobility network is used to model and predict the spread of epidemics and to design strategies for weakening the links in the contact network that form the path through which the epidemic spreads. The SIR models, which incorporate mobility between regions, are examples of powerful tools for designing and testing different strategies to control epidemics [44, 159]. Nowadays, by the advent of mobile technologies, the call-data records (CDRs) collected by cellular services provide a valuable source of information for large scale empirical-validation of mobility models; we use these mobility models to create epidemic models with a high predictability power [19, 25, 26, 69, 84, 184].

Effective measures against an epidemic require an accurate and up-to-date assessment of the situation: a very rapid response and a strong coordination. They require colossal organizational efforts under tight time constraints. To this day, there is no uncontested way of preventing epidemics in general. Traditional epidemic mitigation-methods consist of heavy, top-down approaches such as blockades, quarantines or large-scale vaccination campaigns [136, 167, 174, 182, 204, 205]. Although the arsenal of measures against epidemics is well-established, these measures are costly and insufficient. These methods have several drawbacks: they are difficult and slow to put into place, and can be expensive and also freedom-restrictive. It is clear that any improvement would have a tremendous impact and translate into significant welfare gains.

One of the most important and fascinating applications of network science is the modeling and predicting epidemics, and the suggesting of strategies for mitigating them. In Chapter 7, as an alternative to the traditional methods, we suggest that access to mobile technology and information about human-contacts network at large scales could enable a much richer and sophisticated set of mitigation measures for human-mediated epidemics.

1.5 Contributions

In this thesis, we address three important aspects of network mining: (i) the modeling and theory perspective, (ii) the computational perspective, and (iii) the application perspective. We design and answer challenging questions regarding these aspects. The following is the list of the main contributions of this thesis.

1.5.1 The Modeling and Theory Perspective

From the modeling an theory perspective, we make the following contributions in Chapter 2.

- To generate two Erdős-Rényi random graphs whose vertex sets overlap only partially, we extend the random-bigraph model developed by Pedarsani and Grossglauser [150] The model has two parameters (t and s) to control vertex overlap and edge overlap, respectively.
- We formulate network alignment as an optimization problem over the space of all

possible partial matchings between the two node sets. Our main information theoretic result is a sufficient condition on the graph density (or average vertex degree) and on the amount of noise for perfect matching. We define a cost function for structural mismatch. We show that minimizing the proposed cost function, with high probability, identifies the true matching.

1.5.2 The Computational Perspective

From the computational perspective, we make the following contributions in Chapter 3.

- We develop a new graph-matching algorithm called `ExpandWhenStuck`. The distinguishing feature of this algorithm is that, in comparison to state-of-the-art algorithms [109, 202], it requires a dramatically smaller number of seeds. It is able to match, by using only a handful of seeds, real social-networks with over a million nodes, as well as various types of random graphs (for example, Barabási–Albert [21], Chung–Lu [42] and Erdős–Rényi [58] graphs).
- We analyze the performance of a simplified version of the `ExpandWhenStuck` algorithm (called `ExpandOnce`) by using the random-bigraph model that is introduced in Chapter 2. The simplification needed to make the analysis tractable concerns the generation of candidate couples: Although `ExpandWhenStuck` dynamically percolates from unmatched candidate couples whenever necessary, we can rigorously analyze only a slightly more restrictive setting, where this occurs only once at the outset. Specifically, the `ExpandOnce` algorithm expands the seed set into a larger set that includes many incorrect couples; a second algorithm (called `NoisySeeds`) then percolates from this latter set.
- We demonstrate a phase transition in the number of required seeds for `NoisySeeds`, as a function of the network size, overlap between the two graphs, and structural similarity. We prove that the `NoisySeeds` algorithm is robust to partial node-overlap. More precisely, we prove that `NoisySeeds` naturally filters out the nodes without counterparts in the other graph, and that it correctly matches the rest.

1.5.3 The Application Perspective

From the application perspective, we consider the two problems of PPI-network alignment, and modeling and mitigating epidemics.

Global Pairwise-Network Alignment

In Chapter 4, we study the global pairwise-network alignment problem.

- We design a new percolation-based pairwise-network alignment algorithm for PPI networks; it is called PROPER.

- We show the excellent performance of PROPER (in terms of both accuracy and speed), compared to several state-of-the-art algorithms.
- We introduce a new measure for evaluating the performance of algorithms in aligning biological pathways. We show that PROPER can detect large portions of conserved biological pathways between species.
- By using a simple parsimonious evolutionary-model (similar to the model we introduce in Chapter 2), we explain why PROPER performs well with respect to different comparison criteria that are used for evaluating pairwise alignments.

Global Multiple-Network Alignment

In Chapter 5, we study the global multiple-network alignment problem.

- For aligning multiple PPI-networks, we introduce a new extension of the PROPER algorithm; it is called MPROPER. The MPROPER algorithm has two main steps: (i) `SeedTupleGeneration` and (ii) `MultiplePercolation`. In the first step, to generate an initial set of seed clusters (or tuples), the `SeedTupleGeneration` algorithm uses only protein sequence similarities. In the second step, to align remaining unmatched nodes, the `MultiplePercolation` algorithm uses network structures and the seed tuples generated from the first step.
- We show that, with respect to different evaluation criteria, MPROPER outperforms the other state-of-the-art algorithms.
- We present a graph-sampling model (as a generalization of the model we introduce in Chapter 2) for generating k correlated networks. By using this model, we prove that, if initially enough seed tuples are provided, the `MultiplePercolation` algorithm correctly aligns almost all the nodes.

IsoRank

In Chapter 6, we make the following contributions.

- We explore IsoRank, one of the first and most referenced global network-alignment algorithms [119, 175, 176].
- We show that when IsoRank similarity depends only on network structure, the similarity of two nodes is only a function of their degrees.
- We develop an approximation algorithm (by using ideas from [14, 71]) that outperforms IsoRank in time and memory complexity by several orders of magnitude, despite only a negligible loss in precision.

Strategies for Mitigating Epidemics

In Chapter 7, we make the following contributions.

- We model and predict the spread of epidemics in a large-scale dynamic contact network, by using many pieces of information about the mobility and behavior of the population, such as call-data records.
- We design personalized behavioral recommendations to individuals, in order to mitigate the effect of epidemics on that network, and we minimize the side-effects on the normal course of daily life.
- We evaluate these strategies over the Orange D4D dataset and show the benefit of these measures, even if only a fraction of the population participates.

The modeling and theory perspective Part I

2 A Model and Achievability Result for Network Alignment

Network alignment (or graph matching) is the problem of identification of a bijection between the (full or partial) vertex sets of two networks. Finding such an alignment is particularly important and challenging when only the structures of the two graphs are available, i.e., the two graphs can be considered *unlabelled*. Obviously, the availability of any side-information, such as node or edge attributes, makes the problem easier.

In the first part of this thesis, we investigate the feasibility of network alignment. We establish an information-theoretic characterization of the region, where the alignment between two correlated networks with overlapping vertex sets is possible. Concretely, we explore to what extend network parameters can be relaxed (in the form of information-theoretic thresholds) such that the exact recovery of node correspondences is feasible, given unbounded computational resources.

From an information-theoretic perspective, Pedarsani and Grossglauser [150] show conditions on the parameters of a random-bigraph model when perfect matching is possible. Their model generates two correlated $G(n, ps)$ random graphs, with a similarity parameter $0 \leq s \leq 1$. When $s < 1$, with high probability the two graphs are not isomorphic, but [150] establishes a threshold function for p such that the correct alignment can nevertheless be identified. The threshold is proportional to $c(s)\log(n)/n$, where the function $c(s)$ is a penalty due to the dissimilarity of the two graphs. In summary, their work shows conditions where graph structure fundamentally contains sufficient information for finding alignments, if computational resources are unlimited. Cullina and Kiyavash [49] improve the achievability bound of [150] by a factor of $\frac{4(2-s)}{s}$. Also, they show that there is only a gap of factor 2 between their bound and a converse threshold [49]. Furthermore, Cullina et al. [50] investigate the problem of network alignment for the class of stochastic block models (SBMs). Ji et al. [89, 91], by using the same model as [150], study the effect of seed information on both perfect matchability and partial matchability[1] of networks.

The models from [49, 88, 89, 91, 150] make several strong and unrealistic assumptions, in-

---

[1] Matching a $1-\epsilon$ fraction of all the nodes

cluding that the vertex sets of the two graphs are of the same size, and that a full matching between these sets can be found. In most practical scenarios, node overlap would be only partial. For example, when reconciling two social networks, we should be able to permit users of one network to not be users of the other; or an adversary should take into account that only a subset of nodes might be included in a privately released dataset.

To the best of our knowledge, it is an open question as to what extent partial overlap of the node sets hampers the feasibility of network alignment. In this thesis, we address this question by using a random-graph model. This model generates two (or several) correlated networks and permits the two networks to overlap only partially. Indeed, this model is parameterized by the expected node-overlap t^2 and by the expected edge-overlap s^2 of the two networks. For a particular alignment, we define a cost function for structural mismatch. We show that, if the average node degrees of the random graphs grow as $s^{-2}t^{-1}\left(\log(n)+\omega(1)\right)$, the minimization of the proposed cost function (assuming that we have access to infinite computational power), with high probability, results in an alignment that recovers the set of shared nodes between the two networks; this minimization also recovers the true matching between the shared nodes. Our result shows that network alignment is fundamentally robust to partial edge and node overlaps, hence an motivation to look for network-alignment algorithms with low computational and memory complexity.

In this chapter, we make the following contributions.

(a) First, we extend the random-bigraph model of [150] in order to generate two Erdős-Rényi random graphs whose vertex sets overlap only partially. The model has two parameters (t and s) for controlling vertex overlap and edge overlap, respectively.

(b) Second, our main result is a sufficient condition for the graph density (or average vertex degree) and for the amount of noise for perfect matchability. A perfect matching amounts to (i) filtering out nodes without counterparts in both G_1 and G_2, and (ii) correctly match the remaining nodes that are present in both graphs.

(c) Third, we formulate network alignment as an optimization problem over the space of all possible partial matchings between the two node sets. We show scaling conditions such that minimizing a cost function identifies the true matching with high probability. Although the optimization formulation does not lend itself to a scalable algorithm, our results delineate the boundary between the fundamentally possible and impossible.

This chapter is structured as follows. In Section 2.1, we introduce our model for generating correlated graphs with partial vertex overlap, and we state our main result. In Section 3.2.2, we prove the result. Section 2.3 concludes the chapter. Some technical details are relegated to appendices.

2.1 Model and Conditions for Perfect Matching

In this section, we first formally state the graph-matching problem. Then, to formalize a partial overlap in the vertex sets of the graphs, we present a random-bigraph model that generates two correlated Erdős-Rényi random graphs. We introduce a cost function for quantifying the structural mismatch, for a given candidate alignment, between the two graphs. Finally, we state the main theorem of this chapter. Our theorem shows that under surprisingly mild conditions, minimizing this cost function, with high probability, finds the correct matching.

2.1.1 Problem Definition

Assume we are given two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ that represent, for example, two social networks (e.g., G_1 is Facebook, G_2 is LinkedIn). We know that some users have profiles in several social networks. In this chapter, we study the graph-matching problem that refers to inferring the alignment of the common users of the networks G_1 and G_2 by structural information only.

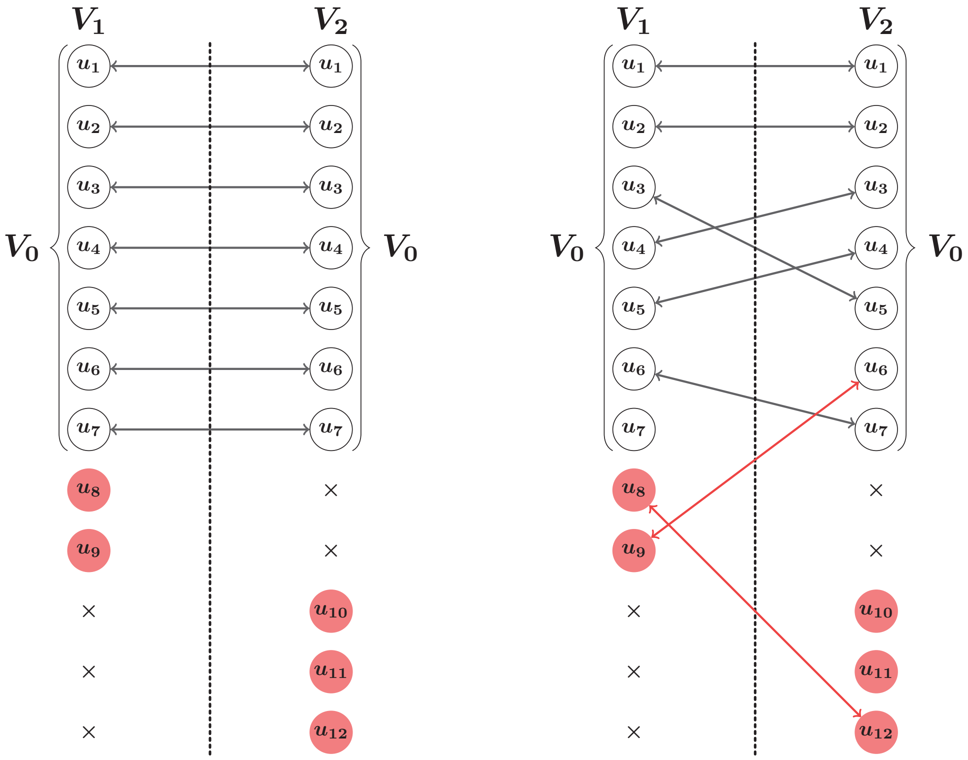
The graph-matching problem is defined formally as follows. Given the two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$, the goal is to find a matching between the nodes in $V_0 = V_1 \cap V_2$, where V_0 (we define $n_0 = |V_0|$) is the set of vertices common to both graphs. We call this true hidden matching π_0. We assume that, without loss of generality, $V_{1,2} \subset [n] = \{1, \ldots, n\}$ and denote $n_1 = |V_1|$, $n_2 = |V_2|$. Next, we define the set of all possible matchings Π from graph G_1 to G_2.

Definition 1. Π is the set of all *partial matchings (or alignments)* π from the vertex set V_1 to V_2. A partial matching π is a subset of $V_1 \times V_2$ such that any node in $V_1 = \{1, \ldots, n_1\}$ and $V_2 = \{1, \ldots, n_2\}$ is matched to at most one node in the other graph.

Thus, the *identity* hidden matching π_0 is the set of couples of nodes that are present in both graphs G_1 and G_2, i.e., $\pi_0 = \{[u, u] : u \in V_0\}$. Furthermore, if node $v_1 \in V_1$ is matched to node $v_2 \in V_2$, we say $v_2 = \pi(v_1)$ and $v_1 = \pi^{-1}(v_2)$. For a pair of nodes $e = (u, v)$ we define $\pi(e) = (\pi(u), \pi(v))$. Let us define $V_{1,2}(\pi)$ as the sets of vertices in $V_{1,2}$ that are matched by π, and $E_{1,2}(\pi)$ as the sets of matched edges (an edge is matched if both endpoints are matched). For a node u, we say $\pi(u)$ is *null* (denoted by $\pi(u) = \emptyset$) if either u is not present ($u \notin V_1$) or u is not matched (i.e., $u \in V_1$ but $u \notin V_1(\pi)$). Similarly, for a node v, we say $\pi^{-1}(v)$ is *null* ($\pi^{-1}(v) = \emptyset$) if $v \notin V_2$ or $v \notin V_2(\pi)$. For a pair $e = (u, v)$, $\pi(e)$ is defined to be null (denoted by $\pi(e) = \emptyset$) if either $\pi(u) = \emptyset$ or $\pi(v) = \emptyset$. Similarly, $\pi^{-1}(e) = \emptyset$ if either $\pi^{-1}(u) = \emptyset$ or $\pi^{-1}(v) = \emptyset$.

Definition 2. For a matching π, we define (i) $|\pi|$ as the size of matching π, (ii) l as the number of correctly matched couples of the form $[i, i]$, and (iii) $k = |\pi| - l$ as the number of incorrectly matched couples. Let Π_k^l represent a class of matchings of size $|\pi| = l + k \le \min\{n_1, n_2\}$ with l correctly matched couples. Note that the sets Π_k^l partition the set Π of all partial matchings.

Figure 2.1 shows two examples of alignments: (i) the identity matching $\pi_0 \in \Pi_0^7$, and (ii) the matching $\pi \in \Pi_6^2$ from V_1 to V_2.



$(i)\ \pi_0 \in \Pi_0^7$ $(ii)\ \pi \in \Pi_6^2$

Figure 2.1 – Examples of two alignments: (i) The true matching $\pi_0 \in \Pi_0^7 = \{[u_1, u_1], \ldots, [u_7, u_7]\}$, and (ii) the matching $\pi \in \Pi_6^2$. White nodes are common to both graphs, whereas red nodes are present in only one but not the other.

2.1.2 Random Bigraph Model

We study the graph-matching problem under a random bigraph model. This stochastic model assumes that graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ are sampled from an Erdős-Rényi (i.e., $G(n, p)$) graph [58] $G(V, E)$ as follows: First, the unseen generator graph $G(V, E)$ is sampled from the probability space of $G(n, p)$ random graphs with n nodes, where each of the $\binom{n}{2}$ possible edges exists independently with probability $0 < p \leq 1$. Second, vertex sets $V_{1,2}$ are sampled independently from the vertex set V with probability t, i.e., $\mathbb{P}(u \in V_1) = \mathbb{P}(u \in V_2) = t$ for all $u \in V$. Third, edges of graph $G_1(V_1, E_1)$ are sampled from those edges of graph G whose both endpoints are sampled in V_1 by independent edge sampling processes with probability s. The edges of graph $G_2(V_2, E_2)$ are generated similarly. Formally, for an edge $e = (u, v) \in E$ we have $\mathbb{P}(e \in E_1 | u, v \in V_1) = \mathbb{P}(e \in E_2 | u, v \in V_2) = s$.

We refer to this model as the $G(n, p; t, s)$ bigraph model. For this model, we were inspired by [150]; we consider a more challenging and realistic scenario, where the two graphs have partially overlapping vertex sets (this is modeled by the node sampling process). Figure 2.2 provides a schematic overview of the $G(n, p; t, s)$ model.

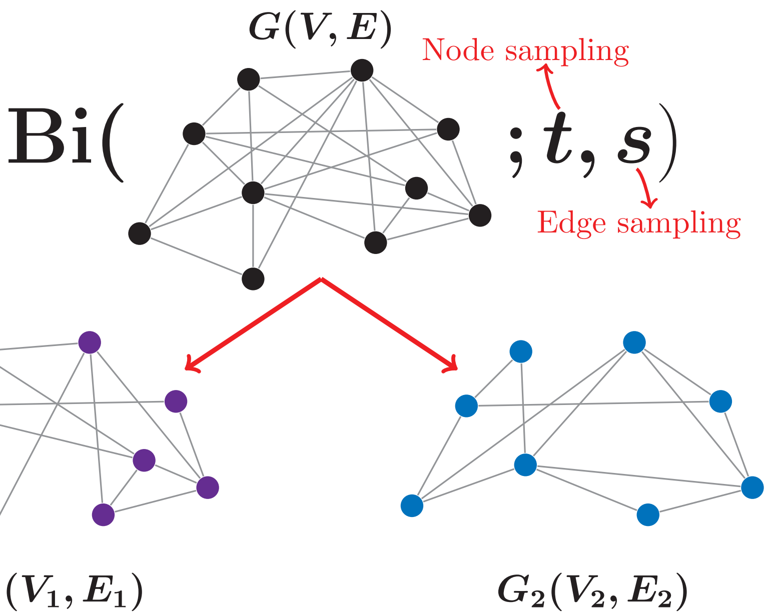


Figure 2.2 – The $G(n,p;t,s)$ random bigraph model. The two graphs $G_1(V_1,E_1)$ and $G_1(V_1,E_1)$ are sampled from the generator graph $G(V,E)$ through node sampling (with probability t) and edge sampling (with probability t) processes. Also, we assume that the hidden underlying graph $G(V,E)$ is sampled from the probability space of $G(n,p)$ random graphs.

2.1.3 Perfect Matchability and Structural Mismatch

We now define a cost function that, for a given partial matching π, quantifies the structural mismatch between the two graphs $G_1(V_1,E_1)$ and $G_2(V_2,E_2)$. The cost function has two terms Φ_π and Ψ_π:

- Mismatched edges:

$$\Phi_\pi = \sum_{e\in E_1(\pi)} \mathbb{1}_{\{\pi(e)\notin E_2\}} + \sum_{e\in E_2(\pi)} \mathbb{1}_{\{\pi^{-1}(e)\notin E_1\}}.$$

- Unmatched edges: $\Psi_\pi = \Psi_\pi^1 + \Psi_\pi^2$, where Ψ_π^1 and Ψ_π^2 are the number of unmatched edges in E_1 and E_2, respectively. More precisely, we define

$$\Psi_\pi^1 = |\{e \in E_1 \setminus E_1(\pi)\}| \text{ and } \Psi_\pi^2 = |\{e \in E_2 \setminus E_2(\pi)\}|.$$

The cost function is a weighted sum of Φ_π and Ψ_π:

$$\Delta_\pi = \Phi_\pi + \alpha\Psi_\pi. \tag{2.1}$$

Our approach consists in minimizing the cost function Δ_π over all possible partial matchings π. There is a trade-off between the two cost terms (2.1): adding node couples to the matching π cannot decrease Φ_π (and it can increase even for correct couples because of edge sampling), while Ψ_π cannot increase. The parameter α controls this trade-off: with $\alpha = 0$, the trivial empty matching minimizes Δ_π; with $\alpha > 1$ the optimal matching is always of the largest possible size $\min\{n_1, n_2\}$, because the increase in Φ_π when adding a couple to π is smaller than the decrease in $\alpha\Psi_\pi$. Below, we identify constraints on α and provide an appropriate value such that with high probability, matching found by minimizing Δ_π is the correct partial matching π_0.

We now state the main result of the chapter.

Theorem 3. *In the $G(n, p; t, s)$ bigraph model with $\frac{6144 \log n + \omega(1)}{ns^3 t^2} = p \ll 1$, for each p, t and s there exists a value of α such that with high probability*

$$\pi_0 = \underset{\pi}{\operatorname{argmin}}\, \Delta_\pi. \tag{2.2}$$

Expressed in terms of the expected degree $npst$ of the two observable graphs $G_{1,2}$, the threshold is $\log(n)/s^2 t$ for perfect matchability. Before proving Theorem 3, we provide some context for the result.

The dependence on n is tight. To see this, consider the intersection graph $G_0 = G(V_0, E_0)$, where $V_0 = V_1 \cap V_2$ and $E_0 = E_1 \cap E_2$. Its expected degree is $nps^2 t^2$.[2] If this is asymptotically less than $\log nt^2$, then G_0, with high probability, has symmetries, i.e., the automorphism group of G_0 is not trivial (that, in fact, stem from the isolated vertices [33]). In this case, the correct matching cannot be determined uniquely. To see this, assume that an oracle reveals, separately for G_1 and for G_2, the set of nodes and edges without counterpart. These sets contain no useful information for estimating π_0 over the common nodes, due to the independence assumptions in the model. Essentially, given an oracle, G_0 is a sufficient statistic for π_0, whose symmetries would preclude inferring π_0.

Based on this argument, the dependence on t is tight, where there is a gap of a factor of s between the achievability result in Theorem 3 and the trivial lower bound based on G_0. It is not clear whether the upper or lower bound is loose with respect to s.

With $t = 1$, we can recover the achievability result of Pedarsani and Grossglauser [150] up to a constant. Note that this is not trivial, as their problem formulation minimizes a cost function[3] over the set $\{\Pi_k^l : k + l = n\}$, where here we minimize over the larger set $\{\Pi_k^l : k + l \leq n\}$. Then, our result shows that there is asymptotically no penalty for not knowing *a priori* the overlap set V_0. Also, we can derive the maximum a posteriori (MAP) rule for the $G(n, p; t, s)$ model

---

[2] To be precise, $(n-1)ps^2 t^2$; we sometimes omit lower-order terms for readability.

[3] Identical to ours with $\alpha = 0$.

with $t = 1$. The MAP rule is to choose the permutation $\hat{\pi}$ from the symmetric group S_n[4] given by

$$\hat{\pi}_{\text{MAP}} = \underset{\pi \in S_n}{\operatorname{argmax}} \mathbb{P}[\pi | G_1, G_2].$$

The MAP rule can be equivalently stated in terms of a simpler concept, by the following lemma.

Lemma 4. *We have*

$$\hat{\pi}_{\text{MAP}} = \underset{\pi}{\operatorname{argmax}} \mathbb{P}[\pi | G_1, G_2] = \underset{\pi}{\operatorname{argmin}} \Phi_\pi. \tag{2.3}$$

The proof of Lemma 4 is given in Appendix 2.A. Lemma 4 and Theorem 3 (along the result of [150]) show that the MAP estimator recovers the true alignment with high probability for $t = 1$.

The cost function Δ_π with $\alpha = 1$ is similar to a simple graph edit distance between G_1 and G_2. Suppose we wanted to find the cheapest way to transform the unlabeled graph G_1 into G_2 through edge additions and deletions. Then the number of operations is exactly Δ_π. Our conditions on α (discussed in detail within the proof) show that minimizing this edit distance does not work. Instead, the trade-off between penalizing mismatched mapped edges and unmapped edges needs to be controlled more finely through an appropriate choice of α, which depends on p and s.

The result is for the Erdős-Rényi random-graph model with uniform sampling. This parsimonious model is a poor approximation of most real networks that have salient properties not shared with random graphs (skewed degree distribution, high clustering, community structure, etc.). However, we conjecture that network alignment for random graphs is harder than for real graphs, because the structural features of real networks make nodes more distinguishable in these networks than in random graphs. Our results suggest that, even for the difficult case of random graphs, network alignment is fundamentally easy given sufficient computational power.

2.2 Proof of Theorem 3

We provide a brief sketch followed by the detailed proof. Let S be the number of matchings $\pi \in \Pi$ such that $\Delta_\pi - \Delta_{\pi_0} \leq 0$. Following the Markov inequality, as S is a non-negative integer-valued random variable, we have $\mathbb{P}[S \geq 1] \leq \mathbb{E}[S]$. We will prove that, under the conditions of Theorem 3,

$$\mathbb{P}[S \geq 1] \leq \mathbb{E}[S] = \sum_{\pi \in \Pi} \mathbb{P}(\Delta_\pi - \Delta_{\pi_0} \leq 0) \to 0. \tag{2.4}$$

---

[4] As we assume that $t = 1$, the set of possible matchings is $\{\Pi_k^l : k + l = n\}$. This is equivalent to the set of all possible permutations over $\{1, 2, \cdots, n\}$, i.e., the symmetric group S_n.

The main complication of the proof stems from the fact that the random variables $\varDelta_\pi$ and $\varDelta_{\pi_0}$ are correlated in a complex way, because they are both functions of the random vertex and random edge sets $V_{1,2}$ and $E_{1,2}$. Both $\varDelta_\pi$ and $\varDelta_{\pi_0}$ can be written as sums of Bernoulli random variables. The main challenge in the proof is to decompose the difference $\varDelta_\pi - \varDelta_{\pi_0}$ into components that are mutually independent and can be appropriately bounded.

For this, we first partition the node sets V_1 and V_2 with respect to how they are mapped by π and π_0. This node partition induces an edge partition. Elements of some parts of the edge partition contribute equally to $\varDelta_\pi$ and $\varDelta_{\pi_0}$ and can be ignored. The remaining parts can be further subdivided into linear structures (specifically, chains and cycles) with only internal and short-range correlation. Finally, this leads to the desired decomposition of the sums of Bernoullis to apply standard concentration arguments to $\varDelta_\pi$ and $\varDelta_{\pi_0}$ individually, and to then stochastically bound their difference.

Detailed Proof of Theorem 3

We consider the contribution of edges (or potential edges) to the terms $\varDelta_\pi$ and $\varDelta_{\pi_0}$ as a random variable in the $G(n,p;t,s)$ probability space. More precisely, for a pair of nodes $u, v \in V_1$ and their images under the matching π (i.e., $\pi(u), \pi(v)$) we look at the probability of having/not having an edge between these nodes in $G_{1,2}$. From now on, a *pair e* represents a possible edge $e = (u, v)$ which, based on the realization of the $G(n,p;t,s)$ bigraph random model, might have or not have an actual edge between the nodes u and v.

Let us call the set of all pairs in G_1 as V_1^2 (here, we slightly abuse the notation, meaning $\binom{V_1}{2}$). The set V_2^2 is defined similarly. We define, by analogy, the set of matched pairs $V_1^2(\pi)$ as the set of all the pairs $(u,v) \in \binom{V_1(\pi)}{2}$. Also, the set $V_2^2(\pi)$ is defined similarly.

The term $\varPhi_\pi$ counts the number of edges, that in both graphs, are matched to a nonexistent edge in the other graph. More precisely, the contribution of pair $e \in V_1^2(\pi)$ and its image $\pi(e) \in V_2^2(\pi)$ to $\varPhi_\pi$ is $\phi(e) = |\mathbb{1}_{\{e \in E_1(\pi)\}} - \mathbb{1}_{\{\pi(e) \in E_2(\pi)\}}|$. Note that pairs e and $\pi(e)$ contribute to $\varPhi_\pi$ if and only if exactly one of them exists in G_1 or G_2. Also, for $e \in V_1^2 \setminus V_1^2(\pi)$, we define $\psi_1(e) = \mathbb{1}_{\{e \in E_1 \setminus E_1(\pi)\}}$ which represents the contribution of pair e to $\varPsi_\pi^1$. This indicator term is equal to 1 if the edge between unmatched pair e in G_1 exists. Similarly, for $e \in V_2^2 \setminus V_2^2(\pi)$, we define $\psi_2(e) = \mathbb{1}_{\{e \in E_2 \setminus E_2(\pi)\}}$. To sum up, we can write $\varDelta_\pi$ as

$$\varDelta_\pi = \sum_{e \in V_1^2(\pi)} \phi(e) + \alpha \left[\sum_{e \in V_1^2 \setminus V_1^2(\pi)} \psi_1(e) + \sum_{e \in V_2^2 \setminus V_2^2(\pi)} \psi_2(e) \right].$$

In order to compute contributions of different pairs to $\varDelta_\pi$ and $\varDelta_{\pi_0}$, we first partition the set of vertices $V_1 \cup V_2$ based on the matchings π and π_0. Then we partition the node pairs with respect to this node partition.

2.2.1 Node Partition

We partition the nodes in $V_1 \cup V_2$ into the following five parts based on the matching π:

1. $\checkmark(\pi)$ is the set of nodes that are matched correctly by π, i.e., $\checkmark(\pi) = \{u \in V_1 \cup V_2 | \pi(u) = u\}$.
2. $\rightarrow(\pi)$ is the set of nodes that are matched in the graph G_1, but π^{-1} is null for them, i.e., $\rightarrow(\pi) = \{u \in V_1 \cup V_2 | \pi(u) \neq \emptyset, \pi^{-1}(u) = \emptyset\}$.
3. $\leftarrow(\pi)$ is the set of nodes that are matched in the graph G_2, and π is null for them, i.e., $\leftarrow(\pi) = \{u \in V_1 \cup V_2 | \pi(u) = \emptyset, \pi^{-1}(u) \neq \emptyset\}$.
4. $\leftrightarrow(\pi)$ is the set of nodes that are matched in both graphs $G_{1,2}$, but incorrectly, i.e., $\leftrightarrow(\pi) = \{u \in V_1 \cup V_2 | \pi(u) \neq \{u, \emptyset\}, \pi^{-1}(u) \neq \{u, \emptyset\}\}$.
5. $\times(\pi)$ is the set of nodes which are null in both graphs $G_{1,2}$ under the matching π, i.e., $\times(\pi) = \{u \in V_1 \cup V_2 | \pi(u) = \emptyset, \pi^{-1}(u) = \emptyset\}$.

In the matching π_0 all the nodes in V_0 are matched correctly and the other nodes are left unmatched; therefore, only the two sets $\checkmark(\pi_0)$ and $\times(\pi_0)$ are nonempty. The pairwise intersections of the partitions under the two matchings π and π_0 are shown in Table 2.1. For an example of these pairwise intersections, see Table 2.2.

Table 2.1 – Partition of the nodes in $V_1 \cup V_2$ into eight sets based on the pairwise intersections of partition of the nodes in $V_1 \cup V_2$ under π and π_0.

π π_0	$\checkmark$	$\leftrightarrow$	$\rightarrow$	$\leftarrow$	$\times$
$\checkmark$	$\mathcal{C}$	$\mathcal{W}$	$\mathcal{L}$	$\mathcal{R}$	$\mathcal{S}$
$\times$	$\emptyset$	$\emptyset$	$\mathcal{Q}$	$\mathcal{X}$	$\mathcal{U}$

Table 2.2 – Example of partition of the nodes $V_1 \cup V_2$ of the graphs $G_{1,2}$ from Figure 2.1.

π π_0	$\checkmark$	$\leftrightarrow$	$\rightarrow$	$\leftarrow$	$\times$
$\checkmark$	u_1, u_2	u_3, u_4, u_5, u_6	$\emptyset$	u_7	$\emptyset$
$\times$	$\emptyset$	$\emptyset$	u_8, u_9	u_{12}	u_{10}, u_{11}

2.2.2 Edge Partition

We now partition the set of pairs with respect to the classes of nodes defined in Table 2.1. A pair e contributes equally to Δ_π and Δ_{π_0} (i) if it is matched in the same way by π and π_0

(i.e., $\pi_0(e) = \pi(e)$), or (ii) if it is null in both alignments. The following sets are the pairs that contribute equally to Δ_π and Δ_{π_0}, consequently, their contributions will cancel-out in the difference $\Delta_\pi - \Delta_{\pi_0}$:

1. Pairs between the nodes in the set $\mathcal{C}$. These pairs are present in both graphs and their endpoints are matched correctly by both π and π_0. For example, in Figure 2.1, the pair (u_1, u_2) is matched with the same pair by alignments π_0 and π.
2. Pairs, in G_1 between $\mathcal{U} \cap V_1$ (i.e., the nodes in V_1 which are unmatched by π and not sampled in V_2) and V_1, contribute equally to both Ψ_π and Ψ_{π_0}. Similarly, for the pairs in $(\mathcal{U} \cap V_2) \times V_2$ in the graph G_2. Note that these pairs are present in only one of the graphs. For example, in Figure 2.1, the pairs (u_{10}, u_{11}), (u_{10}, u_{12}) and (u_{10}, u_2) in graph G_2 are not matched either under π or under π_0.
3. Pairs e between $\mathcal{Q}$ and $\mathcal{S} \cup \mathcal{R}$ in the graph G_1 contribute equally to both Ψ_π and Ψ_{π_0} by a term $\psi_1(e)$. Similarly, the pairs between $\mathcal{X}$ and $\mathcal{S} \cup \mathcal{L}$ in the graph G_2 contribute with a term $\psi_2(e)$ under both alignments π and π_0. Note that these pairs are present only in one of the graphs. In Figure 2.1, (u_7, u_8) and (u_7, u_9) provide two examples of pairs in this class from graph G_1.

Let Z_π and Z_{π_0} denote the contribution of all the pairs from these partitions (mentioned above) to Δ_π and Δ_{π_0}, respectively. We know that $Z_\pi = Z_{\pi_0}$. Let's define $\mathcal{E}$ as the set of all the remaining pairs that are matched differently under π and π_0. Note that $\mathcal{E}$ depends on both alignments π and π_0. As for each instance of the $G(n, p; t, s)$ bigraph model, the matching π_0 is fixed; for simplicity of notation, we drop the dependence on π_0. Furthermore, we define $X_\pi = \Delta_\pi - Z_\pi$ and $Y_\pi = \Delta_{\pi_0} - Z_{\pi_0}$. Here, X_π and Y_π represent the sums of indicator terms over the contribution of pairs in the set $\mathcal{E}$ under alignments π and π_0, respectively. In summary, we have

$$\Delta_\pi - \Delta_{\pi_0} = (X_\pi + Z_\pi) - (Y_\pi + Z_{\pi_0}) = X_\pi - Y_\pi. \tag{2.5}$$

The next step of the proof is to find a lower-bound for $X_\pi - Y_\pi$. In order to compute contributions of pairs from the set $\mathcal{E}$ to different indicator terms in X_π and Y_π, we partition this set into the following sub-classes:

1. The set of pairs present in only one of the graphs $G_{1,2}$ and matched by π. Note that at least one of the endpoints of these pairs are not sampled in either $V_{1,2}$. Therefore, these pairs are not matched by π_0. These pairs are divided into the two following sets:
    - $\mathcal{E}_{\emptyset,M*} = \{(i,j) \in (\mathcal{Q} \times V_1(\pi))\}$ is the set of pairs that contribute with $\psi_1(e)$ to $\Psi^1_{\pi_0}$ and with $\phi(e)$ to Φ_π.
    - $\mathcal{E}_{\emptyset,*M} = \{(i,j) \in (\mathcal{X} \times V_2(\pi))\}$ is the set of pairs that contribute with $\psi_2(e)$ to $\Psi^2_{\pi_0}$ and with $\phi\left(\pi^{-1}(e)\right)$ to Φ_π.

For example, in Figure 2.1, we have $(u_3, u_8) \in \mathcal{E}_{\emptyset,M*}$ and $(u_1, u_{12}) \in \mathcal{E}_{\emptyset,*M}$.

2. The set of pairs is present in both graphs $G_{1,2}$, but is unmatched by π in at least one of the graphs. These pairs can be further partitioned into three sub-classes:

    - $\mathcal{E}_{M,M\emptyset} = \big\{(i,j) \in \mathcal{L} \times (\mathcal{C} \cup \mathcal{W} \cup \mathcal{L})\big\}$ is the set of pairs that are matched in G_1 and unmatched in G_2. A pair $e \in \mathcal{E}_{M,M\emptyset}$ contributes with $\phi(e)$ to Φ_{π_0} and Φ_π, and with $\psi_2(e)$ to Ψ^2_π.
    - $\mathcal{E}_{M,\emptyset M} = \{(i,j) \in \mathcal{R} \times (\mathcal{C} \cup \mathcal{W} \cup \mathcal{R})\}$ is the set of pairs that are matched in G_2 and unmatched in G_1.
    - $\mathcal{E}_{M,\emptyset\emptyset} = \{(i,j) \in (\mathcal{S} \times V_0) \bigcup (\mathcal{L} \times \mathcal{R})\}$ is the set of pairs that are unmatched by π in both graphs. These pairs contribute with $\phi(e)$ to Φ_{π_0}, and with $\psi_2(e)$ to both Ψ^1_π and Ψ^2_π.

    In Figure 2.1, the unmatched pair (u_4, u_7) in G_1 is matched by π only in G_2, i.e., $(u_4, u_7) \in \mathcal{E}_{M,\emptyset M}$.

3. $\mathcal{E}_{M,MM} = \{(i,j) \in \mathcal{W} \times (\mathcal{C} \cup \mathcal{W})\}$ is the set of pairs that are present and matched incorrectly by π in both graphs $G_{1,2}$. These pairs are matched differently by π and π_0. The pairs in the set $\mathcal{E}_{M,MM}$ contribute with terms $\phi(e)$ to Φ_{π_0}, and contribute with terms $\phi(e)$ and $\phi\big(\pi^{-1}(e)\big)$ to Φ_π. Note that this is not generally true. Indeed, transpositions[5] in π contribute equally to both Φ_π and Φ_{π_0}. We have at most $\lfloor k/2 \rfloor$ pairs of this type, because the number of incorrectly matched couples is k. To be more concrete, we do not consider these pairs in the set $\mathcal{E}_{M,MM}$. For example, in Figure 2.1, the pairs (u_1, u_3) and (u_4, u_5) that are matched differently by π_0 and π belong to the set $\mathcal{E}_{M,MM}$.

Now, let us define the sizes of the described sets as follows: $m_1 = |\mathcal{E}_{\emptyset,M*} \cup \mathcal{E}_{\emptyset,*M}|$, $m_{2,1} = |\mathcal{E}_{M,M\emptyset} \cup \mathcal{E}_{M,\emptyset M}|$, $m_{2,2} = |\mathcal{E}_{M,\emptyset\emptyset}|$, $m_2 = m_{2,1} + m_{2,2}$ and $m_3 = |\mathcal{E}_{M,MM}|$. Also, we define $m = m_1 + m_2 + m_3$.

2.2.3 Indicator Terms and Expected Values

In Lemma 5, the two terms X_π and Y_π are expressed as sums of indicator terms (which are correlated Bernoulli random variables) over the pairs in $\mathcal{E}$.

Lemma 5. *For X_π we have:*

$$X_\pi = \sum_{e \in \mathcal{E}_{\emptyset,M*} \cup \mathcal{E}_{M,M\emptyset} \cup \mathcal{E}_{M,MM}} \phi(e) + \alpha \left[\sum_{e \in \mathcal{E}_{M,\emptyset M} \cup \mathcal{E}_{M,\emptyset\emptyset}} \psi_1(e) + \sum_{e \in \mathcal{E}_{M,M\emptyset} \cup \mathcal{E}_{M,\emptyset\emptyset}} \psi_2(e) \right], \tag{2.6}$$

---

[5] A pair (u, v) is a transposition under π if $\pi(u) = v$ and $\pi(v) = u$.

where $\phi(e) \sim Be\big(2ps(1-ps)\big)$ *and* $\psi_1(e), \psi_2(e) \sim Be(ps)$. *For* Y_π *we have:*

$$Y_\pi = \sum_{e \in \mathcal{E}_{M,M\emptyset} \cup \mathcal{E}_{M,\emptyset M} \cup \mathcal{E}_{M,\emptyset\emptyset} \cup \mathcal{E}_{M,MM}} \phi(e) + \alpha \left[\sum_{e \in \mathcal{E}_{\emptyset,M*}} \psi_1(e) + \sum_{e \in \mathcal{E}_{\emptyset,*M}} \psi_2(e) \right], \tag{2.7}$$

where $\phi(e) \sim Be\big(2ps(1-s)\big)$, *and* $\psi_1(e), \psi_2(e) \sim Be(ps)$.

Proof. First, note that $\mathcal{E}_{\emptyset,M*} \cup \mathcal{E}_{M,M\emptyset} \cup \mathcal{E}_{M,MM} = \mathcal{E} \cap V_1^2(\pi)$ is the set of all matched pairs from G_1 in the set $\mathcal{E}$. Remember that by (2.5) the term X_π is the sum of indicators in Δ_π over pairs in the set $\mathcal{E}$. Thus, we get the first term in the right-hand side of (2.6). Each pair e (the same is true for $\pi(e)$) exists in each of the graphs $G_{1,2}$ with probability ps; thus $\phi(e) = Be\big(2ps(1-ps)\big)$. Second, we compute the number of terms $\psi_{1,2}(e)$ that contribute to X_π. These terms are (i) pairs of type $\mathcal{E}_{M,M\emptyset} \cup \mathcal{E}_{M,\emptyset M}$ that contribute to either Ψ_π^1 or Ψ_π^2, and (ii) pairs of type $\mathcal{E}_{M,\emptyset\emptyset}$ that contribute to both Ψ_π^1 and Ψ_π^2. The probability of a pair e to have an actual edge $e \in E_{1,2}$ is ps, hence $\psi_1(e), \psi_2(e) \sim Be(ps)$.

Y_π is the contribution of the pairs in the set $\mathcal{E}$ to Δ_{π_0}. For each pair e matched by π_0 and π, $e \in \mathcal{E}_{M,M\emptyset} \cup \mathcal{E}_{M,\emptyset M} \cup \mathcal{E}_{M,\emptyset\emptyset} \cup \mathcal{E}_{M,MM}$ there is an indicator $\phi(e)$ in Y_π. Note that this $\phi(e)$ is an indicator of the event that e is sampled in G_1 and $\pi(e) = e$ is not sampled in G_2 (or vice versa). Thus $\phi(e) = Be\big(2ps(1-s)\big)$. The argument for $\psi_1(e), \psi_2(e)$ is the same as for X_π. This proves (2.7). □

In the next corollary, we compute the expected values of X_π and Y_π.

Corollary 6. *For* X_π *and* Y_π *we have*

$$\mathbb{E}[X_\pi] = \Big(m_3 + \frac{m_1 + m_{2,1}}{2}\Big) 2ps(1-ps) + \alpha m_{2,1} ps + 2\alpha m_{2,2} ps.$$

$$\mathbb{E}[Y_\pi] = (m_2 + m_3) 2ps(1-s) + \alpha m_1 ps.$$

Proof. Note that the term $\phi(e)$, which is defined as $\phi(e) = |\mathbb{1}_{\{e \in E_1(\pi)\}} - \mathbb{1}_{\{\pi(e) \in E_2(\pi)\}}|$, depends on pairs e and $\pi(e)$ from graphs G_1 and G_2, respectively. Also, as the matching π is an injective function, each pair $e \in V_1^2$ can be matched to at most one pair from V_2^2. This is generally true for pairs $e \in V_2^2$ from G_2. Therefore, the number of pairs from graph G_1, which contribute to terms $\{\phi(e)\}$, is equal to the number of pairs from graph G_2 in these terms, i.e., $|\mathcal{E}_{\emptyset,M*} \cup \mathcal{E}_{M,M\emptyset} \cup \mathcal{E}_{M,MM}| = |\mathcal{E}_{\emptyset,*M} \cup \mathcal{E}_{M,\emptyset M} \cup \mathcal{E}_{M,MM}|$. Remember that $|\mathcal{E}_{\emptyset,M*} \cup \mathcal{E}_{\emptyset,*M}| = m_1$ and $|\mathcal{E}_{M,M\emptyset} \cup \mathcal{E}_{M,\emptyset M}| = m_2$. In short, the number of $\{\phi(e)\}$ terms that contribute to X_π (defined precisely in Lemma 5) is $m_3 + \frac{m_1 + m_{2,1}}{2}$. The rest of the proof comes directly from the definitions of m_1, m_2 and m_3. □

In the following lemma, we prove that the expected value fof X_π is larger than the expected value of Y_π.

Lemma 7. *If* $1 - ps > \alpha > 1 - s$, *then* $\mathbb{E}[X_\pi] > \mathbb{E}[Y_\pi]$.

Proof. From Corollary 6, we have

$$\mathbb{E}[X_\pi] > ps\big((1-ps)m_1 + 2\alpha m_2 + 2(1-ps)m_3\big) > \mathbb{E}[Y_\pi],$$

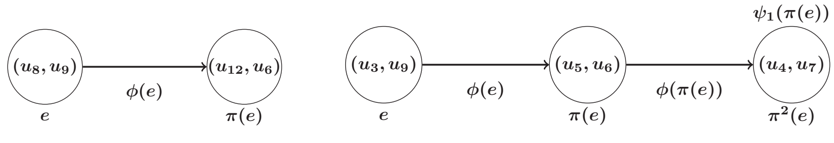
if the following inequalities hold: (i) $(1-ps) > \alpha$, (ii) $\alpha > (1-s)$, and (ii) $(1-ps) > (1-s)$. Note that if the first two inequalities hold, then the third inequality is true also. □

2.2.4 Correlation Structure

Lemma 7 guarantees that for any $\pi \neq \pi_0$, $\mathbb{E}[\Delta_\pi] > \mathbb{E}[\Delta_{\pi_0}]$. Next, we demonstrate that X_π and Y_π, which are sums of correlated Bernoulli random variables, concentrate around their means.

Due to the edge sampling process, the presence of edges between the nodes in V_0 is correlated in the two graphs G_1 and G_2. For example, consider an event $\phi(e)$ that is a function of edges $e \in G_1$ and $\pi(e) \in G_2$. Furthermore, assume $\pi(e)$ is sampled and matched in the graph G_1. Then, the presence of $\pi(e)$ in G_1 is correlated with the presence of $\pi(e)$ in G_2. Therefore, the two terms $\phi(e)$ and $\phi(\pi(e))$ are correlated. By the same lines of reasoning, if $\pi^2(e)$ is sampled and matched in G_1, the two terms $\phi(\pi(e))$ and $\phi\left(\pi^2(e)\right)$ are correlated, and so on.

To address the correlations contributing to terms Φ_π and Ψ_π, we first define, for an alignment π, the two concepts of *chains* and *cycles*. We call a sequence of non-repeating pairs $(e_1, \cdots, e_i \cdots, e_q)$ a *chain* if (i) $\pi^{-1}(e_1) = \emptyset$, i.e., e_1 is either unmatched or not sampled in G_2; (ii) $\pi(e_q) = \emptyset$, i.e., e_q is either unmatched or not sampled in G_1; and (iii) $\pi(e_i) = e_{i+1}$ for $1 \le i < q$, i.e., each pair in a chain is the image of the previous pair in that chain.[6] In Figure 2.3b, the sequence $((u_3,u_9),(u_5,u_6),(u_4,u_7))$ is an example of a chain of length three. Also, we call a sequence of differing pairs $(e_1, \cdots, e_i, \cdots, e_q)$ a *cycle* if (i) $\pi(e_i) = e_{i+1}$ for $1 \le i < q$; and (ii) $\pi(e_q) = e_1$. As an example, see the cycle $((u_2,u_3),(u_2,u_5),(u_2,u_4))$ in Figure 2.4a.



(a) (b)

Figure 2.3 – (a) Example of a chain with length one from the matching π from Figure 2.1. (b) Example of a chain with length three from the matching π from Figure 2.1: The term $\psi_1(\pi(e))$ corresponds to the contribution of pair (u_2,u_6) in the graph G_1. In this chain, the term $\phi(\pi(e))$ is correlated with the two terms $\phi(e)$ and $\psi_1(\pi(e))$.

---

[6] Note that a chain or cycle of pairs is defined for a given alignment π.

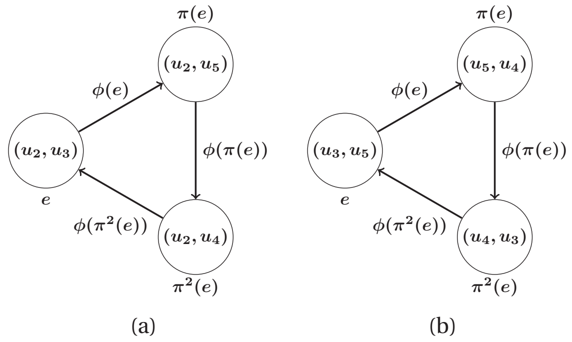


Figure 2.4 – Examples of two cycles from the matching π from Figure 2.1: Pairs generate a cycle of dependent terms. In these cycles, the terms $\phi(e), \phi(\pi(e))$ and $\phi\left(\pi^2(e)\right)$ are correlated pairwise.

Following the discussion above, we state Lemmas 8 and 9. In Lemma 8, we (i) partition all the pairs of $\mathcal{E}$ into chains and cycles and (ii) demonstrate contributions of these pairs to different indicator terms. In Lemma 9, we characterize correlations between terms in the induced sequence of indicators.

Lemma 8. *All the pairs in the set $\mathcal{E}$ can be partitioned into chains and cycles, where they induce sequences of indicator terms as follows:*

- *For each cycle $(e_1, \cdots, e_i, \cdots e_q), 1 \leq i < q$, the e_i pairs contribute to the induced sequence of indicator terms $\left(\phi(e_1), \cdots \phi(e_i), \cdots \phi(e_q)\right)$.*
- *For each chain $(e_1, \cdots e_i, \cdots e_q), 1 \leq i < q$, the e_i pairs contribute to one of the following five types of induced sequences of indicator terms:*
    1. *$e_1 \in \mathcal{E}_{\emptyset,M*}$ and $e_q \in \mathcal{E}_{\emptyset,*M}$: these pairs contribute to the induced sequence of indicator terms $\left(\phi(e_1), \cdots \phi(e_i), \cdots \phi(e_{q-1})\right)$.*
    2. *$e_1 \in \mathcal{E}_{\emptyset,M*}$ and $e_q \in \mathcal{E}_{M,\emptyset M}$: these pairs contribute to the induced sequence of indicator terms $\left(\phi(e_1), \cdots \phi(e_i), \cdots \phi(e_{q-1}), \psi_1(e_q)\right)$.*
    3. *$e_1 \in \mathcal{E}_{M,M\emptyset}$ and $e_q \in \mathcal{E}_{\emptyset,*M}$: these pairs contribute to the induced sequence of indicator terms $\left(\psi_2(e_1), \phi(e_1), \cdots \phi(e_i), \cdots \phi(e_{q-1})\right)$.*
    4. *$e_1 \in \mathcal{E}_{M,M\emptyset}$ and $e_q \in \mathcal{E}_{M,\emptyset M}$: these pairs contribute to the induced sequence of indicator terms $\left(\psi_2(e_1), \phi(e_1), \cdots \phi(e_i), \cdots \phi(e_{q-1}), \psi_1(e_q)\right)$.*
    5. *$e_1 \in \mathcal{E}_{M,\emptyset\emptyset}$: we have a chain of length one. The edge e_1 contributes to the induced sequence of indicator terms $\left(\psi_2(e_1), \psi_1(e_1)\right)$.*

Lemma 9. *For sequences of induced indicator terms from partitions in Lemma 8, we have*

- *All the induced indicators from $\{\phi \cup \psi\}$, associated with different chains and cycles, are mutually independent.*

- *For a chain, each indicator from* $\{\phi \cup \psi\}$ *is correlated with at most the preceding and subsequent indicators in the induced sequence.*

- *For a cycle, each indicator from* $\{\phi\}$ *is correlated with at most the preceding and subsequent indicators in the induced sequence, and* $\phi(e_1)$ *is correlated with* $\phi(e_q)$.

For details regarding the correctness of this partition, their induced indicator terms and the correlation arguments refer to Appendix 2.B.

From Lemma 9, we know that each term from $\{\phi(e) \cup \psi_{1,2}(e)\}$ is correlated with at most two of its neighbors (e.g., see Figure 2.3 and 2.4). We associate a label 0 or 1 with all the induced $\phi(e)$ and $\psi_{1,2}(e)$ terms by alternating these labels. We obtain a labeling that all the indicators with the same label are independent. Note that this is not generally true for the terms that are at the start and end of cycles with odd number of indicator terms. For more explanation on how to handle these special cases, see the discussions and the detailed computation of the concentration bounds in Appendix 2.C. Based on this labeling strategy, we can split the terms which contribute to X_π into two sums of independent random variables and derive concentration bounds for each sum. Next, by using these bounds, we find an upper-bound for $\mathbb{P}[X_\pi - Y_\pi \le 0]$.

2.2.5 Concentration

We define $\mu_1 = \mathbb{E}[X_\pi]$ and $\mu_2 = \mathbb{E}[Y_\pi]$ and apply a union bound for the difference $X_\pi - Y_\pi$ (2.5) as follows

$$\mathbb{P}[X_\pi - Y_\pi \le 0] \le \mathbb{P}\left[X_\pi < \frac{\mu_1+\mu_2}{2}, Y_\pi > \frac{\mu_1+\mu_2}{2}\right] \le \mathbb{P}\left[X_\pi < \frac{\mu_1+\mu_2}{2}\right] + \mathbb{P}\left[Y_\pi > \frac{\mu_1+\mu_2}{2}\right]. \quad (2.8)$$

From the result of Lemma 13 from Appendix 2.C, we use the following bounds for the concentration of X_π and Y_π around their means:

$$\begin{aligned} \mathbb{P}\left[X_\pi < \frac{\mu_1+\mu_2}{2}\right] &\le 2\exp\left(-\frac{(\mu_1-\mu_2)^2}{96\mu_1}\right), \\ \mathbb{P}\left[Y_\pi > \frac{\mu_1+\mu_2}{2}\right] &\le \exp\left(-\frac{(\mu_1-\mu_2)^2}{12\mu_1}\right). \end{aligned} \quad (2.9)$$

The first step to upper-bound (2.8) is to find a lower-bound for $\frac{\mu_1-\mu_2}{\mu_1}$ (2.9). Let's define

$$\alpha' = \min\big((1-ps-\alpha), (\alpha-(1-s))\big).$$

From Corollary 6, we have

$$\mu_1 - \mu_2 \ge \alpha' ps(m_1+m_2+m_3) \ge ps\alpha' m.$$

Also, note that $\mu_1 \leq 2mps$ and $\mu_2 \leq 2mps$. Therefore, we have

$$\frac{(\mu_1-\mu_2)^2}{\mu_1} \geq \frac{\alpha'^2 mps}{2}.$$

To sum up, we have

$$\mathbb{P}[X_\pi - Y_\pi \leq 0] \leq \mathbb{P}\left[X_\pi < \frac{\mu_1+\mu_2}{2}\right] + \mathbb{P}\left[Y_\pi > \frac{\mu_1+\mu_2}{2}\right] \leq 3\exp\left(-\alpha'^2 \frac{mps}{192}\right). \tag{2.10}$$

Thus the expected number of alignments $\pi \neq \pi_0$ such that $\Delta_\pi \leq \Delta_{\pi_0}$ is

$$E(S) \leq \sum_{k,l} |\Pi_k^l| \mathbb{P}[X_\pi - Y_\pi \leq 0] \leq \sum_{k,l} |\Pi_k^l| 3\exp\left(-\frac{\alpha'^2}{192} mps\right).$$

To finalize our proof, it remains to find a lower bound for m (which is the number of node pairs in the set $\mathcal{E}$) and an upper bound for $|\Pi_k^l|$.

Lemma 10. *We have*

1. *if $k \leq n_0 - l$, then $m > \frac{(n_0-l)(n_0-2)}{2}$ and $|\Pi_k^l| < n^{3(n_0-l)}$.*
2. *if $k > n_0 - l$, then $m > \frac{k(n_0-2)}{2}$ and $|\Pi_k^l| < n^{3k}$.*

Proof. First, we upper-bound the number of alignments in the set Π_k^l. For this reason. assume we first choose l nodes from n_0 nodes in the set V_0 that are matched correctly. Then, we choose k other nodes from the remaining nodes of V_1 and V_2. Also, there are at most $k!$ possible alignments between these k chosen nodes. Therefore,

$$|\Pi_k^l| \leq \binom{n_0}{l}\binom{n_1-l}{k}\binom{n_2-l}{k} k! \leq n_0^{n_0-l} n_1^k n_2^k. \tag{2.11}$$

Based on the value of k we consider two different cases:

- if $k \leq n_0 - l$, then $|\Pi_k^l| < n^{3(n_0-l)}$. By definition, $m = |\mathcal{E}|$ is the number of pairs that are matched differently by π and π_0. This includes the set of pairs between any sampled node $v_1 \in V_0$ and any node $v_2 \in V_0$ matched differently by π and π_0. Note that these pairs are all the present pairs and there are $m_2 + m_3$ of them. Also, we should consider the pairs that contribute equally to both terms due to transpositions. Thus we have

  $$m \geq \binom{n_0-l}{2} + (n_0-l)l - \lfloor \frac{k}{2} \rfloor \geq \frac{(n_0-l)(n_0-2)}{2}.$$

- if $k > n_0 - l$, then $|\Pi_k^l| < n^{3k}$. Here note that the set $\mathcal{E}$ includes all the pairs between any sampled node $v_1 \in V_0$ and any node $v_2 \in V_1(\pi) \cup V_2(\pi)$ that are matched differently by π and π_0. Again, we should consider transpositions. We compute the number of pairs matched by π as $m \geq m_3 + m_1 \geq \binom{k}{2} + kl - \lfloor \frac{k}{2} \rfloor$. After that, if $k \geq n_0$, we have the statement immediately; otherwise, we use $l > n_0 - k$, and obtain

$$m \geq \binom{k}{2} + k(n_0 - k) - \lfloor \frac{k}{2} \rfloor \geq \frac{k(n_0 - 2)}{2}.$$

So, we can lower-bound m for two different cases and find an upper-bound for $|\Pi_k^l|$. □

Now, we find an upper bound for $\mathbb{E}[S]$ from the result of Lemma 10. (1) If $k \leq n_0 - l$: we define $i = n_0 - l$. Using the facts that $m > \frac{(n_0 - l)(n_0 - 2)}{2}$, $k \leq n$ and $|\Pi_k^l| < n^{3(n_0 - l)}$, we obtain

$$\mathbb{E}[S] \leq \sum_{k,l} 3 \exp\left(i \left(3 \log n - ps \frac{\alpha'^2}{384} (n_0 - 2) \right) \right) \leq \sum_{i=1}^{n_0} 3 \exp\left((3i + 1) \log n - i ps \frac{\alpha'^2}{384} (n_0 - 2) \right).$$

(2) If $k > n_0 - l$: using the facts that $m > \frac{k(n_0 - 2)}{2}$ and $|\Pi_k^l| < n^{3k}$, we obtain

$$\mathbb{E}[S] \leq \sum_{k,l} 3 \exp\left(k \left(3 \log n - ps \frac{\alpha'^2}{384} (n_0 - 2) \right) \right) \leq \sum_{k=1}^{n} 3 \exp\left((3k + 1) \log n - kps \frac{\alpha'^2}{384} (n_0 - 2) \right). \tag{2.12}$$

The geometric sum of (2.12) goes to 0, if its first term goes to 0. Thus if we assume $ps \frac{\alpha'^2}{384} n_0 - 4 \log n = \omega(1)$, we obtain $\mathbb{E}[S] \to 0$. We can show that $n_0 = nt^2 (1 + o(1))$ from a Chernoff bound and conclude $ps \frac{\alpha'^2 t^2}{384} n - 4 \log n = \omega(1)$.[7]

To conclude the proof of Theorem 3, we choose $\alpha = \frac{(1-ps)+(1-s)}{2} = 1 - \frac{s(1+p)}{2}$; then $\alpha' = \frac{s(1+p)}{2}$. In summary, we derive the final bound $ps \frac{s^2 t^2}{1536} n - 4 \log n = \omega(1)$ or $ps^3 t^2 = \frac{6144 \log n + \omega(1)}{n}$.

2.3 Summary

In this chapter, we have addressed the problem of matching two unlabeled graphs by their edge structure alone. We have proposed a stochastic model for generating two correlated graphs with partial node and edge overlaps. More precisely, we introduce the $G(n, p; t, s)$ bigraph generator model, where $G(n, p)$ is the underlying ground-truth graph, and t and s are two parameters that control the similarities of the vertex and edge sets, respectively. We take an information-theoretic perspective, in that we ignore computational limitations and identify sufficient conditions such that a combinatorial optimization problem yields the correct answer with high probability.

We have given conditions on the graph density p, and have proved that within these conditions

---

[7] For any $\alpha \in [1 - s, 1 - ps]$.

the true partial matching between the node sets of the two graphs can be inferred with zero error. The conditions on the node and edge similarity parameters t and s are quite benign: essentially, the average node degree has to grow as $\frac{\log(n)+\omega(1)}{s^2 t}$.

Beyond establishing the scaling relation of network alignment in the presence of partial node overlap, the configuration of the cost function suggests heuristics for efficient algorithms. In particular, the cost function takes the form of a graph edit distance, but with a trade-off between the two types of error (mismatch and map-to-null) quite delicate to control (through the parameter α). We therefore expect our model and result to be useful in the development and tuning of matching heuristics in practice and to shed light on the connection between exact and approximate graph isomorphism.

Appendix

2.A Graph Matching and Edge Mismatch: Proof of Lemma 4

In this appendix, we provide the proof of Lemma 4. Assume (i) a graph $G(V,E)$ is sampled from $G(n,p)$; (ii) the edges of graphs $G_1(V_1,E_1)$ and $G_2(V_2,E_2)$ are sampled from the edges of G by two different independent edge sampling processes with probability s; and (iii) a permutation π_0 is chosen uniformly at random from the symmetric group S_n and it is applied to G_2. The goal is then to identify the permutation π_0 based on the two given graphs G_1 and G_2. For convenience of notation, we assume with out loss of generality that the unknown π_0 is the identity permutation. Note that we can interpret graphs $G_{1,2}$ as samples of the $G(n,p;t,s)$ model with $t=1$.

As the choice of permutation π is uniformly at random from the symmetric group S_n, the MAP rule is equivalent to the *maximum likelihood (ML)* rule. More precisely, we can write

$$\mathbb{P}[\pi|G_1,G_2]=\frac{\mathbb{P}[G_1,G_2|\pi]\,\mathbb{P}[\pi]}{\mathbb{P}[G_1,G_2]}.$$

By noting that $\mathbb{P}[\pi]=\frac{1}{n!}$ and $\mathbb{P}[G_1,G_2]$ is fixed for any permutation π, we have

$$\hat{\pi}_{\mathrm{MAP}}=\hat{\pi}_{\mathrm{ML}}=\underset{\pi\in S_n}{\operatorname{argmax}}\,\mathbb{P}[G_1,G_2|\pi]. \tag{2.13}$$

To compute the right-hand side of (2.13), let's define the following:

$$\begin{aligned}\nabla_\pi&=|\pi(E_1)\cap E_2|=\frac{|E_1|+|E_2|-\Phi_\pi}{2},\\ U_\pi&=|\pi(E_1)\cup E_2|=\Phi_\pi+\nabla_\pi.\end{aligned}$$

We then have

$$\mathbb{P}[G_1,G_2|\pi]=\sum_G\mathbb{P}[G_1,G_2,G|\pi]=\sum_G\mathbb{P}[G_1,G_2|\pi,G]\mathbb{P}[G].$$

To compute this sum, we note on one hand that if G does not contain all the edges in $E_1\cup\pi^{-1}(E_2)$, then $\mathbb{P}[G_1,G_2|\pi,G]=0$. On the other hand, if G does contain all the edges in $E_1\cup\pi^{-1}(E_2)$ (note that the total number of such graphs G is $2^{\binom{n}{2}-U_\pi}$), then by letting i to be the number of edges in G that do not belong to the set $E_1 \cup \pi^{-1}(E_2)$, i.e., by assuming that G has $U_\pi + i$ edges, and $k = \binom{n}{2}$, we obtain

$$\begin{aligned}\mathbb{P}\left[G_1, G_2|\pi, G\right]\mathbb{P}[G] &= s^{|E_1|+|E_2|}(1-s)^{\Phi_{\pi,1}+\Phi_{\pi,2}+2i} \times p^{U_\pi+i}(1-p)^{k-U_\pi-i}\\ &= s^{U_\pi+\nabla_\pi}(1-s)^{\Phi_\pi+2i}p^{U_\pi+i}(1-p)^{k-U_\pi-i},\end{aligned}$$

where $\Phi_{\pi,1} = \sum_{e\in E_1(\pi)} \mathbb{1}_{\{\pi(e)\notin E_2\}}, \Phi_{\pi,2} = \sum_{e\in E_2(\pi)} \mathbb{1}_{\{\pi^{-1}(e)\notin E_1\}}$ and $\Phi_{\pi,1} + \Phi_{\pi,2} = \Phi_\pi$. As a result, we have

$$\begin{aligned}\mathbb{P}[G_1, G_2|\pi] = &\sum_{i=0}^{k-U_\pi}\binom{k-U_\pi}{i}s^{U_\pi+\nabla_\pi}(1-s)^{\Phi_\pi+2i}p^{U_\pi+i}(1-p)^{k-U_\pi-i}\\ =&s^{U_\pi+\nabla_\pi}(1-s)^{\Phi_\pi}p^{U_\pi}\big(p(1-s)^2+(1-p)\big)^{k-U_\pi}.\end{aligned}$$

We note further that

(i) $U_\pi = \frac{|E_1|+|E_2|+\Phi_\pi}{2}$.

(ii) the two values $U_\pi + \nabla_\pi = |E_1| + |E_2|$ and $k = \binom{n}{2}$ are not dependent on the choice of π.

As a consequence

$$\begin{aligned}\hat{\pi}_{\text{MAP}} &= \operatorname*{argmax}_{\pi\in S_n}\mathbb{P}[G_1, G_2|\pi] = \operatorname*{argmax}_{\pi\in S_n}\log\mathbb{P}[G_1, G_2|\pi]\\ &= \operatorname*{argmax}_{\pi\in S_n}\Big\{U_\pi \log\frac{p}{p(1-s)^2+1-p} + \Phi_\pi\log(1-s)\Big\}\\ &= \operatorname*{argmax}_{\pi\in S_n}\Big\{\frac{\Phi_\pi}{2}\log\frac{p}{p(1-s)^2+1-p} + \Phi_\pi\log(1-s)\Big\}\\ &= \operatorname*{argmax}_{\pi\in S_n}\Big\{\Phi_\pi \log\frac{p(1-s)^2}{p(1-s)^2+1-p}\Big\}.\end{aligned}$$

Finally, as $1-p \geq 0$ and $\frac{p(1-s)^2}{p(1-s)^2+1-p} \leq 1$ we always have

$$\log\frac{p(1-s)^2}{p(1-s)^2+1-p} \leq 0.$$

Therefore, we can conclude that

$$\hat{\pi}_{\text{MAP}} = \operatorname*{argmax}_{\pi}\mathbb{P}[\pi|G_1, G_2] = \operatorname*{argmin}_{\pi}\Phi_\pi. \tag{2.15}$$

This proves Lemma 4.

2.B Partition of Node Pairs into Chains and Cycles

In this appendix, we provide the detailed proof for Lemmas 8 and 9. More precisely, we prove that the set of chains and cycles correctly partition the pairs in set $\mathcal{E}$, and we characterize the dependence structure of the indicators within this partition.

Firstly, note that each pair $e \in \mathcal{E}_{\emptyset,M*}$ is present only in G_1, thus it contributes only to one $\phi(e)$ indicator term. Consider the chain $(e, \pi(e), \ldots \pi^c(e))$, when c is the smallest number such that $\pi^{c+1}(e)$ is null. This case happens in one of the two following scenarios:

- if $\pi^c(e) \in \mathcal{E}_{\emptyset,*M}$, then $\pi^c(e)$ is matched and exists only in G_2. Therefore, this chain of pairs induces the sequence $\left(\phi(e), \cdots, \phi\left(\pi^{c-1}(e)\right)\right)$ of indicator terms. Figure 2.3a is an example of such a chain under the matching π from Figure 2.1.
- if $\pi^c(e) \in \mathcal{E}_{M,\emptyset M}$, then $\pi^c(e)$ exists in both graphs, but is matched only in G_2. Therefore, this chain induces the sequence $\left(\phi(e), \cdots, \psi_1\left(\pi^c(e)\right)\right)$ of indicator terms. Figure 2.3b is an example of such a chain under the matching π from Figure 2.1.

Secondly, each pair $e \in \mathcal{E}_{M,M\emptyset}$ is present in both G_1 and G_2, but is matched only in G_1, thus it contributes to terms $\phi(e)$ and $\psi_2(e)$. Consider the chain $(e, \pi(e), \ldots \pi^c(e))$ when c is the smallest number such that $\pi^{c+1}(e)$ is null. This case happens in one of the two following scenarios:

- if $\pi^c(e) \in \mathcal{E}_{\emptyset,*M}$, then $\pi^c(e)$ is matched and exists only in the graph G_2. Therefore, this chain induces the sequence $\left(\psi_2(e), \phi(e), \cdots, \phi\left(\pi^{c-1}(e)\right)\right)$ of indicator terms.
- if $\pi^c(e) \in \mathcal{E}_{M,\emptyset M}$, then $\pi^c(e)$ exists in both graphs but is matched only in the graph G_2. Therefore, this chain induces the sequence $\left(\psi_2(e), \phi(e), \cdots, \psi_1\left(\pi^c(e)\right)\right)$ of indicator terms.

Now we formulate the cycle and chain partition processes as follows:

- **Chain partition:** (i) For each pair, we build a chain as described above; (ii) for each pair $e \in \mathcal{E}_{M,M\emptyset}$, we build another chain; and (iii) for each pair of type $e \in \mathcal{E}_{M,\emptyset\emptyset}$, we build another chain $(\psi_1(e), \psi_2(e))$. Note that the first two types of chains are duals of each other: For each chain of pairs that ends with a pair $e \in \mathcal{E}_{\emptyset,*M}$ or $e \in \mathcal{E}_{M,\emptyset M}$, we can build the same chain of pairs backwards; starting from e and applying π^{-1} instead of π. Based on this observation, we conclude that there are $m_1 + m_2$ pairs that start or end a chain. Thus, the fourth step is to partition the remaining, unvisited pairs that all have type $\mathcal{E}_{M,MM}$ (note that they are sampled and matched by π in both graphs).
- **Cycle partition:** For each unvisited pair e, the unvisited pair $\pi(e)$ also has type $\mathcal{E}_{M,MM}$ (otherwise $\pi(e)$ and e belong to some chain, hence e is visited), thus the pairs e and $\pi(e)$ are not null. To build a cycle, we start with a pair e and build the sequence $(e, \cdots, \pi^c(e))$, where c is the smallest number such that $\pi^c(e) = e$. We continue until there are no more unvisited pairs. Note that each indicator of a pair belongs to at most one chain or cycle because π is an injective function from V_1^2 to V_2^2. Figure 2.4 provides examples of cycles

of pairs under the matching π from Figure 2.1. Note that pairs induced by transpositions generate cycles of length two, i.e., for a pair $e = (u, v)$ with $\pi(u) = v$ and $\pi(v) = v$ the cycle $\big(\phi(e), \phi(\pi(e)\big)$ is generated where $\pi^2(e) = e$.

Remember that we defined the indicator terms as follows: (i) $\phi(e) = |\mathbb{1}_{\{e\in E_1(\pi)\}} - \mathbb{1}_{\{\pi(e)\in E_2(\pi)\}}|$; (ii) $\psi_1(e) = \mathbb{1}_{\{e\in E_1\setminus E_1(\pi)\}}$; and (iii) $\psi_2(e) = \mathbb{1}_{\{e\in E_2\setminus E_2(\pi)\}}$. From the definition, it is clear that for two node pairs $e_i \neq e_j$, we have $\psi_1(e_i) \perp\!\!\!\perp \psi_2(e_j)$. Also, if $e_j \notin \{e_i, \pi(e_i)\}$, then $\phi(e_i) \perp\!\!\!\perp \psi_1(e_j), \psi_2(e_j)$. Furthermore, if $e_j, \pi(e_j) \notin \{e_i, \pi(e_i)\}$, then $\phi(e_i) \perp\!\!\!\perp \phi(e_j)$. Following these independence arguments, we can conclude that indicators associated with different chains and cycles are mutually independent, and these indicators are correlated only with their precedent and subsequent terms in induced sequences.

2.C Labeling the Indicator Terms

In this appendix, we show that (i) there is an efficient algorithm for labeling the indicator terms to break the dependency between them; and (ii) based on this labeling strategy, we derive a bound for the concentration of X_π around its expected value.

In Lemmas 8 and 9, we defined induced sequences of indicator terms and characterized their correlation. Now we explain how to label each indicator term with alternating 0 and 1 labels in a way such that almost all of the indicators with the same label are independent. This is true for all terms except for those that are at the beginning and the end of cycles with an odd length: although they have the same label, but they are not independent. A sufficient condition for a successful labeling strategy, which can help us to derive good concentration bounds, is that for each type of indicators $\phi(e)$ and $\psi_{1,2}(e)$ at least a constant fraction of them should be labeled with 0 and a constant fraction of them with 1.

For a sequence of indicators $\big(\phi(e_1), \cdots \phi(e_i), \cdots \phi(e_q)\big)$ induced by a cycle (See Lemma 8), we start with a pair $\phi(e_1)$ and label it with $m\big(\phi(e_1)\big) = 0$. Next, we label $\phi(e_2)$ with 1, $\phi(e_3)$ with 0 and so on. We continue the next sequence with a new label (if we ended with 1 then we start with 0 and vice versa) until there is no more unlabeled cycle.

For sequences, which are induced by chains, the labeling strategy is more complicated. First, note that we can iteratively label a sequence from its beginning or its end. Second, remind that all the indicators induced by pairs e_1 and e_q (i.e., the beginning and end of chains) are either type $\phi(e)$ for a $e \in \mathcal{E}_{\emptyset,M*} \cup \mathcal{E}_{\emptyset,*M}$ or type $\psi(e)$ for a $e \in \mathcal{E}_{M,M\emptyset} \cup \mathcal{E}_{M,\emptyset M} \cup \mathcal{E}_{M,\emptyset\emptyset}$. Now, we label all the sequences of indicators, which are induced by chains, in the following five steps:

1. Take sequences that start (or end) with an indicator of type $\phi(e)$ and label $\phi(e)$ with $m\big(\phi(e)\big) = 0$. Next label $\phi(\pi(e))$ (or $\phi\big(\pi^{-1}(e)\big)$) with 1 and so on.
2. Take sequences that start (or end) with an indicator of type $\psi(e)$ and label $\psi(e)$ with $m\big(\psi(e)\big) = 0$. Next we label $\phi(\pi(e))$ (or $\phi\big(\pi^{-1}(e)\big)$) with 1 and so on.

3. Take sequences that start (or end) with an indicator of type $\phi(e)$ and label $\phi(e)$ with $m\left(\phi(e)\right) = 1$. Next label $\phi\left(\pi(e)\right)$ (or $\phi\left(\pi^{-1}(e)\right)$) with 0 and so on.

4. Take sequences that starts (or ends) with an indicator of type $\psi(e)$ and label $\psi(e)$ with $m\left(\psi(e)\right) = 1$. Next label $\phi\left(\pi(e)\right)$ (or $\phi\left(\pi^{-1}(e)\right)$) with 0 and so on.

5. Then, we continue by labeling the remaining sequences with an alternating 0 and 1 labels.

Lemma 11. *The labeling strategy assigns the labels 0 and 1 to the indicator terms $\{\phi(e) \cup \psi_{1,2}(e)\}$ in a way such that*

1. *at least $\frac{1}{6}$ of indicators of type $\{\psi_1(e) \cup \psi_2(e)\}$ from pairs in $\{\mathcal{E}_{M,\emptyset M} \cup \mathcal{E}_{M,\emptyset\emptyset}\}$ and $\{\mathcal{E}_{M,M\emptyset} \cup \mathcal{E}_{M,\emptyset\emptyset}\}$ is label with 0, and at least $\frac{1}{6}$ of them is labeled with 1.*

2. *at least $\frac{1}{3}$ of indicators induced by pairs in $\{\mathcal{E}_{\emptyset,M*} \cup \mathcal{E}_{M,M\emptyset} \cup \mathcal{E}_{M,MM}\}$ is labeled with 0, and at least $\frac{1}{3}$ of them is labeled with 1.*

3. *if $m\left(\phi(e_1)\right) = m\left(\phi(e_2)\right)$ and $e_1 \neq \pi^c(e_2)$ for some $c \geq 0$, then $\phi(e_1)$ and $\phi(e_2)$ are independent. The same is true for indicators of type $\psi_{1,2}$.*

Proof. We start by proving the first clause of the lemma. At each iteration, out of eight considered start and end indicators (i.e., four starts and four ends) at least two and at most six terms have type ψ. Out of these six, at least one is labeled with 0 at step 2 and at least one labeled with 1 at step 4 of the labeling procedure (which exactly amounts to at least $\frac{1}{6}$ of the considered subset). If we are in the case that there is no more chain that is starting or ending from an indicator ϕ, we label every second chain-start with 0. In this case, at least $\frac{1}{4}$ of indicators of type ψ is labeled with 0. The same argument is true for label 1.

To proof the second clause of the lemma, consider indicators of type $\{\phi(e)\}$ from pairs $\{\mathcal{E}_{\emptyset,M*} \cup \mathcal{E}_{M,M\emptyset} \cup \mathcal{E}_{M,MM}\}$. For the indicators induced by cycles, we start labeling with 0, and alternating 0 and 1. Thus approximately (depending if we stopped at 0 or 1) half of the pairs is labeled with 0 and the rest is labeled with 1. For the chains, at least $\frac{1}{6}$ of start (and end) indicators of type ϕ is labeled with 1 and the same for label 0 (the argument here is the same as for indicators of type ψ). For internal indicators, as we alternate the start counter at each iteration, at least $\frac{1}{3}$ of the indicators is labeled with 0 and at least $\frac{1}{3}$ of the indicators is labeled with 1.

The final statement of the lemma follows directly from the definition of the chains and cycles. □

For the rest of this section, for simplicity of notation, we assume $m(e) = m\left(\phi(e)\right)$ and $m(e) = m\left(\psi(e)\right)$.

Using the introduced labeling strategy, we split the $X_\pi = S_1 + S_2$ into two terms S_1 and S_2, such that

$$S_1 = \sum_{\substack{e \in \mathcal{E}_{\emptyset,M*} \cup \mathcal{E}_{M,M\emptyset} \cup \mathcal{E}_{M,MM} \\ m(e)=0}} \phi(e) + \alpha \left[\sum_{\substack{e \in \mathcal{E}_{M,\emptyset M} \cup \mathcal{E}_{M,\emptyset\emptyset} \\ m(e)=0}} \psi_1(e) + \sum_{\substack{e \in \mathcal{E}_{M,M\emptyset} \cup \mathcal{E}_{M,\emptyset\emptyset} \\ m(e)=0}} \psi_2(e) \right]$$

and

$$S_2 = \sum_{\substack{e \in \mathcal{E}_{\emptyset,M*} \cup \mathcal{E}_{M,M\emptyset} \cup \mathcal{E}_{M,MM} \\ m(e)=1}} \phi(e) + \alpha \left[\sum_{\substack{e \in \mathcal{E}_{M,\emptyset M} \cup \mathcal{E}_{M,\emptyset\emptyset} \\ m(e)=1}} \psi_1(e) + \sum_{\substack{e \in \mathcal{E}_{M,M\emptyset} \cup \mathcal{E}_{M,\emptyset\emptyset} \\ m(e)=1}} \psi_2(e) \right]$$

Lemma 12. *We have*

$$\mathbb{E}[S_1] \geq \frac{\mathbb{E}[X_\pi]}{6} \text{ and } \mathbb{E}[S_2] \geq \frac{\mathbb{E}[X_\pi]}{6}.$$

Proof. This follows directly from Lemma 11 and the linearity of expectation. □

Lemma 13. *Denote by* $\mu_1 = \mathbb{E}[X_\pi]$ *and by* $\mu_2 = \mathbb{E}[Y_\pi]$. *We have*

$$\begin{aligned} \mathbb{P}[X_\pi < \frac{\mu_1+\mu_2}{2}] &\leq 2\exp(-\frac{(\mu_1-\mu_2)^2}{96\mu_1}), \\ \mathbb{P}[Y_\pi > \frac{\mu_1+\mu_2}{2}] &\leq \exp(-\frac{(\mu_1-\mu_2)^2}{12\mu_1}). \end{aligned}$$

Proof. As $X_\pi = S_1 + S_2$, then

$$\begin{aligned} \mathbb{P}[X_\pi < (1-\epsilon)\mu_1] &\leq \mathbb{P}\left[S_1 < (1-\epsilon)\mathbb{E}[S_1] \bigcup S_2 < (1-\epsilon)\mathbb{E}[S_2]\right] \\ &\leq \mathbb{P}\left[S_1 < (1-\epsilon)\mathbb{E}[S_1]\right] + \mathbb{P}\left[S_2 < (1-\epsilon)\mathbb{E}[S_2]\right]. \end{aligned}$$

We first prove that $\mathbb{P}[S_1 < (1-\epsilon)\mathbb{E}[S_1])$. From the result of Lemma 11, we know that all the terms in S_1 are independent except for those that are the beginning and end of cycles with odd lengths. For those cycles $\phi(e_1), \ldots, \phi(e_c)$, we introduce a new variable $W_{e_1} = \frac{\phi(e_1)+\phi(e_c)}{2}$. And for the rest of indicators, we define $W_e = \frac{\phi(e)}{2}$. Note that if $W = \sum W_{e_i}$, then $2W = S_1$ and all W_e

terms are independent. Consequently, we have

$$\begin{aligned}
&\mathbb{P}\left[S_1 < (1-\epsilon)\mathbb{E}[S_1]\right] = \mathbb{P}\left[\sum W_i < (1-\epsilon)\mathbb{E}[W]\right] \\
&\leq \exp\left(-\frac{\mathbb{E}[W]\epsilon^2}{2}\right) \text{ (by the Chernoff-Hoeffding bound from Appendix 2.D)} \\
&\leq \exp\left(-\frac{\mathbb{E}[S_1]\epsilon^2}{4}\right) \leq \exp\left(-\frac{\mathbb{E}[X_\pi]\epsilon^2}{24}\right) \text{ (by Lemma 12).}
\end{aligned}$$

To prove the lemma, we set $\epsilon = \frac{\mu_1-\mu_2}{2\mu_1}$. As we have $\frac{\mu_1+\mu_2}{2} = \mu_1 - \frac{\mu_1-\mu_2}{2} = \mu_1(1-\frac{\mu_1-\mu_2}{2\mu_1})$, we can conclude the bound for S_1. By proving the bound for S_2 in a similar way, we obtain an upper bound for X_π with a high probability.

For μ_2, we can write similarly $\frac{\mu_1+\mu_2}{2} = \mu_2(1+\frac{\mu_1-\mu_2}{2\mu_1})$. The result for Y_π follows directly from a Chernoff bound due to the fact all of its terms are independent. □

2.D Chernoff-Hoeffding Lemma

Lemma 14. [Chernoff-Hoeffding bound [54]]
Let $X \triangleq \sum_{i=1}^n X_i$ *where* $X_i, 1 \leq i \leq n$, *are independently distributed in* $[0,1]$. *Then for* $\epsilon > 0$,

$$\begin{aligned}
&\mathbb{P}\left[X > (1+\epsilon)\mathbb{E}[X]\right] \leq \exp\left(-\frac{\epsilon^2}{3}\mathbb{E}[X]\right), \\
&\mathbb{P}\left[X < (1-\epsilon)\mathbb{E}[X]\right] \leq \exp\left(-\frac{\epsilon^2}{2}\mathbb{E}[X]\right).
\end{aligned}$$

The computational perspective Part II

3 Percolation Graph Matching

The problem of identifying a (full or partial) matching between nodes in two structurally similar graphs is known as network alignment (or graph matching). In this problem, we are given two unlabeled graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$. We assume a true but unknown equivalence between nodes that are in the intersection of the two graphs. We denote such a hidden alignment by π_0. Given the two graphs $G_{1,2}$, the purpose of graph matching is to find the hidden matching π_0.

In Chapter 2, we looked at the problem of network alignment from a modeling and information-theoretic perspective. We showed that partially overlapping graphs can be correctly matched under quite mild conditions. For graphs $G_{1,2}$, a correct matching amounts to (i) filtering out nodes without counterparts in both G_1 and G_2, and then (ii) correctly matching the nodes in the intersection of the two graphs. Compared to the idealized situation of two fully overlapping graphs, it is quite remarkable that the added difficulty of (i) has such a benign effect on the fundamental ability of matching. Of course, this is true for the information-theoretic setting considered so far, where unlimited computational power is assumed. In this chapter[1], we complement our feasibility result by introducing an accurate and scalable graph-matching algorithm. We investigate the effects of partial node overlap and edge overlap on this algorithm, both theoretically and experimentally.

The best-studied and most scalable class of graph-matching algorithms assumes the existence of auxiliary information in the form of a set of pre-matched node-couples called *seeds*. These algorithms then incrementally build the full mapping from this pre-matched *seed set*. We refer to this class as percolation graph-matching (PGM) algorithms [40, 41, 109, 140, 202]. All the algorithms in the PGM class are based on the same key idea introduced by the work of Narayanan and Shmatikov [140]: A (small) subset of nodes across the two graphs are identified a priori and are matched. Then, in an iterative procedure, the matched couples “infect” neighboring potential matches, with some threshold criterion that turns a potential match into a permanent match. For example, in [202], the rule is extremely simple: (i) Every seed

---

[1] The material of this chapter is based on [99].

couple is considered matched; (ii) a node-couple $[i, j]$ is matched if it has at least r already matched neighbours[2] and i, j are not part of another matched couple already. Yartseva and Grossglauser [202] show that the recursive application of rule (ii) can, under some conditions, correctly match all the nodes.

Narayanan and Shmatikov [140] empirically observe that their seed-based heuristic PGM algorithm is sensitive to the seed-set size. Furthermore, Yartseva and Grossglauser [202] rigorously analyze the PGM algorithms, within a random graph model from [150] (they assume the same vertex set for the two graphs), and they characterize a phase transition. More precisely, they prove that if the number of couples in the seed set (where all of them are correct) is above a threshold and $r \geq 4$, then the PGM algorithm correctly matches almost all the nodes; and if the seed set is too small, the percolation does not take off, hence dying young. A similar model is analyzed in [40, 109].

In this chapter, we give a new PGM algorithm by separating the decision to match a couple from the use of a potential match as evidence for other matches. The distinguishing feature of this algorithm is that it requires a dramatically smaller number of seeds, in comparison to other algorithms from [40, 41, 109, 140, 202]. Whereas Yartseva and Grossglauser [202] prove that, with high probability, their algorithm matches every node couple (with zero errors) for $r \geq 4$, this performance criterion has to be slightly relaxed. Specifically, we would be content with a vanishing fraction of incorrectly aligned couples (with high probability). In summary, we manage to trade-off a very significant reduction in the seed-set size, with a fairly benign increase in the error rate.

The reason this works is quite subtle: For a PGM algorithm to succeed, two conditions have to be satisfied. First, the algorithm has to percolate: if at some point, there is not enough evidence to match a new couple, the algorithm stops. If this happens before a significant portion of the nodes have been matched, the algorithm fails. Second, if the algorithm does percolate, it has to percolate correctly. If at some point, the evidence for matching an incorrect couple is stronger than the evidence for any correct couple, then the algorithm makes an error. Furthermore, this incorrect match could percolate to other incorrect couples in future steps, thus (potentially) leading to a cascade of errors.

Clearly, there is a trade-off between these two conditions. This trade-off can be controlled by the strength of the required evidence for permanently matching a couple. For example, consider the parameter r above: if r is chosen quite high ($r = 5$, say), then percolation might easily stop early; however, a high r makes errors less likely; for $r = 2$, the algorithm percolates easily, but might often incorrectly match couples.

In this chapter, we control this trade-off in a different way, by decoupling the decision to match a couple from its ability to infect other couples. We refer to a tentative couple that is not yet matched as a *candidate couple*. Essentially, a candidate couple provides evidence for

---

[2] Two couples $[i, j]$ and $[i', j']$ are called neighbours if there is an edge (i, i') in E_1, and an edge (j, j') in E_2.

other couples, thereby fueling the percolation process, but this couple is not yet matched. It is not a priori obvious that this decoupling is a good idea; showing this is the key theoretical contribution of this chapter. The reason is not obvious and has to do with the way the evidence for correct and incorrect couples percolates. Basically, correct couples tend to infect a small number of neighbouring correct couples, each with relatively high probability; incorrect couples tend to infect only other incorrect couples, but crucially this effect is uniform over all incorrect couples and becomes “diluted”.

This observation leads us to create an algorithm that is highly robust to incorrect candidate couples. We prove that under a wide range of network parameters, this algorithm will percolate with high probability, generating a large number of incorrect candidate couples along the way. However, the majority of matched couples are correct. It is important to note that our algorithm has specific provisions for treating (i) filtering and (ii) matching. Consequently, we might suspect that this algorithm could be quite sensitive to partial node overlap. One of the key findings in this chapter is that this is not the case: We observe that quite often, the percolation process over node-couples matches correct couples, and then stops “at the right time”, i.e., when it has exhausted the nodes that are in the interaction of both vertex sets.

In summary, our contributions in this chapter are as follows:

- We develop a new graph-matching algorithm called `ExpandWhenStuck`. The distinguishing feature of this algorithm is that it requires a dramatically smaller number of seeds, in comparison to state-of-the-art algorithms [109, 202]. It is able to match real-social networks with over a million nodes and various types of random graphs (for example, Barabási-Albert [21], Chung-Lu [42] and Erdős-Rényi [58] graphs), by using only a handful of seeds (see Section 3.4).
- We analyze the performance of a simplified version of the `ExpandWhenStuck` algorithm over an Erdős-Rényi random-bigraph model with partial-overlapping vertex sets.[3] The simplification is needed to make the analysis tractable: Whereas `ExpandWhenStuck` dynamically percolates from unmatched candidate couples whenever necessary, we can rigorously analyze only a slightly more restrictive setting (the `ExpandOnce` algorithm), where this occurs only once at the outset. Specifically, `ExpandOnce` expands the seed set into a larger set that includes many incorrect couples; a second algorithm called `NoisySeeds` then percolates from this latter set in a manner similar to [202]. In Section 3.2, (i) we demonstrate a phase transition in the number of required seeds, as a function of the network size, overlap between the two graphs, and structural similarity, and (ii) we prove that the algorithm is robust to partial node-overlap. More precisely, we prove that the proposed PGM algorithm naturally filters out the nodes that are without counterparts in the other graph and correctly matches the rest.

The remainder of this chapter is organized as follows. In Section 3.1, we explain our proposed

---

[3] The model is introduced in Section 2.1 from Chapter 2.

PGM algorithm. In Section 3.2, we prove a performance guarantee for the algorithm of Section 3.1. In Section 3.3, using the ideas from previous sections, we present a heuristic algorithm whose performance is better in practice. In Section 3.4, we report the simulation results of our algorithms over real and random graphs. We compare our proposed algorithms with two state-of-the-art graph matching algorithms [109, 202] over several real graphs. In Section 3.5, we conclude this chapter.

3.1 Algorithms

In this section, we define and explain the `ExpandOnce` algorithm that first performs one round of expansion of the initial seed-set and then applies a novel PGM algorithm, called `NoisySeeds`, over the expanded seed set. This expansion step helps the percolation process overcome the bottleneck due to a small seed-set size. This algorithm is kept deliberately simple for mathematical tractability. We provide an intuitive explanation for the performance of our approach based on the model from Section 2.1. A rigorous analysis of our algorithm is then provided in Section 3.2. A more practical but heuristic algorithm, based on the key ideas developed here, will be presented in Section 3.3.

3.1.1 Notation

Let us introduce the necessary notation. For graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ let $V_0 = V_1 \cap V_2$. We assume $n_1 = |V_1|, n_2 = |V_2|$ and $n_0 = |V_0|$. Also, $d_{1,i}$ and $d_{2,j}$ denote degrees of nodes i and j in graphs G_1 and G_2, respectively. Let pairs $(i, i') \in E_1$ and $(j, j') \in E_2$ represent the edges between nodes i, i' and j, j' in graphs G_1 and G_2, respectively; and $[i, j]$ represent a *couple* of nodes where $i \in V_1$ and $j \in V_2$. A couple $[i', j'] \in V_1 \times V_2$ is a neighbour of another couple $[i, j]$ if $(i, i') \in E_1$ and $(j, j') \in E_2$. Indeed, the evidence for deciding which couple to match is the number of common neighbors (called the score of a couple) each couple has in the set of currently used seed couples. We refer to the process of *spreading out marks* from a seed couple $[i, j]$ as adding one mark to each of its neighbouring couples. Therefore, the score of a couple is defined, equivalently, as the number of marks it has received from other couples in the matching process.

For convenience of notation, without loss of generality we assume that the hidden correct mapping between the nodes in V_0 is the identity mapping. Therefore, a couple is correct if and only if it is of the form $[i, i]$. Let $\Lambda(\mathcal{S})$ denote the number of correct couples in a set $\mathcal{S}$ of couples, and let $\Psi(\mathcal{S})$ represent the number of incorrect couples. Also, $V_1(\mathcal{S})$ is the set of nodes from graph G_1 in a set of couples $\mathcal{S}$, i.e., $V_1(\mathcal{S}) = \{i | \exists j \text{ s.t. } [i, j] \in \mathcal{S}\}$. Similarly, we define $V_2(\mathcal{S}) = \{j | \exists i \text{ s.t. } [i, j] \in \mathcal{S}\}$. Table 5.2 summarizes all the notations used in this chapter.

3.1.2 NoisySeeds Algorithm

In this part, we first give a new PGM algorithm, called `NoisySeeds`; it is a main building-block of the `ExpandOnce` algorithm. Before that, for the sake of better illustration, we explain the algorithm from [202] that in this chapter we refer to as `PercolateMatched`.

The `PercolateMatched` algorithm starts from an initial seed-set (a predefined matching) and iteratively matches couples having at least r matched neighbours. More specifically, (i) initially we are given as inputs a set of a_0 predefined (and correct) matched couples, called seed set $\mathcal{A} = \mathcal{A}_0$ ($|\mathcal{A}_0| = a_0$), and a fixed threshold r; (ii) at each time step τ, the algorithm picks an unused couple from set $\mathcal{A}$ and spreads marks to all of its neighbouring couples; (iii) as soon as a couple obtains at least r marks, i.e., it is a neighbour of at least r used couples in the set $\mathcal{A}$, it will be added to the set $\mathcal{A}$; and (iv) the algorithm stops when there exist no further unused couples in the set $\mathcal{A}$. The `User-Matching` algorithm [109] is similar to `PercolateMatched`, with a slight difference: nodes are matched in several rounds based on a simple degree-bucketing method that matches high-degree nodes first.

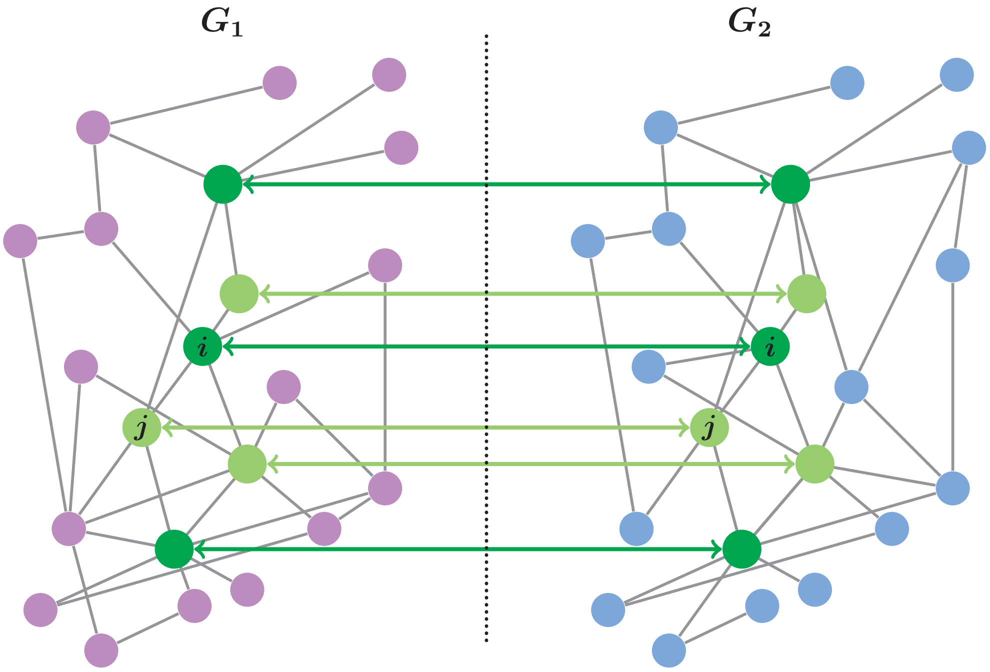


Figure 3.1 – The `PercolateMatched` algorithm: It starts from a predefined matching called seed set and iteratively matches couples having at least r matched neighbours. Dark-green nodes are initial seeds (e.g., couple $[i, i]$). Light-green nodes are the newly matched couples after the first three iterations (e.g., couple $[j, j]$). These matched couples are served as new seeds in later steps. In this example, we set $r = 2$.

The success of the `PercolateMatched` algorithm heavily relies on the condition that all the matched couples (including the initial seed-couples) are indeed correct couples. In order for

`PercolateMatched` to succeed, this condition then results in some constraints on r, namely $r \geq 4$. Our main theoretical contribution in this chapter is to show that (i) the matching process can be made robust to a large number of incorrect couples in the seed set, provided there are enough correct couples in the seed set as well; and (ii) $r = 2$ is sufficient to match almost all the nodes correctly.

The `NoisySeeds` algorithm (see Algorithm 1) starts with an initial noisy seed set $\mathcal{A}_0$, i.e., a set with possibly many incorrect couples. First, the marks coming from all the couples in $\mathcal{A}_0$ are computed at the beginning (lines 1 to 4) and all these couples are added to the set of used couples $\mathcal{Z}$ (line 5). The algorithm proceeds as follows. We consider a set of matched couples, denoted by $\mathcal{M}$, which is initially empty. If there is any couple with a score of at least r, then we add this couple to the matched set $\mathcal{M}$. Each time a couple $[i, j] \in \mathcal{M} \setminus \mathcal{Z}$ is chosen randomly, it spreads out marks to its neighbouring couples and is added to $\mathcal{Z}$. Because the couple $[i, j]$ is in the matching $\mathcal{M}$, any other couple in the form of $[i, j']$ or $[i', j]$ is not a candidate for matching any longer and is permanently removed from consideration.

The percolation process stops if there is no remaining unused couple with a score of at least r. Note that as not all the couples in the noisy set $\mathcal{A}_0$ are necessarily correct, they are not added to the matched set initially, i.e., the matched set is decoupled from the seed set. These couples are used only for the sake of creating an initial set of marks for different couples associated with the two graphs.

Example 15. The execution of `NoisySeeds` after four iterations (for $r = 2$) is illustrated in Figure 3.2. `NoisySeeds` begins by spreading out marks from the initial noisy seed-set (dark-green and red nodes in Figure 3.2). Afterwards, all the newly matched couples (light-green and red nodes in Figure 3.2) are added to the seed set, and the matching process continues by spreading out their marks.

Algorithm 1: `NoisySeeds`

```
Input: $G_1(V_1, E_1), G_2(V_2, E_2)$, noisy seed set $\mathcal{A}_0$ and threshold $r$
Output: The set of matched couples $\mathcal{M}$
1  for all couples $[i, j] \in \mathcal{A}_0$ do
2      add one mark to all the neighbouring couples of $[i, j]$;
3      if score of a couple $[i', j'] \geq r$ and $i' \notin V_1(\mathcal{M})$ and $j' \notin V_2(\mathcal{M})$ then
4          add $[i', j']$ to the set $\mathcal{M}$;
5  $\mathcal{Z} \leftarrow \mathcal{A}_0$;
6  while $\mathcal{M} \setminus \mathcal{Z} \neq \emptyset$ do
7      randomly choose a couple $[i, j] \in \mathcal{M} \setminus \mathcal{Z}$ and add $[i, j]$ to the set $\mathcal{Z}$;
8      add one mark to all the neighbouring couples of $[i, j]$;
9      if score of a couple $[i', j'] \geq r$ and $i' \notin V_1(\mathcal{M})$ and $j' \notin V_2(\mathcal{M})$ then
10         add $[i', j']$ to the set $\mathcal{M}$;
11 return $\mathcal{M}$;
```

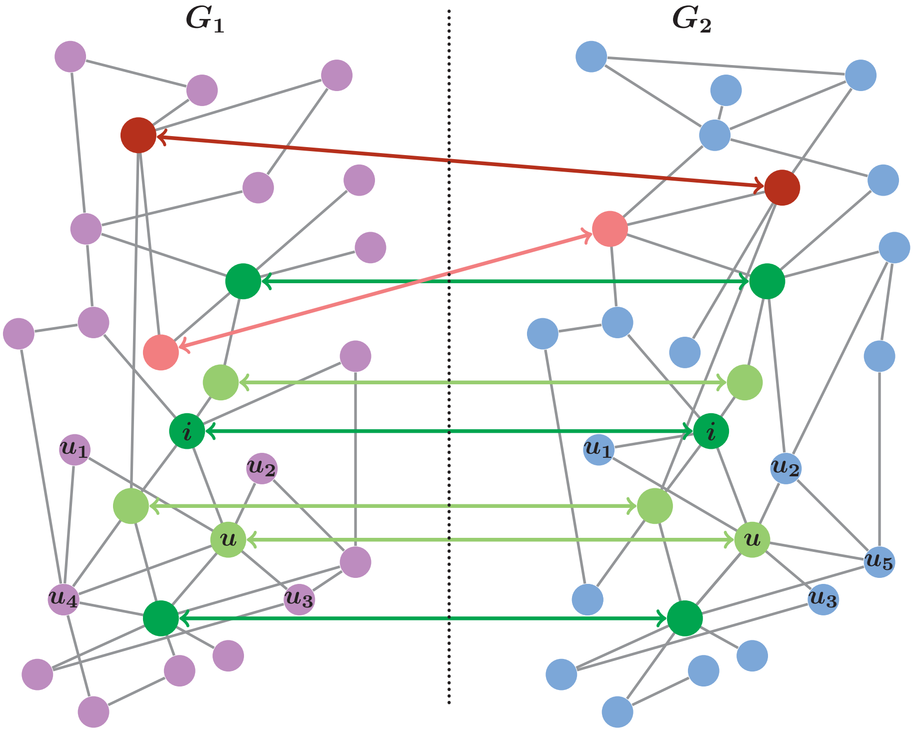


Figure 3.2 – The `NoisySeeds` algorithm: Dark-green and dark-red nodes correspond to the initial correct and incorrect seed-couples, respectively. After the first four iterations (for $r = 2$), light-green nodes form correctly matched couples, and light-red nodes form incorrectly matched couple (see Example 15).

A convenient way to evaluate graph-matching algorithms is to analyze their performance over the $G(n, p; t, s)$ model, a parsimonious model for generating two correlated graphs with partially overlapping vertex sets (see Section 2.1.2). In Section 3.2, by using the $G(n, p; t, s)$ model, we prove that the `NoisySeeds` algorithm is robust to the noise in the seed set. An intuitive explanation for this robustness is as follows: (i) A correct couple obtains a mark from any other correct couple with probability ps^2 (an edge exists in the generator graph with a probability p and is sampled in both $G_{1,2}$ with a probability s^2). Also, an incorrect couple obtains a mark from any other incorrect or correct couple with probability p^2s^2 (it corresponds to two different edges in the generator graph). Note that $p^2s^2 \ll ps^2$. Thus, the effect of spreading marks from an incorrect couple, compared to a correct couple is negligible. (ii) Consider a couple that contains a node without any counterpart in the other graph (this is necessarily an incorrect couple): This couple obtains $r \geq 2$ marks from any other r couples with probability at most $O(p^4s^4)$. Therefore, only a small fraction of incorrect couples (in expectation $O(n_1^2 n_2^2 p^4 s^4)$) obtains more than one mark.

3.1.3 ExpandOnce Algorithm

In this sub-section, we introduce the `ExpandOnce` algorithm that trades-off a small decrease in precision, relative to `PercolateMatched`, with a dramatic reduction in the seed-set size. This algorithm accepts as input a seed set $\mathcal{A}_0$ of correct couples. It expands the seed set $\mathcal{A}_0$ to a larger set of candidate couples $\mathcal{A}'_0$ of size a'. Then, it runs `NoisySeeds` over the expanded seed-set. In other words, in its first step, `ExpandOnce` creates, from a small set of correct couples ($\mathcal{A}_0$), a larger set of candidate couples $\mathcal{A}'_0$, many of which are incorrect in general. In Section 3.2, we will prove that these incorrect couples in $\mathcal{A}'_0$ have only a negligible effect on the performance of the matching process in `ExpandOnce`. Also, the new correct couples in $\mathcal{A}'_0$ (rather than those from $\mathcal{A}_0$) enable the percolation process to kick-off. As a result, by calling `NoisySeeds` on the expanded seed set $\mathcal{A}'_0$, we obtain a successful matching of the two graphs. In summary, the process of expanding correct couples to a noisy seed-set enables us to successfully match graphs with much fewer initial seeds.[4] For a schematic overview of the `ExpandOnce` algorithm refer to Figure 3.3. Algorithm 2 explains `ExpandOnce` in detail.

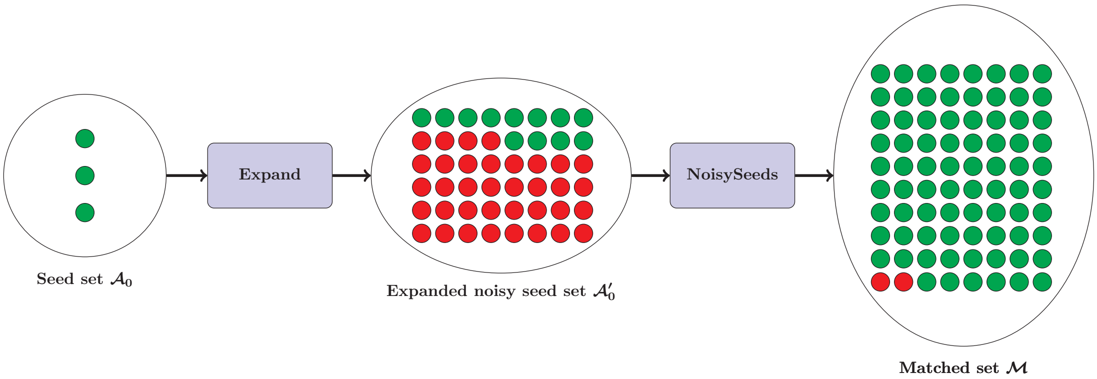

Seed set $\mathcal{A}_0$
Expand
Expanded noisy seed set $\mathcal{A}'_0$
NoisySeeds
Matched set $\mathcal{M}$

Figure 3.3 – The `ExpandOnce` algorithm: Input to the algorithm is a seed set of correct couples. This algorithm expands the initial seed-set to a larger set of candidate couples with more correct couples and many incorrect couples. Then, it runs the `NoisySeeds` algorithm over the expanded seed-set. In this figure, green and red circles represent correct and incorrect couples, respectively.

3.2 Performance of Matching with Noisy Seeds

In this section, (i) we identify a phase transition in the seed-set size of `NoisySeeds` (explained in Algorithm 1); (ii) we prove `NoisySeeds` correctly matches almost all the nodes that are in the intersection of the two graphs and filters-out the nodes without counterparts in the other graph; and (iii) we prove the addition of many incorrect couples to the initial correct seed set $\mathcal{A}_0$ of `NoisySeeds` would have a negligible effect on the performance of this algorithm.

---

[4] Experiments over different types of graphs show that expanding an initial correct seed set $\mathcal{A}_0$ to the noisy seed set $\mathcal{A}'_0$ whose size is of order of $\min(n_1, n_2)$ results in an excellent matching performance.

Algorithm 2: `ExpandOnce`

```
Input: $G_1(V_1,E_1), G_2(V_2,E_2)$, seed set $\mathcal{A}_0$ of correct couples, integer value $a' \ge 1$ and threshold $r$
Output: The set of matched couples $\mathcal{M}$
1 $\mathcal{A}'_0 \leftarrow \mathcal{A}_0$ and $\mathcal{A} \leftarrow \mathcal{A}_0$;
2 while $|\mathcal{A}'_0| < a'$ do
3     $\mathcal{Z} \leftarrow \emptyset$ and $\mathcal{U} \leftarrow \mathcal{A}'_0$;
4     for all couples $[i,j] \in \mathcal{A}$ do
5         for all neighbouring couples $[i',j']$ of couple $[i,j]$ do
6             if $|\mathcal{A}'_0| < a'$ and $i' \notin V_1(\mathcal{U})$ and $j' \notin V_2(\mathcal{U})$ then
7                 add $[i',j']$ to $\mathcal{A}'_0$ and $\mathcal{Z}$;
8     $\mathcal{A} \leftarrow \mathcal{Z}$;
9 return $\mathcal{M} \leftarrow$ NoisySeeds($G_1, G_2, \mathcal{A}'_0, r$);
```

The robustness guarantee for the `NoisySeeds` algorithm, with respect to a noisy seed-set, explains why `ExpandOnce` (Algorithm 2) requires a small set of initial seeds. Indeed, lines 2 to 8 of `ExpandOnce` turn a small set of clean seeds into a large noisy seed-set that contains both correct and incorrect couples. This new set is then fed into `NoisySeeds` as an input and, as this algorithm is robust to incorrect couples, it succeeds with high probability.

Here, we analyze a simplified variant of the `ExpandOnce` algorithm. We assume that $\mathcal{A}_0$ is a random set in the following sense: each correct couple $[i,i] \in V_0^2$ is placed in $\mathcal{A}_0$ independently, with a constant probability λ. Also, each incorrect couple $[i,j]$, $i \neq j \in V_1 \times V_2$, is placed in $\mathcal{A}_0$ independently, with a constant probability ψ. Hence, we expect λn correct couples and $\psi n(n-1)$ incorrect couples as the initial noisy seed-set for the `NoisySeeds` algorithm. Throughout this section, we assume that the number of nodes n and average degree np tend to infinity. We also assume that the nodes and edge sampling probabilities $0 < t, s \le 1$ are arbitrary constants. Let $\mathcal{Z}_\tau$ and $\mathcal{M}_\tau$ be, respectively, the set of used and matched couples at time step τ of `NoisySeeds`. Also, let $\mathcal{M}^*$ denote the final set of matched couples from `NoisySeeds`. We now state our main theorem in this chapter.

We first define two parameters $b_{t,s,r}$ and $a_{t,s,r}$ [86, 202]:

$$b_{t,s,r} = \left[\frac{(r-1)!}{nt^2(ps^2)^r}\right]^{\frac{1}{r-1}} \text{ and } a_{t,s,r} = \left(1-\frac{1}{r}\right) b_{t,s,r} = \left(1-\frac{1}{r}\right)\left[\frac{(r-1)!}{nt^2(ps^2)^r}\right]^{\frac{1}{r-1}}. \tag{3.1}$$

Theorem 16 (Robustness of `NoisySeeds`)**.** *For an arbitrarily small but fixed $\frac{1}{6} > \epsilon > 0$, assume that $n^{-1} \ll p \le n^{-\frac{5}{6}-\epsilon}$. If all the couples in the noisy seed-set $\mathcal{A}_0$ are chosen uniformly at random, such that the expected number of correct couples is $\mathbb{E}[\Lambda(\mathcal{A}_0)] > (1+\epsilon)a_{t,s,r}$ and the expected number of incorrect couples is $\mathbb{E}[\Psi(\mathcal{A}_0)] \le wn$ for a constant w,*[5] *then with high probability the* `NoisySeeds` *algorithm percolates and the size of its final matching is $nt^2 \pm o(n)$*

---

[5] Note that in general the algorithm is robust to a number of additional incorrect couples (in the seed set), which scales with n and p. Here, we have chosen this number to be wn in order to simplify our statements.

with $\Lambda(\mathcal{M}^*) = nt^2 \pm o(n)$ *and* $\Psi(\mathcal{M}^*) = o(n)$.

Yartseva and Grossglauser [202] proved that it is possible to correctly match almost all the nodes under the following limited assumptions: (i) the vertex sets of the two graphs $G_{1,2}$ are exactly the same, i.e., $t = 1$ in our model, (ii) there is no incorrect couple in the initial seed set, i.e., $\Psi(\mathcal{A}_0) = 0$, and (iii) the size of initial seed set is at least $a_{1,s,4}$ $(r = 4)$. For this special case, Theorem 16 guarantees that a seed set of size $a_{1,s,2}$ $(r = 2)$ is enough for matching almost all the nodes correctly with a vanishing fraction of errors. Note further that the ratio $a_{1,s,4}/a_{1,s,2}$ goes to infinity which amounts to a huge increase in the required number of seeds. Also, with $r = 4$, we are able to align only nodes with degrees at least four in the two networks.

Next, we prove Theorem 16 for the case $r = 2$; it needs the least number of seeds, i,e., $a_{t,s,r}$ is minimized when $r = 2$. Generalization for values $r > 2$ is straightforward. For ease of notation, we define $a_c = a_{t,s,2} = \frac{1}{2nt^2p^2s^4}$. We first provide a brief sketch for the proof of Theorem 16. The detailed proof is given afterwards.

3.2.1 Sketch of the Proof

In the beginning of `NoisySeeds`, all the couples in the seed set $\mathcal{A}_0$ spread out marks to their neighbouring couples. Then, at each time step $\tau \geq 1$, one couple from $\mathcal{M}_\tau \setminus \mathcal{Z}_\tau$ is picked and spreads out marks to its neighbouring couples. It is easy to see that the matching process stops at a time step T^*, where $|\mathcal{M}^*| = |\mathcal{M}_{T^*}| = T^*$ (i.e., T^* is the first time when all the couples inside $\mathcal{M}_{T^*}$ have already been picked). Note that T^* is at most $\min(n_1, n_2)$, as each node can be matched at most once. In order to prove Theorem 16, we show that with high probability (w.h.p.) $T^* = nt^2 \pm o(n)$, and the number of incorrectly matched couples is at most $o(a_c)$. More precisely, the proof can be summarized in the following two steps:

(a) We provide an upper-bound on the number of incorrectly matched couples at each step of the algorithm through computing its expected value. Using this upper-bound we prove that the effect of incorrect couples is negligible in the final result of `NoisySeeds` (see Lemmas 17 to 19).

(b) By using step (a) and the results from the bootstrap percolation process [86, 202], we prove that w.h.p. the correct couples percolate when their initial number $\Lambda(\mathcal{A}_0)$ is more than the percolation threshold a_c. Therefore, as a result of percolation of the correct couples, the number of correctly matched couples at time step T^* is $nt^2 \pm o(n)$.

3.2.2 Proof of Theorem 16

Let us first introduce the notations used in this subsection. For an integer ℓ let $\mathcal{P}_{\ell,\tau}$ denote the set of couples with score ℓ at time step τ; also let $\mathcal{P}_{\geq\ell,\tau}$ be the set of couples with score at least ℓ at time τ. We let $\mathcal{Z}_\tau$ and $\mathcal{M}_\tau$ be the set of used and matched couples at time step τ, respectively. Assume the time τ_{cip} corresponds to the completion of the initial phase (cip) of the algorithm, i.e., at the time τ_{cip} all the initial seeds are used for spreading out marks. All the other notations are explained in Section 3.1.1.

In the beginning of `NoisySeeds` (lines 1 to 4 in Algorithm 1), all the couples in the seed set $\mathcal{A}_0$ spread out marks to their neighbouring couples. Afterwards, at each time step $\tau \geq 1$ (lines 5 to 10 in Algorithm 1), one couple from $\mathcal{M}_\tau \setminus \mathcal{Z}_\tau$ is picked and spreads marks to its neighbouring couples. The matching process stops at a time step T^*, where $|\mathcal{M}^*| = |\mathcal{M}_{T^*}| = T^*$. Note that T^* is at most $\min(n_1, n_2)$, as each node can be matched at most once. In order to prove Theorem 16, we will show that w.h.p. $T^* = n_0 - o(n)$. Using the Chernoff bound, with high probability $n_0 = nt^2 \pm o(n)$. Therefore, we have $T^* = nt^2 \pm o(n)$, and the number of incorrectly matched couples is at most $o(a_c)$. More precisely, we bound the number of incorrectly matched couples at each step through computing their expected value. Using this upper-bound, we prove that the effect of incorrect couples is negligible. Also, we prove that the correct couples percolate, if the number of initial correct-seeds $\Lambda(\mathcal{A}_0)$ is more than the percolation threshold a_c.

We proceed by computing the expected number of incorrectly matched couples at time τ_{cip}.

Lemma 17. $\mathbb{E}\left[\Psi(\mathcal{M}_{\tau_{\text{cip}}})\right] = O(w^2 n^4 p^4 s^4 t^2) = o(a_c)$.

Proof. We first recall that the time τ_{cip} corresponds to the completion of the initial phase (cip) of the `NoisySeeds` algorithm. We define the random variables $X_{i,j}, i \neq j$ as

$$X_{i,j} = \begin{cases} 1 & \text{if } [i,j] \in \mathcal{P}_{\geq 2,\tau_{\text{cip}}}, \text{i.e., couples with score at least 2 at the end of initial phase} \\ 0 & \text{o.w.} \end{cases}$$

and $X = \sum_{\forall [i,j], i \neq j} X_{i,j}$. Note that as each node can be matched at most once, X is an upper bound for the total number of incorrectly matched couples at time τ_{cip}, i.e., $\Psi(\mathcal{M}_{\tau_{\text{cip}}})$. Therefore, we have

$$\mathbb{E}\left[\Psi(\mathcal{M}_{\tau_{\text{cip}}})\right] \leq \mathbb{E}[X] = \sum_{\forall [i,j], i \neq j} \mathbb{E}\left[X_{i,j}\right] \leq n_1 n_2 \mathbb{P}\left[[i,j] \in \mathcal{P}_{\geq 2,\tau_{\text{cip}}}\right]. \tag{3.2}$$

We will prove that

$$\mathbb{P}\left[[i,j] \in \mathcal{P}_{2,\tau_{\text{cip}}}\right] \leq (n_1 n_2)^2 p^4 s^4 \left(\frac{wn}{n_1 n_2}\right)^2 \left[1 + O\left(\frac{1}{np}\right)\right] = O(n^2 w^2 p^4 s^4), \tag{3.3}$$

and for all $3 \leq r \leq n$

$$\mathbb{P}\left[[i,j] \in \mathcal{P}_{r,\tau_{\text{cip}}}\right] \leq (n_1 n_2)^r p^{2r} s^{2r} \left(\frac{wn}{n_1 n_2}\right)^r \left[1 + O\left(\frac{1}{np}\right)\right] = O(n^r w^r p^{2r} s^{2r}). \tag{3.4}$$

Note that for $r > 3$, we have

$$\frac{n^r w^r p^{2r} s^{2r}}{n w^2 p^4 s^4} = n^{r-1} w^{r-2} p^{2(r-2)} s^{2(r-2)} = o(1). \tag{3.5}$$

From (3.5), we conclude that for $r > 3$

$$\mathbb{P}\big[[i,j] \in \mathcal{P}_{r,\tau_{\text{cip}}}\big] = O(n w^2 p^4 s^4).$$

Therefore, by using union bound the probability that a couple $[i,j]$ obtains two marks is bounded from above by

$$\begin{aligned}\mathbb{P}\big[[i,j] \in \mathcal{P}_{\geq 2,\tau_{\text{cip}}}\big] &\leq \sum_{r=2}^{n} \mathbb{P}\big[[i,j] \in \mathcal{P}_{r,\tau_{\text{cip}}}\big] \\ &= \mathbb{P}\big[[i,j] \in \mathcal{P}_{2,\tau_{\text{cip}}}\big] + \mathbb{P}\big[[i,j] \in \mathcal{P}_{3,\tau_{\text{cip}}}\big] + \sum_{r=4}^{n} \mathbb{P}\big[[i,j] \in \mathcal{P}_{r,\tau_{\text{cip}}}\big] \\ &\leq 2O(n^2 w^2 p^4 s^4) + nO(n w^2 p^4 s^4) = O(n^2 w^2 p^4 s^4),\end{aligned}$$

consequently

$$\mathbb{E}\left[\Psi(\mathcal{M}_{\tau_{\text{cip}}})\right] \leq n_1 n_2 \mathbb{P}\big[[i,j] \in \mathcal{P}_{\geq 2,\tau_{\text{cip}}}\big] = O(w^2 n^4 p^4 s^4 t^2).$$

This proves Lemma 17.

We will prove Equation (3.3); Equation (3.4) is proven in a similar way. Consider a couple $[i,j]$, $i \neq j$. This couple obtains two marks if there exist two other couples $[u_1, v_1]$ and $[u_2, v_2]$ in the seed set $\mathcal{A}_0$ such that $(i,u_1),(i,u_2) \in E_1$ and $(j,v_1),(j,v_2) \in E_2$. Note that as a couple $[i,j]$ is added to the matching $\mathcal{M}$, any other couple in the form of $[i,j']$ or $[i',j]$ cannot also be in the matching $\mathcal{M}$ and will be discarded to ensure that each node is matched at most once. Hence, for the sake of analysis, we assume all the marks that were previously created from all the couples that have the form of $[i,j']$ or $[i',j]$ are subtracted.[6]

Let us first assume that $i \notin \{v_1, v_2\}$ and $j \notin \{u_1, u_2\}$. We consider three cases.

(i) All the four nodes u_1, u_2, v_1 and v_2 are different: in this case, the edges $(i,u_1),(i,u_2) \in E_1$ and $(j,v_1),(j,v_2) \in E_2$ exist independently. Thus $[i,j]$ obtains two marks from these two couples with probability $p^4 s^4$. The number of such couples $[u_1,v_1],[u_2,v_2]$ is at most $(n_1 n_2)^2$, and each such couple is in the seed set with probability $\frac{wn}{n_1 n_2}$. Thus the probability that $[i,j]$ obtains two marks from such couples is bounded from above by $(n_1 n_2)^2 (\frac{wn}{n_1 n_2})^2 p^4 s^4 = O(w^2 n^2 p^4 s^4)$.

(ii) We assume $u_1 \neq v_1$ and $u_2 \neq v_2$. We further assume that either $u_1 = u_2$ or $v_1 = v_2$ (but not both): let us, with out loss of generality, take $u_1 = u_2$ and $v_1 \neq v_2$. In this case, the edge $(i,u_1) = (i,u_2) \in E_1$ exists with probability ps, and both edges $(j,v_1),(j,v_2) \in E_2$

---

[6] We observe that in practice, this step only has a small effect on the performance, but is computationally costly.

exist with probability p^2s^2. Thus, the couple $[i,j]$ with probability p^3s^3 obtains two marks from these two couples. The number of such couples $[u_1,v_1]$ and $[u_1,v_2]$ is at most $O(n_0n_1n_2)$, therefore, the probability that $[i,j]$ obtains two marks from such kind of couples is upper-bounded by $n_0n_1n_2(\frac{wn}{n_1n_2})^2p^3s^3 = O(w^2np^3s^3)$.

(iii) Either $u_1 = v_1$, $u_2 = v_2$ or both: along the same lines as above, it is easy to see the probability that $[i,j]$ obtains two marks is upper bounded by $O(w^2np^3s^3)$.

Now, assume $i \in \{v_1,v_2\}$ or $j \in \{u_1,u_2\}$; similarly to the method we used above, we upper-bound the probability that a couple $[i,j]$ obtains two marks from the couples $[u_1,v_1]$ and $[u_2,v_2]$. There are three different cases:

(i) One node in $\{u_1,v_1,u_2,v_2\}$ is equal to i or j: The number of such couples $[u_1,v_1],[u_2,v_2]$ is at most $O(n^3t^3)$. Couple $[i,j]$ obtains two marks from these couples with probability p^4s^4.

(ii) Two nodes in $\{u_1,v_1,u_2,v_2\}$ are equal to i or j: There are at most $O(n^2t^2)$ such couples, and the probability that $[i,j]$ obtains two marks from these couples is $O(p^3s^3)$.

(iii) Three nodes in $\{u_1,v_1,u_2,v_2\}$ are equal to i or j: For the couples $[u_1,v_1]$,$[u_2,v_2]$, there are at most $O(nt)$ choices. The couple $[i,j]$ obtains two marks from such kind of couples with probability $O(p^2s^2)$.

To summarize, by considering all the cases mentioned above, the probability that a couple $[i,j]$ obtains two marks at time τ_{cip} is bounded from above by $O((n_1n_2)^2p^4s^4(\frac{wn}{n_1n_2})^2[1+\frac{1}{np}]) = O(n^2w^2p^4s^4)$. This proves (3.3). □

The next step is to prove that the number of incorrectly matched couples at each time step $1 \le \tau \le T^*$ of the matching process is at most $O(w^2n^4p^4s^4) = o(a_c)$. At each time step $\tau \ge 1$, `NoisySeeds` picks a random couple $[i,j] \in \mathcal{M}_\tau \setminus \mathcal{Z}_\tau$ and adds one mark to its neighbouring couples. It is easy to see that $\Lambda(\mathcal{M}_\tau)$ and $\Psi(\mathcal{M}_\tau)$ are increasing by τ. In Lemma 19 stated below, using Markov's inequality and the results of Lemmas 17 and 18, we will prove that $\Psi(\mathcal{M}^*) = \Psi(\mathcal{M}_{T^*}) = o(a_c)$. Consequently, from monotonicity of $\Psi(\mathcal{M}_\tau)$ with respect to τ, we conclude that $\Psi(\mathcal{M}_\tau) = o(a_c)$ holds for all $1 \le \tau \le T^*$.

Lemma 18. $\mathbb{E}[\Psi(\mathcal{M}_{T^*})] = O(w^2n^4p^4s^4t^2) = o(a_c)$.

Proof. We define the random variables $X_{i,j}, i \ne j$, as

$$X_{i,j} = \begin{cases} 1 & \text{if } [i,j] \in \mathcal{P}_{\ge 2,T^*} \\ 0 & \text{o.w.,} \end{cases}$$

and let $X = \sum_{\forall [i,j], i\ne j} X_{i,j}$. In words, the random variable $X_{i,j}$ indicates whether an individual couple $[i,j]$ can collect at least two marks during the steps of `NoisySeeds`. Of course as each node can be matched at most once, not all couples in $\mathcal{P}_{\geq 2,T^*}$ will end up in the final match set. Hence we have

$$\mathbb{E}\left[\Psi(\mathcal{M}_{T^*})\right] \leq \sum_{i \neq j} \mathbb{E}\left[X_{i,j}\right] \leq n_1 n_2 \mathbb{P}\left[[i,j] \in \mathcal{P}_{\geq 2,T^*}\right]. \tag{3.6}$$

We proceed by finding an upper bound for $\mathbb{P}\left[[i,j] \in \mathcal{P}_{\geq 2,T^*}\right]$. Let $\mathcal{P}_{q,\mathcal{M}^*}$ and $\mathcal{P}_{\geq q,\mathcal{M}^*}$, respectively, represent the set of couples that obtain exactly q and at least q marks from all the T^* matched couples $\mathcal{M}^* = \mathcal{M}_{T^*}$. Assuming $i \neq j$, the couple $[i,j]$ is in the set $\mathcal{P}_{\geq 2,T^*}$ if one of the three following cases holds (we thus can use the union bound for $\mathbb{P}\left[[i,j] \in \mathcal{P}_{\geq 2,T^*}\right]$):

Case 1 $[i,j] \in \mathcal{P}_{\geq 2,\tau_{\text{cip}}}$, i.e., $[i,j]$ obtains at least two marks from couples in $\mathcal{A}_0$. This means that the couple $[i,j]$ is added to the set of matched couples already at time step τ_{cip}. Indeed, for the result of Lemma 17, we have

$$\mathbb{P}\left[[i,j] \in \mathcal{P}_{\geq 2,\tau_{\text{cip}}}\right] = O(w^2 n^2 p^4 s^4).$$

Case 2 $[i,j] \in \mathcal{P}_{\geq 2,\mathcal{M}^*}$, i.e., $[i,j]$ obtains at least two marks from all the matched couples in $\mathcal{M}_{T^*}$ from time step $\tau = 1$ to $\tau = T^* \leq \min(n_1, n_2) = O(nt)$ (see Figure 3.4a). To upper bound this probability, we consider two cases: $[i,j] \in \mathcal{P}_{2,\mathcal{M}^*}$, and $[i,j] \in \mathcal{P}_{r,\mathcal{M}^*}$ for $2 < r \leq n$. Let us first find an upper bound for the former. Assume couple $[i,j]$ obtains two marks from the couples $[u_1, v_1]$ and $[u_2, v_2]$. As each node could be matched at most once, then i, u_1, and u_2 are mutually different. The same is true for j, v_1 and v_2. In this regard, there are three cases:

(i) Either $[u_1, v_1] = [j,i]$ or $[u_2, v_2] = [j,i]$. It is obvious that both cases cannot hold simultaneously. We assume without loss of generality that $[u_1, v_1] = [j,i]$, and thus $u_2 \neq j$ and $v_2 \neq i$. In this case, each one of the edges $(i,j), (i,u_2), (j,v_2)$ is sampled in the underlying graph G independently with probability p. If an edge exists in the hidden underlying graph G, then it appears with probability s in each one of the sampled graphs G_1 or G_2. Therefore, couple $[i,j]$ obtains two marks with probability $p^3 s^4$. As the couple $[u_1, v_1] = [j,i]$ is fixed, for couple $[u_2, v_2]$ we have at most $T^* - 1 = O(nt)$ choices.

(ii) Either $i = v_1$ and $j = u_2$, or $i = v_2$ and $j = u_1$(but not both): we assume without loss of generality that the former case holds. Similarly to the case (i), couple $[i,j]$ obtains two marks with probability $p^3 s^4$. As nodes i and j can be matched only once, the number of this type of couples is at most one.

(iii) None of the cases above hold, i.e., $i \notin \{u_1, u_2\}$ and $j \notin \{v_1, v_2\}$: all the four edges $(i,u_1), (i,u_2) \in E_1$ and $(j,v_1), (j,v_2) \in E_2$ are independent, and they exist simultaneously with probability $p^4 s^4$. The number of such $[u_1, v_1]$ and $[u_2, v_2]$ couples is at most $\binom{T^*}{2}$, where $T^* = O(nt)$. To sum up all the above three cases we have

$$\mathbb{P}[[i,j] \in \mathcal{P}_{2,\mathcal{M}^*}] \leq O(np^3s^4t) + O(p^3s^4) + O(n^2p^4s^4t^2) = O(n^2p^4s^4t^2).$$

A couple $[i,j]$ is added to the set of matched couples if it obtains $r \geq 2$ marks. So far, we have found an upper bound for the probability of obtaining exactly two marks from all the matched couples $\mathcal{M}_{T^*}$. We proceed by finding an upper bound for cases $2 < r \leq n$. Let $\mathcal{C}_{r,\mathcal{M}^*}$ be the set of all the possible ways of choosing r couples from the set $\mathcal{M}^* = \mathcal{M}_{T^*}$. We also represent a generic element of $\mathcal{C}_{r,\mathcal{M}^*}$ by c_r. To upper bound the probability that a couple $[i,j]$ obtains exactly r marks from all the possible combinations c_r, we consider two cases: (a) For a given c_r, $i \in V_1(c_r)$ and $j \in V_2(c_r)$, couple $[i,j]$ obtains r marks from c_r with probability $p^{2r-1}s^{2r}$. We know that nodes i and j appear at most once in the sets $V_1(c_r)$ and $V_2(c_r)$, respectively. Hence, the number of possible c_r in this case is at most $O((nt)^{r-1})$. (b) If case (a) does not hold, then couple $[i,j]$ with probability $p^{2r}s^{2r}$ obtains r marks from each one of at most $O((nt)^r)$ possible combinations for c_r. To conclude, we obtain

$$\mathbb{P}[[i,j] \in \mathcal{P}_{r,\mathcal{M}^*}] \leq O((nt)^{r-1}p^{2r-1}s^{2r}) + O((nt)^r p^{2r}s^{2r}) = O(n^r p^{2r}s^{2r}t^r).$$

Note that for $r > 3$, we have

$$\frac{n^r p^{2r} s^{2r} t^r}{ntp^4s^4} = n^{r-1}p^{2(r-2)}s^{2(r-2)}t^{r-1} = o(1). \tag{3.7}$$

From (3.7), we conclude that for $r > 3$

$$\mathbb{P}[[i,j] \in \mathcal{P}_{r,\mathcal{M}^*}] = O(ntp^4s^4).$$

By using a union bound, the probability that a couple $[i,j]$ obtains two marks is bounded from above by

$$\mathbb{P}\left[[i,j] \in \mathcal{P}_{\geq 2,\mathcal{M}^*}\right] \leq \sum_{r=2}^{n} \mathbb{P}\left[[i,j] \in \mathcal{P}_{r,\mathcal{M}^*}\right] \leq O(n^2p^4s^4t^2) + ntO(np^4s^4) = O(n^2p^4s^4t^2).$$

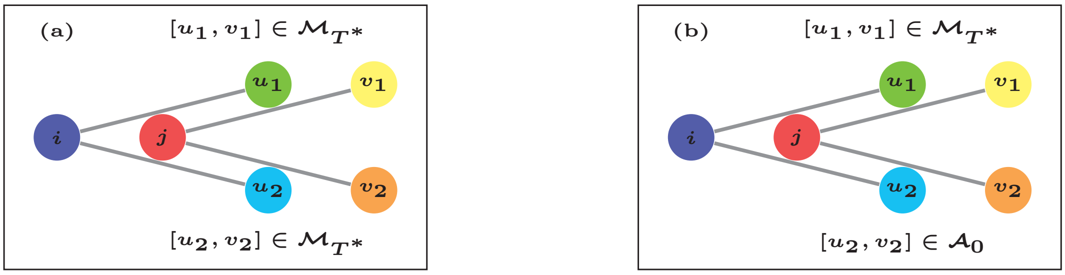

(a)
$[u_1, v_1] \in \mathcal{M}_{T^*}$
u_1
v_1
i
j
u_2
v_2
$[u_2, v_2] \in \mathcal{M}_{T^*}$
(b)
$[u_1, v_1] \in \mathcal{M}_{T^*}$
u_1
v_1
i
j
u_2
v_2
$[u_2, v_2] \in \mathcal{A}_0$

Figure 3.4 – (a) A couple $[i,j], i \neq j$ obtains two marks from couples $[u_1, v_1]$ and $[u_2, v_2] \in \mathcal{M}_{T^*}$. (b) A couple $[i,j], i \neq j$ obtains one mark from $[u_1, v_1] \in \mathcal{M}_{T^*}$ and one from $[u_2, v_2] \in \mathcal{A}_0$.

Case 3 $[i, j]$ obtains one mark from the couples in $\mathcal{A}_0$ and one mark from the matched couples $\mathcal{M}_{T^*}$, i.e., $[i, j] \in \mathcal{P}_{1,\tau_{\text{cip}}} \cap \mathcal{P}_{1,\mathcal{M}^*}$ (see Figure 3.4b). We know that $u_1 \notin \{i, u_2\}$ and $v_1 \notin \{j, v_2\}$. To upper-bound the probability of obtaining one mark from matched couples, there are two cases: (i) If $[u_1, v_1] = [j, i]$ then two edges $(i, j) \in E_1$ and $(j, i) \in E_2$ exist with probability ps^2. The number of these couples is at most one. (ii) If $[u_1, v_1] \neq [j, i]$ then two independent events $(i, u_1) \in E_1$ and $(j, v_1) \in E_2$ happen with probability p^2s^2. In this case, we have at most $T^* = O(nt)$ couples. Therefore, by using the union bound

$$\mathbb{P}\left[[i, j] \in \mathcal{P}_{1,\mathcal{M}^*}\right] \leq T^* p^2 s^2 + ps^2 = O(np^2 s^2 t).$$

Each couple $[u_2, v_2]$ is in the initial seed-set $\mathcal{A}_0$ with probability $\frac{wn}{n_1 n_2}$. To compute an upper-bound for $\mathbb{P}\big[[i, j] \in \mathcal{P}_{1,\tau_{\text{cip}}}\big]$, similarly as above, we consider two cases: (i) There is one couple such that $[u_2, v_2] = [j, i]$. Couple $[i, j]$ obtains one mark from this couple with probability ps^2. (ii) $[u_2, v_2] \neq [j, i]$, there are at most $O((nt)^2)$ possible candidates. Each one of these candidate couples is inside $\mathcal{A}_0$, with probability $\frac{wn}{n_1 n_2}$, and spreads out one mark to the couple $[i, j]$ with probability p^2s^2. To summarize,

$$\mathbb{P}\big[[i, j] \in \mathcal{P}_{1,\tau_{\text{cip}}}\big] \leq \frac{wn}{n_1 n_2} n_1 n_2 p^2 s^2 + ps^2 = O(wnp^2 s^2).$$

Now, we compute an upper bound for the joint probability $\mathbb{P}\big[[i, j] \in \mathcal{P}_{1,\mathcal{M}^*}, [i, j] \in \mathcal{P}_{1,\tau_{\text{cip}}}\big]$. We consider the two following cases: (i) If neither $i = v_1 = u_2$ or $j = u_1 = v_2$ then obtaining marks from the two sets are independent of each other. (ii) If either $i = v_1 = u_2$ or $j = u_1 = v_2$ (but not both) then a couple obtains two marks from these couples with probability p^3s^4. The maximum number of this kind of couples is at most $O(nt)$. Therefore, we have

$$\mathbb{P}\big[[i, j] \in \mathcal{P}_{1,\mathcal{M}^*}, [i, j] \in \mathcal{P}_{1,\tau_{\text{cip}}}\big] = O(wn^2 p^4 s^4) + O(np^3 s^4) = O(wn^2 p^4 s^4).$$

To wrap up all the three cases, we prove $\mathbb{P}\big[[i, j] \in \mathcal{P}_{\geq 2,T^*}\big] = O(wn^2 p^4 s^4)$.

Finally, as a consequence of (3.6), we have $\mathbb{E}\big[\Psi(\mathcal{M}_{T^*})\big] = O(w^2 n^4 p^4 s^4 t^2)$. This proves Lemma 18.

□

Lemma 19. *With high probability, we have* $\Psi(\mathcal{M}_\tau) = o(a_c)$ *and* $\Psi(\mathcal{M}_{\tau_{\text{cip}}}) = o(a_c)$.

Proof. For any $\delta > 0$, by using Markov's inequality, we have

$$\mathbb{P}\left[\frac{\Psi(\mathcal{M}_\tau)}{a_c} \geq \delta\right] \leq \frac{\mathbb{E}\left[\frac{\Psi(\mathcal{M}_\tau)}{a_c}\right]}{\delta}.$$

From Equation (5.14) and Lemma 18 we have $a_c = \frac{1}{2nt^2p^2s^4}$ and $\mathbb{E}\big[\Psi(\mathcal{M}_\tau)\big] \leq \mathbb{E}\big[\Psi(\mathcal{M}_{T^*})\big] =$

$O(wn^4p^4s^4t^2)$ for $\tau \le T^*$, respectively. Therefore,

$$\mathbb{P}\left[\frac{\Psi(\mathcal{M}_\tau)}{a_c} \ge \delta\right] \le \frac{\mathbb{E}\left[\frac{\Psi(\mathcal{M}_\tau)}{a_c}\right]}{\delta} = O(\frac{w^2n^5p^6s^8t^4}{\delta}) = O(n^{-\epsilon}) = o(1).$$

By using Lemma 17, we have $\mathbb{E}\left[\Psi(\mathcal{M}_{\tau_{\text{cip}}})\right] = O(w^2n^4p^4s^4)$. If we use Markov's inequality again and follow the same steps as above, we obtain that w.h.p. $\Psi(\mathcal{M}_{\tau_{\text{cip}}}) = o(a_c)$. These prove Lemma 19. □

In the next step, we use Lemma 19 to prove Theorem 16. We first give a brief overview of bootstrap percolation [86]. Bootstrap percolation is the process of node activation on a $G(n,p)$ random graph [86]. In this process, initially we are given a set $\mathcal{A}(0)$ ($|\mathcal{A}(0)| = a_0$) of active nodes and a threshold $r \ge 2$. A node is activated at time step τ if at least r of its neighbours were activated and used in the previous τ time steps. Let $\mathcal{A}(\tau)$ and $\mathcal{Z}(\tau)$ denote the set of active and used nodes at time step τ. We assume $\mathcal{Z}(0) = \emptyset$. At each time step $\tau \ge 1$, we choose a node u_τ from $\mathcal{A}(\tau-1) \setminus \mathcal{Z}(\tau-1)$ and give each one of its neighbours a mark. We call u_τ a used node and update $\mathcal{Z}(\tau) = \mathcal{Z}(\tau-1) \cup u_\tau$. Assume $\Delta\mathcal{A}(\tau)$ is the set of activated nodes at time step τ and we let $\mathcal{A}(\tau) = \mathcal{A}(\tau-1) \cup \Delta\mathcal{A}(\tau)$. At each step τ (before the activation process stops) one node is added to the set of used nodes, i.e., $|\mathcal{Z}(\tau)| = \tau$. We define $A_{n,a}(\tau) = |\mathcal{A}(\tau)|$. Also, $T^*_{n,a}$ denote the time step when $A_{n,a}(T^*_{n,a}) = |\mathcal{Z}(T^*_{n,a})| = T^*_{n,a}$. The bootstrap percolation process stops when $\mathcal{A}(\tau) \setminus \mathcal{Z}(\tau) = \emptyset$ or equivalently $A_{n,a}(\tau) \le \tau$. The phase transition threshold for bootstrap percolation is stated in the following theorem.

Theorem 20 (Theorem 3.1 and Lemma 8.2 of [86]). *Assume* $b_{c,r} = n\frac{(pn)^{r-1}}{(r-1)!}e^{-pn}$, $\tau_{c,r} = \left[\frac{(r-1)!}{n(ps^2)^r}\right]^{\frac{1}{r-1}}$ *and* $a_{c,r} = (1-\frac{1}{r})\tau_{c,r}$. *And let* $b^* = b_{c,r}w(n)$, *where* $\omega(n) \to \infty$ *slowly, otherwise it is arbitrary. Suppose that* $r \ge 2$ *and* $n^{-1} \ll p \ll n^{-1/r}$. *Then, for any* $a > a_{c,r}$, *w.h.p.* $A_{n,a}(\tau) > \tau$ *for all* $\tau \in [0, n-b^*]$.

Hence Theorem 20 is valid for any choice of $\omega(n) \to \infty$; it is equivalent to the statement that for all $\tau \in [0, n-O(b_{c,r})]$ with high probability $A_{n,a}(\tau) > \tau$ [86]. It is easy to see that $O(b_{c,r}) = o(n)$ [86]. By analogy between graph matching problems over $G(n,p;t,s)$ graphs and the bootstrap percolation process on $G(n_0, ps^2)$ [202], for time steps $\tau \ge 1$ we have

$$\begin{aligned}\mathbb{P}\left[\Lambda(\mathcal{M}_\tau) > \tau\right] &\overset{(a)}{\ge} \mathbb{P}\left[A_{n_0-2\Psi(\mathcal{M}_{T^*}),a_0}\left(\tau - 3\Psi(\mathcal{M}_{T^*}) + a_0\right) > \tau + a_0\right]\\ &\overset{(b)}{\ge} \mathbb{P}\left[A_{n_0-3\Psi(\mathcal{M}_{T^*}),a_0}\left(\tau - 3\Psi(\mathcal{M}_{T^*}) + a_0\right) > \tau + a_0\right]\\ &\overset{(c)}{\ge} \mathbb{P}\left[A_{n_0-6\Psi(\mathcal{M}_{T^*}),a_0-3\Psi(\mathcal{M}_{T^*})}\left(\tau - 3\Psi(\mathcal{M}_{T^*}) + a_0\right) > \tau - 3\Psi(\mathcal{M}_{T^*}) + a_0\right]\end{aligned} \tag{3.8}$$

The three inequalities follow from the following reasons:

(a) In the matching process, we have a_0 initial correct couples. At any time step τ, we have $\Psi(\mathcal{M}_\tau) \leq \Psi(\mathcal{M}_{T^*}) = o(a_c)$ incorrectly matched couples. Each incorrect couple $[i,j], i \neq j$, in the worst case, removes marks produced by the two correct couples $[i,i]$ and $[j,j]$, from the set of used couples $\mathcal{Z}_\tau$. We know that among the matched couples, there are at most $o(a_c)$ incorrect couples. Therefore, we conclude that there are at least $n_0 - 2\Psi(\mathcal{M}_{T^*})$ potential correct couples that obtain marks from at least $\tau - 3\Psi(\mathcal{M}_{T^*}) + a_0$ correct couples at time step τ.

(b) As we assume p and the number of initial active nodes are fixed, decreasing the total number of nodes by $\Psi(\mathcal{M}_{T^*})$ would increase the probability of the process stopping.

(c) We assume, in the worst case, at the first $3\Psi(\mathcal{M}_{T^*})$ steps of the bootstrap percolation, the chosen nodes from $\mathcal{A}(\tau-1) \setminus \mathcal{Z}(\tau-1)$ do not spread out marks. Thus the probability that the percolation process stops would increase.

Note that $\Lambda(\mathcal{A}_0) = Binomial(n_0, \frac{c}{n_0})$ and $c > a_c \to \infty$, therefore, by using the Chernoff bound, we can conclude that for an arbitrarily small but fixed $\epsilon' > 0$ with high probability $a_0 = \Lambda(\mathcal{A}_0) > (1-\epsilon')\mathbb{E}[\Lambda(\mathcal{A}_0)] = (1-\epsilon')c$. Finally, if $a_0 = \Lambda(\mathcal{A}_0) > a_c - 3\Psi(\mathcal{M}_{T^*})$, then from (3.8) and Theorem 20 we conclude that w.h.p. $T^*_{n_0-6\Psi(\mathcal{M}_{T^*}),a_0} = n_0 - o(n)$. Also, (3.8) implies that w.h.p. $T^* \geq T^*_{n_0-6\Psi(\mathcal{M}_{T^*}),a_0}$. From Lemma 19, we know that at the time $T^* \leq \min(n_1, n_2)$ the number of incorrectly matched couples is upper-bounded by $\Psi(\mathcal{A}_{T^*}) = o(a_c)$, and $\Psi(\mathcal{A}_{T^*}) + \Lambda(\mathcal{A}_{T^*}) = T^*$. Thus, w.h.p. $\Lambda(\mathcal{M}_{T^*}) = n_0 - o(n)$ and $\Psi(\mathcal{A}_{T^*}) = o(a_c)$. Note that by using the Chernoff bound w.h.p. we obtain $n_0 = nt^2 \pm o(n)$. This proves Theorem 16.

3.3 ExpandWhenStuck Heuristic

In this section, we introduce a new heuristic algorithm, called `ExpandWhenStuck`; it is designed based on the robustness ideas developed in the previous sections. This algorithm is able to match real social-networks with over a million nodes by using a small number of seeds (e.g., see Figure 3.16 and Table 3.2 in Section 3.4). In comparison with `ExpandOnce`, this algorithm has better performance for both real and random graphs, and its computational complexity is lower. However, we cannot formally characterize its performance. To better illustrate, let us briefly go back to the `PercolateMatched` algorithm described in the beginning of Section 3.1. In the sub-critical regime of `PercolateMatched`, the final number of matched couples is at most twice the number of initial seeds [86]. The robustness arguments of Section 3.2 allow PGM algorithms to be much more aggressive in spreading out marks.

A main feature of `ExpandWhenStuck` is to expand the seed set by many noisy candidate couples whenever there are no other unused matched couples. More precisely, whenever there are no further couples with a score of at least two, we add all the unused[7] and unmatched[8] neighbouring couples of all the matched couples to the candidate couples (line 11 in Algorithm 3) and consequently new marks are spread out. Among these candidate couples, where a small

---

[7] A couple $[i,j]$ is unused if $[i,j] \notin \mathcal{Z}$.

[8] A couple $[i,j]$ is unmatched if $i \notin V_1(\mathcal{M})$ and $j \notin V_2(\mathcal{M})$.

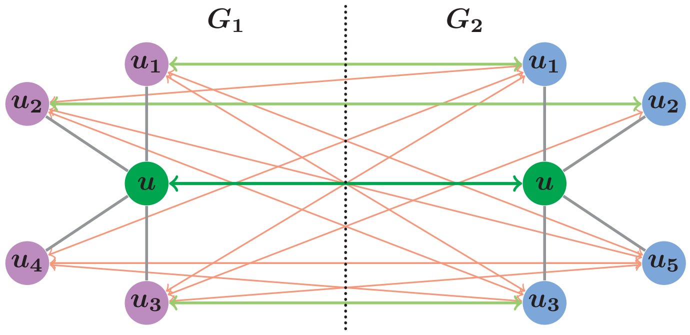


Figure 3.5 – `ExpandWhenStuck` (Algorithm 3): Nodes u_1, u_2, u_3, u_4 and u_5 are unmatched neighbours of node u in the underlying graph G (see Example 21 and Figure 3.2).

fraction is correct and most of them are incorrect, (i) correct couples help us to continue the percolation process and to match remaining unmatched couples, and (ii) incorrect couples have a negligible effect (see Theorem 16).

Example 21. When the percolation graph-matching process is stopped, there is still useful information that can help us match the remaining nodes. Assume, as in Figure 3.2, there are no unmatched couples with a score of at least $r = 2$. Node u is one of these (correctly) matched nodes. Among all the 16 possible unmatched neighbouring-couples of the couple $[u, u]$ (see Figure 3.5), three of them are correct (light-green arrows) and the rest are incorrect (light-red arrows). Therefore, `ExpandWhenStuck` adds them to the set of candidate couples. As our algorithm is robust to the incorrect candidate couples, correct candidate couples can help us in the matching process.

In addition, to enhance the performance of our algorithm (especially for real graphs), we make the following further modification. At each time step, instead of adding all the candidate couples with a score of at least two to the matched set, we choose the one with the highest score among such couples and add it to the matched set; also, each node is matched at most once. We then proceed with spreading out the marks from this matched couple.

In many steps (especially in the beginning), there are several couples with the maximum score. Among all such candidate couples $[i, j]$, we choose the couple that minimizes the difference in the degrees of nodes $|d_{1,i} - d_{2,j}|$. This can be intuitively justified as $d_{1,i}$ is often closer to $d_{2,j}$ when $[i, j]$ is a correct couple, i.e., $i = j$, than when $i \neq j$. This degree tie-break increases the performance, especially in real graphs, because their degree distributions are often heavily skewed and less concentrated compared to the $G(n, p)$ model. For a schematic overview of the `ExpandWhenStuck` algorithm, refer to Figure 3.6. Algorithm 3 explains `ExpandWhenStuck` in detail.

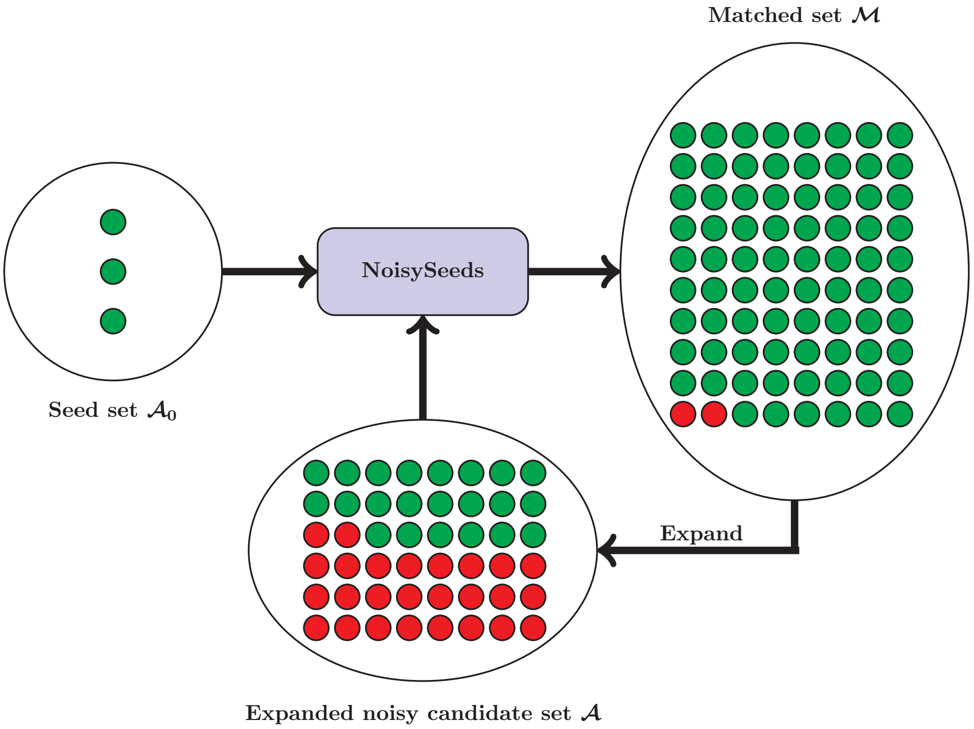

Matched set $\mathcal{M}$
NoisySeeds
Seed set $\mathcal{A}_0$
Expand
Expanded noisy candidate set $\mathcal{A}$

Figure 3.6 – The `ExpandWhenStuck` algorithm: It expands the candidate couples by many noisy couples whenever the percolation process is stuck (and not at the beginning). Because percolation graph matching algorithms are generally robust to the noisy candidate couples, the expansion step can help us in the matching process. In this figure, green and red circles represent correct and incorrect couples, respectively.

3.4 Simulation Results

In this section, we first demonstrate numerically the phase transitions of `NoisySeeds` given by Theorem 16. We next evaluate, through experiments, the performance of `ExpandOnce` and `ExpandWhenStuck` over the $G(n, p; t, s)$ model. We find that these algorithms match correct couples, and then stop at the right time. We show that these two algorithms are able to match graphs with only a handful of seeds. To compare the performance of our algorithm with the other methods in the literature, simulation results for `ExpandWhenStuck` over power-law and preferential attachment random graphs and real graphs are provided. Finally, we explain the MapReduce implementation of a variant of `ExpandWhenStuck`.

In this section, we use precision and recall to evaluate the performance of algorithms: (i) Precision refers to the fraction of errors in the set of matched nodes, and (ii) Recall is the fraction of nodes in the intersection of the two graphs $G_{1,2}$ that are matched correctly. Formally, they are defined as precision $= \frac{\Lambda(\mathcal{M}^*)}{\Lambda(\mathcal{M}^*)+\Psi(\mathcal{M}^*)}$ and recall $= \frac{\Lambda(\mathcal{M}^*)}{n_{\text{ident}}}$ where n_{ident} is the number of nodes that are present in both graphs $G_{1,2}$, with degrees of at least two (for other notations see Section 3.1.1 and Table 5.2 in Appendix 3.A).

Algorithm 3: `ExpandWhenStuck`

```
Input: $G_1(V_1, E_1), G_2(V_2, E_2)$, seed set $\mathcal{A}_0$ of correct couples
Output: The set of matched couples $\mathcal{M}$
$\mathcal{A} \leftarrow \mathcal{A}_0$ is the initial set of seed couples, $\mathcal{M} \leftarrow \mathcal{A}_0$;
$\mathcal{Z} \leftarrow \emptyset$ is the set of used couples;
while $|\mathcal{A}| > 0$ do
    for all couples $[i, j] \in \mathcal{A}$ do
        add the couple $[i, j]$ to $\mathcal{Z}$ and add one mark to all of its neighbouring couples;
    while there exists an unmatched couple with score at least 2 do
        among the couples with the highest score select the unmatched couple $[i, j]$ with the
          minimum $|d_{1,i} - d_{2,j}|$;
        add $[i, j]$ to the set $\mathcal{M}$;
        if $[i, j] \notin \mathcal{Z}$ then
            add one mark to all of its neighbouring couples and add the couple $[i, j]$ to $\mathcal{Z}$;
    $\mathcal{A} \leftarrow$ all neighbouring couples $[i, j]$ of matched couples $\mathcal{M}$ s.t. $[i, j] \notin \mathcal{Z}$, $i \notin V_1(\mathcal{M})$ and
      $j \notin V_2(\mathcal{M})$;
return $\mathcal{M}$;
```

3.4.1 Experimental Results with Random Graphs

The experiments in this part are performed over two different types of random graphs: (i) Erdős-Rényi graphs, and (ii) scale-free networks. Although the performance of our algorithm is guaranteed for the $G(n, p; t, s)$ model (see Theorem 16 in Section 3.2), simulation results show the excellent performance of our algorithm versus state-of-the-art graph-matching algorithms for all types of graphs studied in this chapter.

Erdős-Rényi Random Graphs

We first demonstrate numerically the phase transitions established in Theorem 16 for the `NoisySeeds` algorithm. As shown in Theorem 16, for the $G(n, p; t, s)$ model, such a transition takes place when the number of correct couples in the initial seed-set passes a certain threshold $a_{t,s,r}$, while there are possibly many incorrect couples in the seed set. In Figure 3.7, we plot the total number of correctly matched couples versus the normalized number of seeds (i.e., the number of correct seeds divided by $a_{t,s,r}$) for the set of parameters $n = 10^6, p = 20/n$ and different ranges of node and edge sampling probabilities. As can be seen, (i) the phase transitions take place close to the critical values of $a_{t,s,r}$ and (ii) the total number of correctly matched nodes is very close to the expected number of nodes in the intersection of the two vertex sets, i.e., nt^2. Note that in all the cases considered in Figure 3.7, the fraction of incorrectly matched couples is very small, i.e., the precision is close to one.

We now proceed with the simulation results of `ExpandOnce` and `ExpandWhenStuck`, to compare their performance over $G(n, p; t, s)$ model (we also compare with `PercolateMatched`

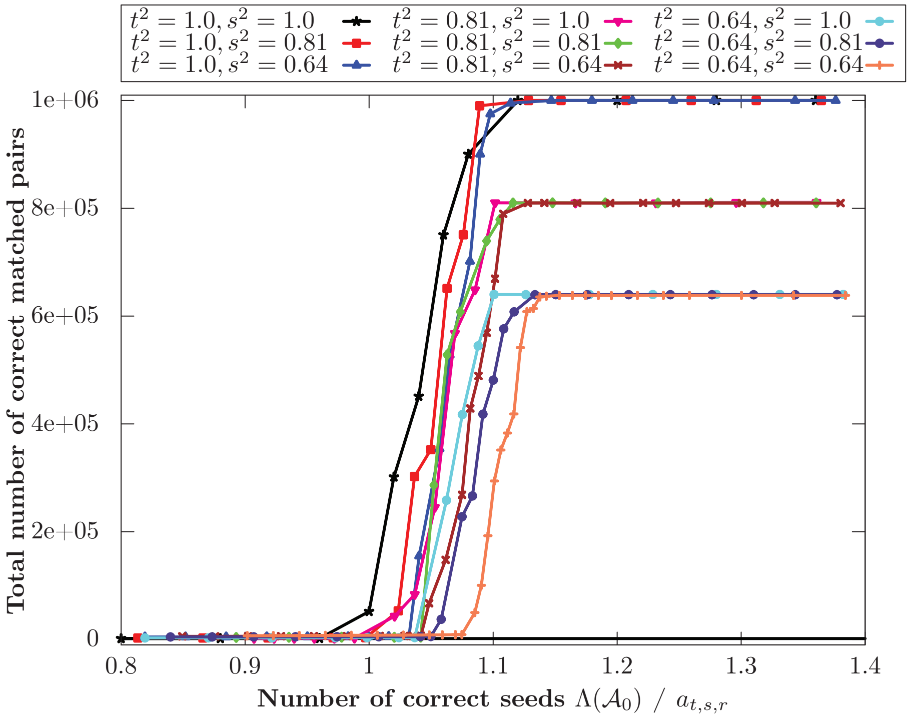


Figure 3.7 – `NoisySeeds` algorithm: Total number of correctly matched couples vs. number of seeds normalized by $\boldsymbol{a_{t,s,r}}$ for $r = 2$. Simulations are done over $\boldsymbol{G(n, 20/n; t, s)}$ with $\boldsymbol{n = 10^6}$.

[202]). Figures 3.8 confirms that the `ExpandOnce` algorithm performs surprisingly well and that with only a handful of seeds (only 13, 67 and 235 seeds for $s^2 = 0.81, 0.64$ and 0.49, respectively) it can correctly match almost all the nodes in a graph with $n = 10^6$ nodes and an average degree of 20. Figure 3.9 shows that when the matching process percolates, the precision is close to one. For comparison, if we set the minimum threshold $r = 2$, then the `PercolateMatched` algorithm [202] would need at least 1906, 3052 and 5207 seeds for matching $G(n, p; s)$ (equivalent to $t = 1$ in our model) graphs with edge overlap probabilities $s^2 = 0.81, 0.64$ and 0.49, respectively. Also, we observe that `ExpandWhenStuck` needs fewer seeds with respect to the `ExpandOnce` algorithm, in order to match correctly almost all the nodes. Specifically, the `ExpandWhenStuck` algorithm for parameters $t^2 = 1.0$ and $s^2 = 0.81$ with only 8 seeds (i.e., a fraction $8 \cdot 10^{-6}$ of the total number of nodes), matches almost all the nodes correctly, whereas for `PercolateMatched` this number is at least 1906 (the threshold for $r = 2$). In other words, in this example, `ExpandWhenStuck` needs 238 times fewer seeds than `PercolateMatched`.

We next analyze to what extent the robustness to the node overlap and edge overlap holds. In Figures 3.10 and 3.11, we observe a phase transition with the size and density of the graphs intersection. We see that for only 100 seeds, if the node overlap is at least 50%, our algorithm successfully identifies, with very high precision, almost all the nodes of the intersection.

Generally, it is true that when the algorithm is provided with enough seeds (i) the number of

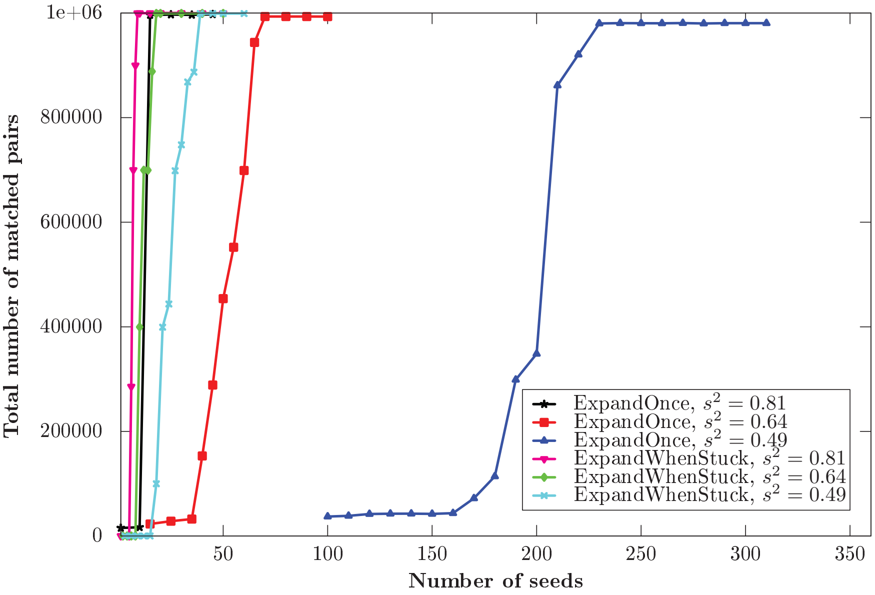


Figure 3.8 – Total number of matched couples vs. number of seeds. Simulations are done over $\boldsymbol{G(n, 20/n; t, s)}$ with $\boldsymbol{t = 1}$ and $\boldsymbol{n = 10^6}$.

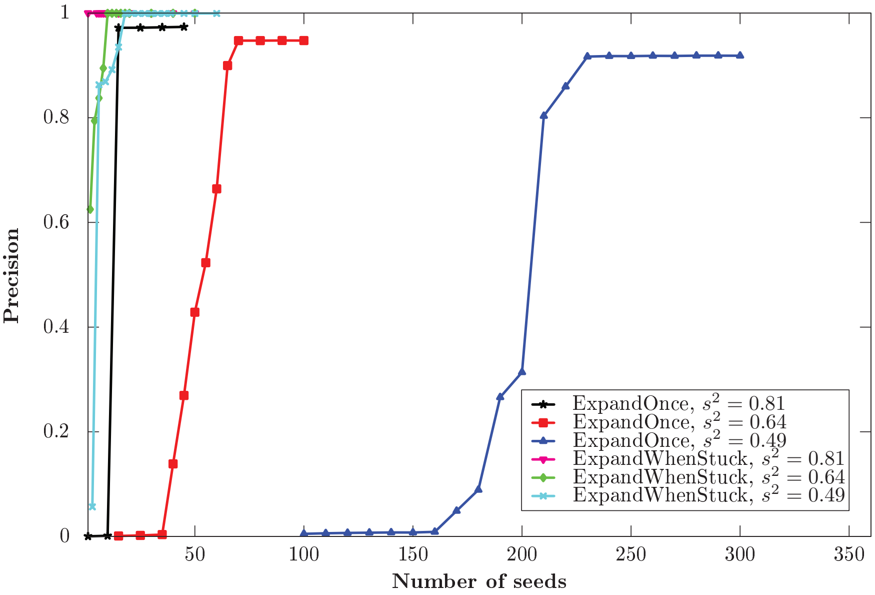


Figure 3.9 – Precision vs. number of seeds. Simulations are done over $\boldsymbol{G(n, 20/n; t, s)}$ with $\boldsymbol{t = 1}$ and $\boldsymbol{n = 10^6}$.

correctly matched nodes is very close to the expected number of nodes in the intersection of the two graphs (i.e., nt^2), (ii) fraction of incorrectly matched nodes is negligible, and (iii) there are phase transitions in the number of correct seeds. Also, we see that the graphs generated by $G(n, p; t, s)$ model are, indeed, matchable with a very few seeds. We observe that the PGM algorithm is robust to partial node-overlap.

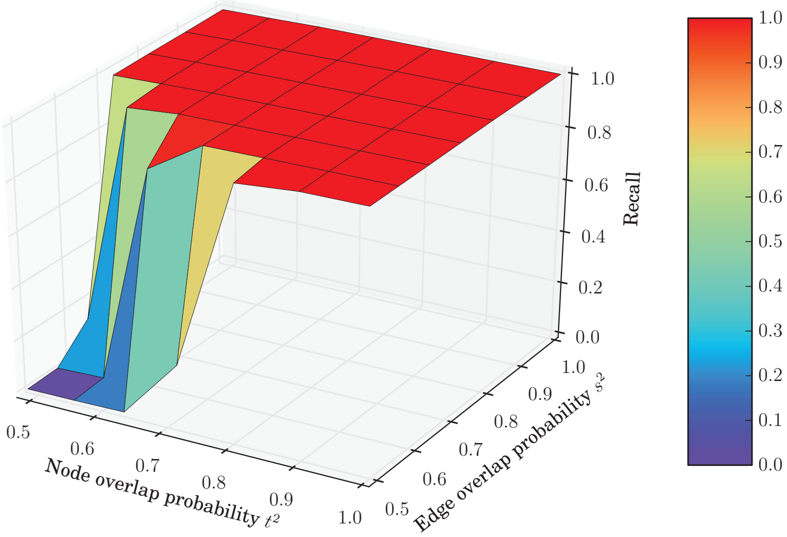


Figure 3.10 – Recall vs. node and edge overlap probabilities (i.e., t^2 and s^2). Number of seeds is 100. Simulations are done over $G(n, p; t, s)$ random graphs with $n = 10^5$ and an average degree of 20.

Scale-Free Random Graphs

We evaluate `ExpandWhenStuck` over scale-free random graphs; they are better representative of real-world (e.g., social and biological) networks. Note that as Erdős-Rényi graphs contain less structural information (for example, degree distribution is concentrated around the mean and a low clustering-coefficient), it is harder to match them. Also, our simulation results confirm that matching scale-free networks is an easier task.

First, we apply the `ExpandWhenStuck` algorithm to the Chung-Lu graphs [42] (a variant of power-law random graphs). In these graphs, the degree distribution of nodes follows a power-law distribution, i.e., the proportion of nodes of degree d scales with $d^{-\beta}$. In this model, the probability of having an edge between two nodes i and j with degrees d_i and d_j (this probability is independent of all the other edges in the graph) is proportional to $d_i d_j$. We generate two graphs $G_{1,2}$ through node-sampling with probability t and edge-sampling with probability s over a Chung-Lu graph. In Figure 3.12, for example, we observe that with only 20 seeds `ExpandWhenStuck` matches almost all the nodes correctly for fairly small node and edge overlap probabilities of 0.75. In all our experiments, we observe that precision is always better than recall.

Next, we apply `ExpandWhenStuck` to the preferential attachment random graphs. The Barabási-Albert model [21] is one of the models for social networks most referred to. This model generates random scale-free networks in a preferential attachment setting. A Barabási-Albert (BA) random graph is generated as follows [34]: (i) It starts with a single node with m self-loops;

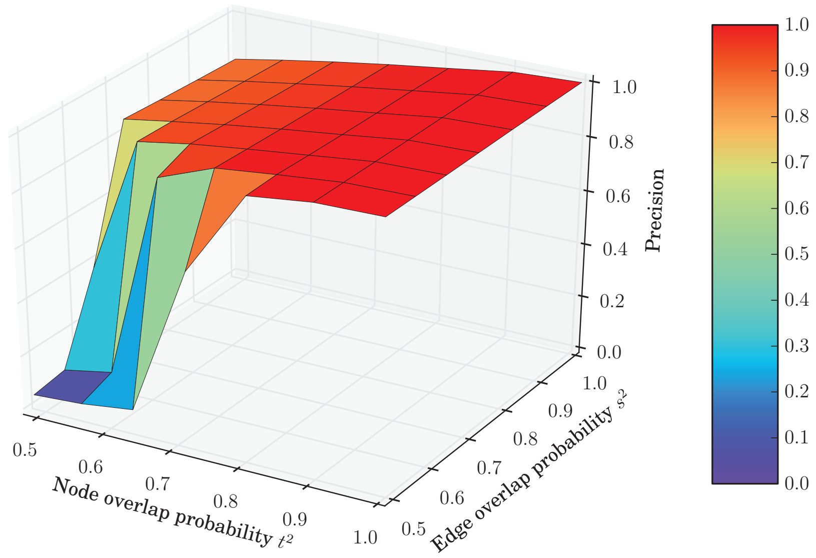


Figure 3.11 – Precision vs. node and edge overlap probabilities (i.e., t^2 and s^2). Number of seeds is 100. Simulations are done over $G(n, p; t, s)$ random graphs with $n = 10^5$ and an average degree of 20.

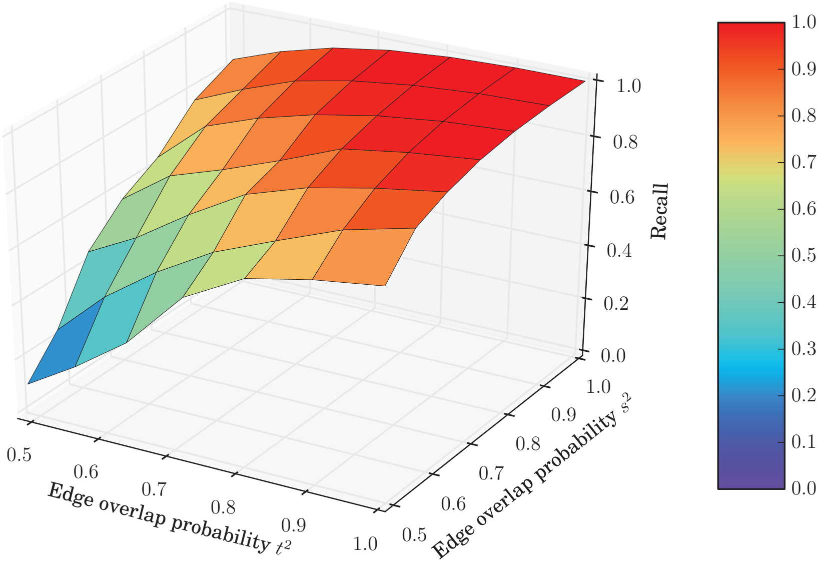


Figure 3.12 – Recall vs. node and edge overlap probabilities (i.e., t^2 and s^2). Number of seeds is 20. Simulations are done over power-law (Chung-Lu) random graphs with $n = 10^5$, $\beta = 2.5$ and an average degree of 20.

and (ii) each new node is connected to m existing nodes with probabilities proportional to their current degrees. Figure 3.13 shows the simulation result of `ExpandWhenStuck` over BA random graphs. In these experiments, the underlying graph G is sampled from the BA model. The two graphs G_1 and G_2 are generated by independent node-sampling and edge-sampling processes from graph G. Figure 3.14 shows the result over BA random graphs with $n = 10^5$ and an average degree of 20, where 10 uniformly chosen seeds are provided.

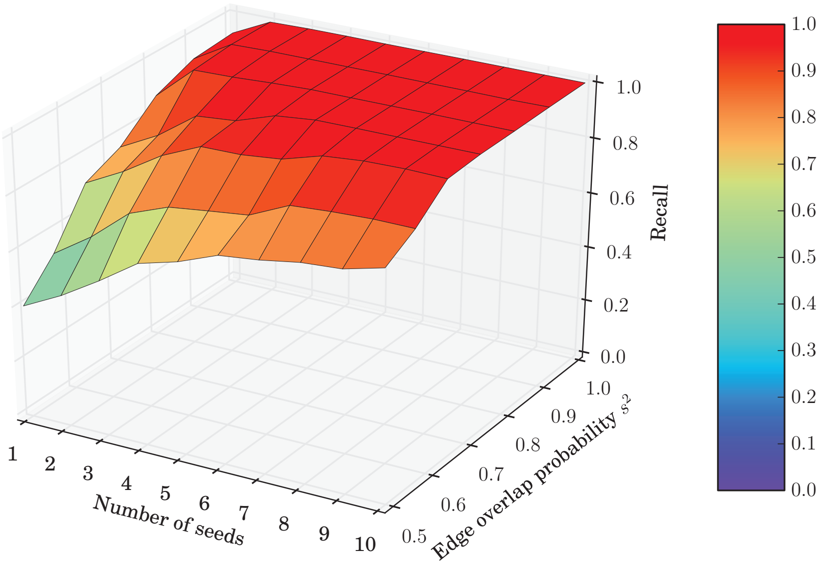


Figure 3.13 – `ExpandWhenStuck` algorithm: Recall vs. edge overlap probabilities (i.e., s^2) and number of seeds. Simulations are done over Barabási-Albert random graphs with edge overlap probability $t^2 = 0.81$, $n = 10^5$ and an average degree of 20.

We register the remarkable performance of the `ExpandWhenStuck` algorithm that, even with one seed, successfully aligns two graphs with 90% node and edge overlaps (see Figures 3.13 and 3.15). This brings up interesting questions about to success of the algorithm over preferential-attachment generated graphs. Korula and Lattanzi [109] analyzed a PGM algorithm in a regime of an extremely large number of seeds (in their result a constant fraction of nodes is needed as seeds), when they make the complete node-overlap assumption. Clearly, as shown in our experiments, very few seeds are needed and the full node-overlap assumption is not essential. We also mention that due to the heavy-tailed degree distribution, these graphs have several high-degree nodes that potentially can be used as seeds for the algorithm.

Our experiments show that choosing seeds among high-degree nodes, instead of picking them randomly, results in better alignments. For example, given only the highest-degree node as a seed is enough to match almost all the nodes correctly in Chung-Lu and BA graphs with $n = 10^6$, an average degree of 20, and sampling probabilities $t^2 = 0.81$ and $s^2 = 0.81$.

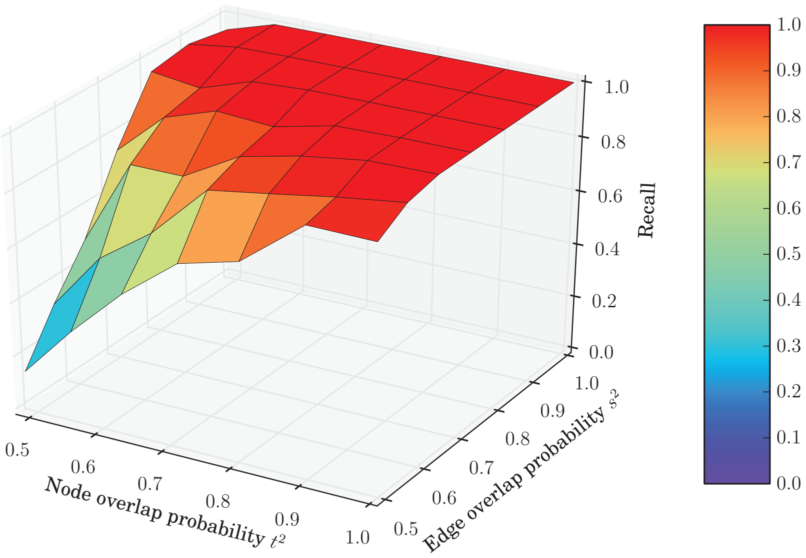


Figure 3.14 – Recall vs. node and edge overlap probabilities (i.e., t^2 and s^2). Number of seeds is 10. Simulations are done over Barabási-Albert random graphs with $n = 10^5$ and an average degree of 20.

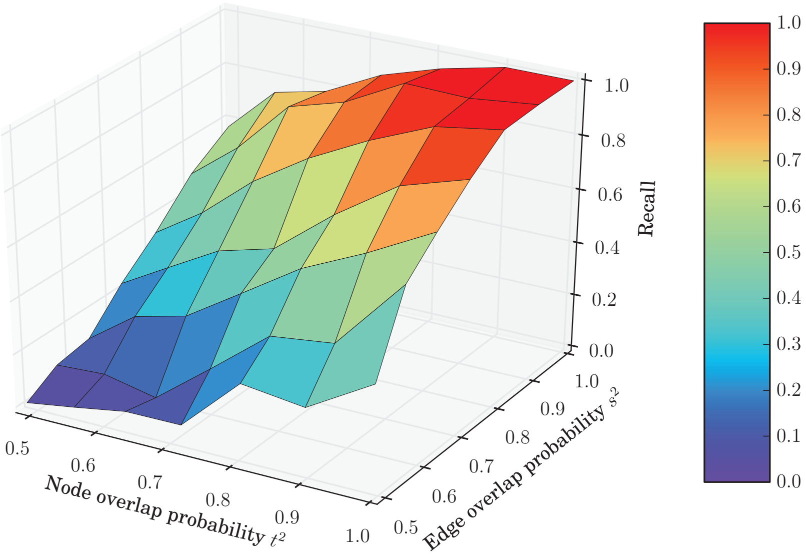


Figure 3.15 – Recall vs. node and edge overlap probabilities (i.e., t^2 and s^2). Number of seeds is 1. Simulations are done over Barabási-Albert random graphs with $n = 10^5$ and an average degree of 20.

3.4.2 Experimental Results with Real Graphs

In this section, we illustrate the experimental results of `ExpandWhenStuck` algorithm over five real social-networks. The baseline for our comparisons are state-of-the-art graph-matching algorithms: (i) `PercolateMatched` [202] (here, by `PercolateMatched` we mean a deferred version of it that has been reported to have better performance compared to its basic version), and (ii) `User-Matching` [109]. In all our experiments, `PercolateMatched` [202] outperforms `User-Matching` [109]. Therefore, due to space limitations we only plot the results corresponding to `PercolateMatched` algorithm in certain figures.

In statistical analysis and machine learning, F_1-score combines both the precision and the recall into one metric to provide an average of them [155]. This measure is defined as

$$F_1\text{-score} = 2\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \tag{3.9}$$

The value of F_1-score is between 0 and 1. When F_1-score is close to 1, we can conclude that (i) precision is close to 1, i.e., the fraction of errors in the set of matched couples is small, and (ii) recall is close to 1, i.e., a large fraction of nodes that are present in the both graphs $G_{1,2}$ are matched correctly. We use this measure to compare the performance of algorithms.

For the first experiment, we choose a very large real graph. We run `ExpandWhenStuck` over the Youtube graph with 1134890 nodes and an average degree of 5.26 [116] (see Table 3.1). In this graph, links correspond to friendships among users. The edge sampling with probability s generates two graphs $G_{1,2}$. To make a comparison with the `User-Matching` [109] and `PercolateMatched` [202] algorithms, we choose the node-sampling probability $t = 1.0$. Figure 3.16 compares our algorithm with the two baseline algorithms. The F_1-scores for `ExpandWhenStuck` is non-zero from the very beginning, and with only few seeds our algorithm finds high-quality alignments. We observe that the F_1-scores of `User-Matching` and `PercolateMatched` (for the sampling probabilities that we have considered here) are always around zero.

The second graph matching is done over friendship links on the Slashdot social network [116]. This network has 77360 nodes and an average degree of 12.13 (see Table 3.1). The two graphs $G_{1,2}$ are generated though node-sampling and edge-sampling processes over the Slashdot network. In Figure 3.17, when 20 seeds are provided, we observe the F_1-score (see (3.9)) for different sampling probabilities.

Table 3.1 – Statistics for Slashdot, Youtube and Pokec datasets.

	Slashdot	Youtube	Pokec
Nodes	77360	1134890	1632803
Edges	469180	2987623	22301964
Average degree	12.13	5.26	27.32
Average clustering coefficient	0.0555	0.0808	0.1094
Diameter (longest shortest path)	12	20	11

To analyze the correlation between the number of seeds, graph overlaps and recall, we run the following simulations. Figure 3.18, shows the recall for different number of seeds and

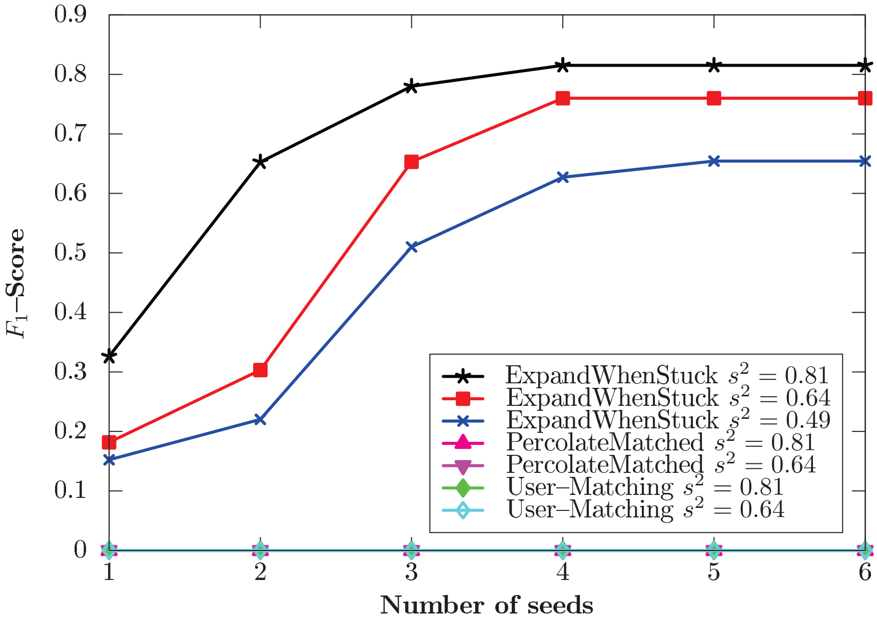


Figure 3.16 – F_1-score (see Equation (3.9)) vs. number of seeds. Simulations are done over Youtube graph with 1134890 node.

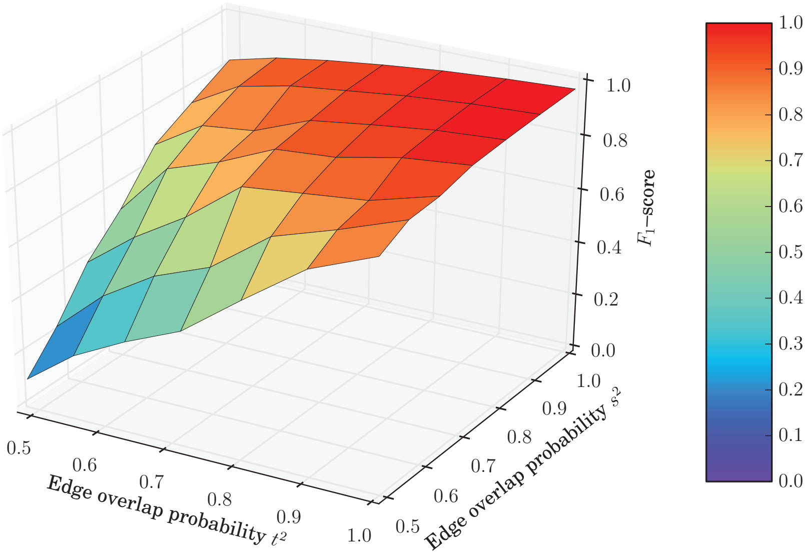


Figure 3.17 – F_1-score vs. node and edge overlap probabilities (i.e., t^2 and s^2). Simulations are done over Slashdot network when the number of seeds is 20.

node-overlap probabilities when the edge-overlap probability is $s^2 = 0.64$. Figure 3.19, shows the recall for different number of seeds and edge-overlap probabilities, and the node-overlap probability is $t^2 = 0.64$.

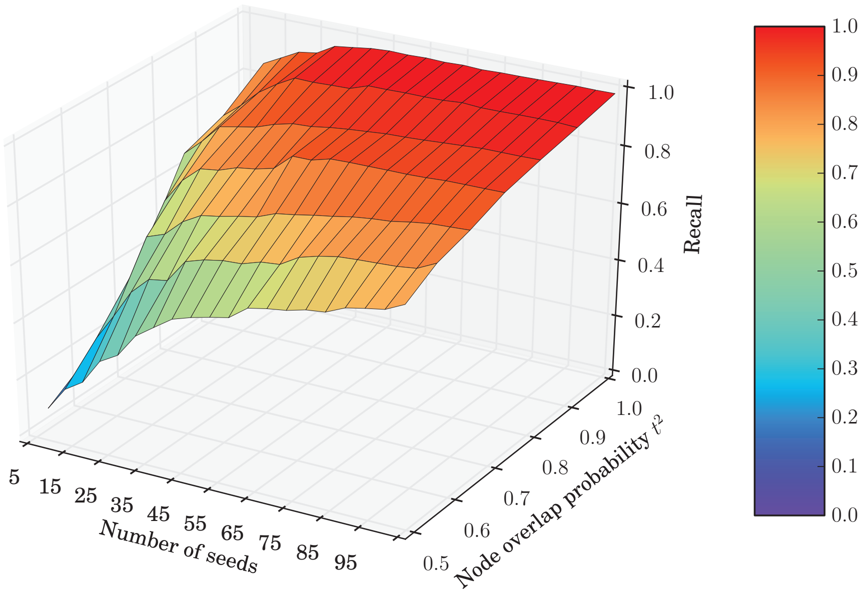


Figure 3.18 – Recall vs. number of seeds and node-overlap probabilities (i.e., t^2). Simulations are done over Slashdot network when the edge overlap probability is $s^2 = 0.64$.

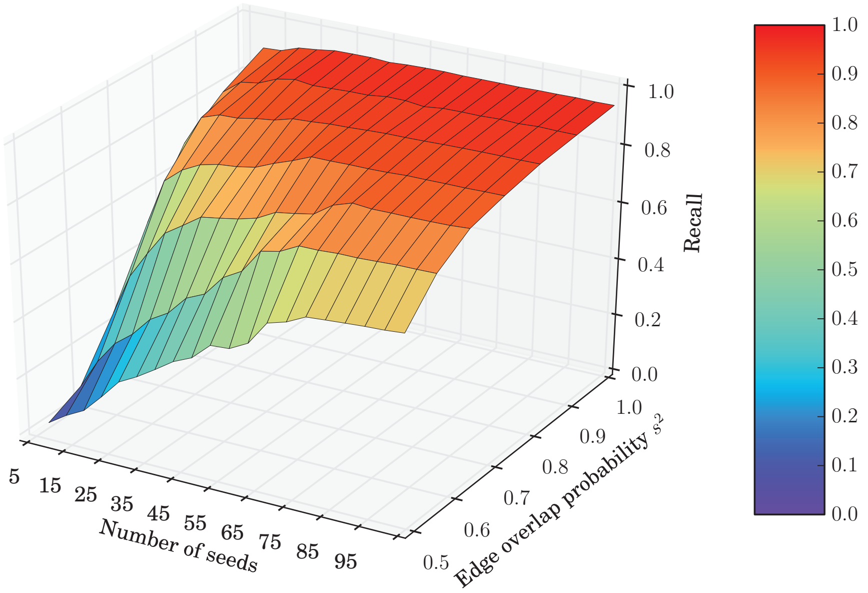


Figure 3.19 – Recall vs. number of seeds and edge overlap probabilities (i.e., s^2). Simulations are done over Slashdot network when the node overlap probability is $t^2 = 0.64$.

For the third experiment, we use the most popular online social network in Slovakia called Pokec, with 1632803 nodes and an average degree of 27.32 [116] (see Table 3.1). Again, the two graphs $G_{1,2}$ are generated though node and edge samplings. The excellent performance of `ExpandWhenStuck` over Pokec social-network is shown in Table 3.2.

Table 3.2 – $\boldsymbol{F_1}$-score (see Equation (3.9)): Simulations are done over Pokec social network with $n = 1632803$ and $t^2 = 1.0$, when 5 seeds are provided.

Algorithm \ s^2	0.81	0.64	0.49
`ExpandWhenStuck`	0.99	0.98	0.97
`User-Matching` [109]	0.04	0.02	≈ 0
`PercolateMatched` [202]	0.05	0.02	≈ 0

In the fourth experiment, we use different snapshots of the e-mail network on the EPFL campus [150]. Each snapshot of the network is created by aggregating all the exchanged e-mails in a given time period. Each node corresponds to an account, and the undirected edges represent exchanged e-mails between entities. In this experiment, we match two real graphs without any modelling assumptions, i.e., we do not assume any node or edge sampling process. As shown in Figure 3.20, with only one seed we can match most of the nodes in the EPFL e-mail network. In all snapshots of the EPFL e-mail network, the nodes with the highest degrees are the same. This is because the node-degree distributions of real graphs are often heavy-tailed. We can use the couple of the highest degree nodes as the starting seed to `ExpandWhenStuck`. For this network, the performance of `PercolateMatched` [202] is superior to `User-Matching` [109], hence for our comparison, we have only provided the results corresponding to `PercolateMatched`.

The fifth experiment is done over the Gowalla social network [116]. This dataset contains friendship relations and timestamped check-ins of users to different locations. Using this information, two snapshots of Gowalla network are generated [109]: In the first snapshot, two nodes are connected if they are friends and they check-in to exactly the same location in an even month; the second snapshot is generated similarly, by considering the friendships and check-ins in odd months. In this experiment the number of identifiable nodes, which is defined as the total number of nodes that are present in both graphs G_1 and G_2 with degrees greater than five, is 6634. Figure 3.21 and its zoomed-in version (Figure 3.22) show the superior performance of `ExpandWhenStuck` versus algorithms from [109, 202]. Note that (i) these two Gowalla graphs are not generated through an edge sampling process, i.e., we match two real graphs without any modelling assumptions, and (ii) as Korula and Lattanzi [109] use check-ins to *approximately* the same locations instead of *exactly* the same locations to generate two Gowalla social graphs, our simulation results differ a little bit from their reported results.

Our experiments show that `ExpandWhenStuck` is indeed robust against low node-overlaps between the graphs. For example, in Gowalla (see Figure 3.22) the overlap between the two

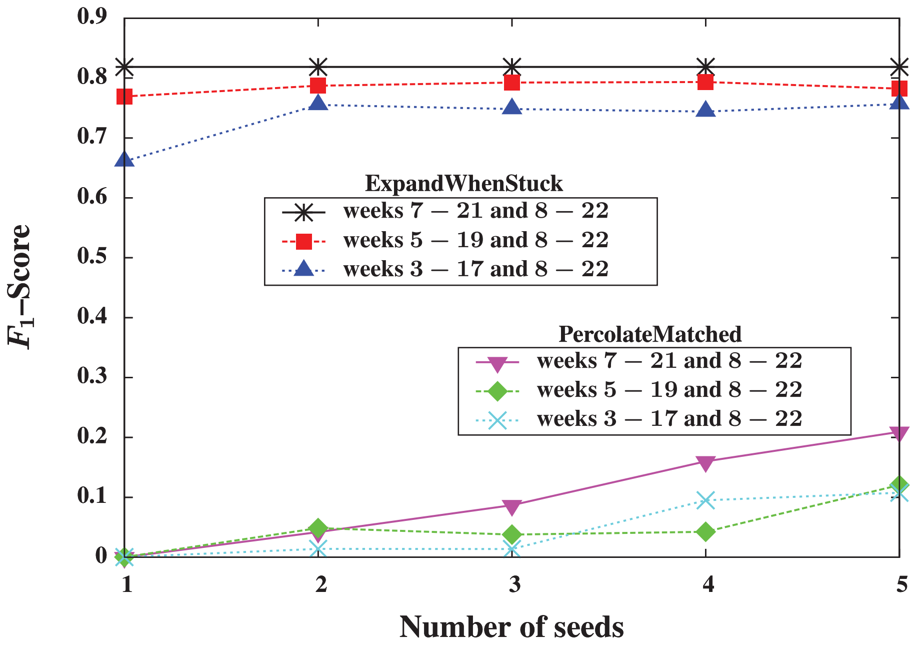


Figure 3.20 – F_1-score (see Equation (3.9)) vs. number of seeds. Simulations are done over EPFL e-mail network with. Each snapshot of the e-mail network is created by aggregating all the exchanged e-mails in a given time period.

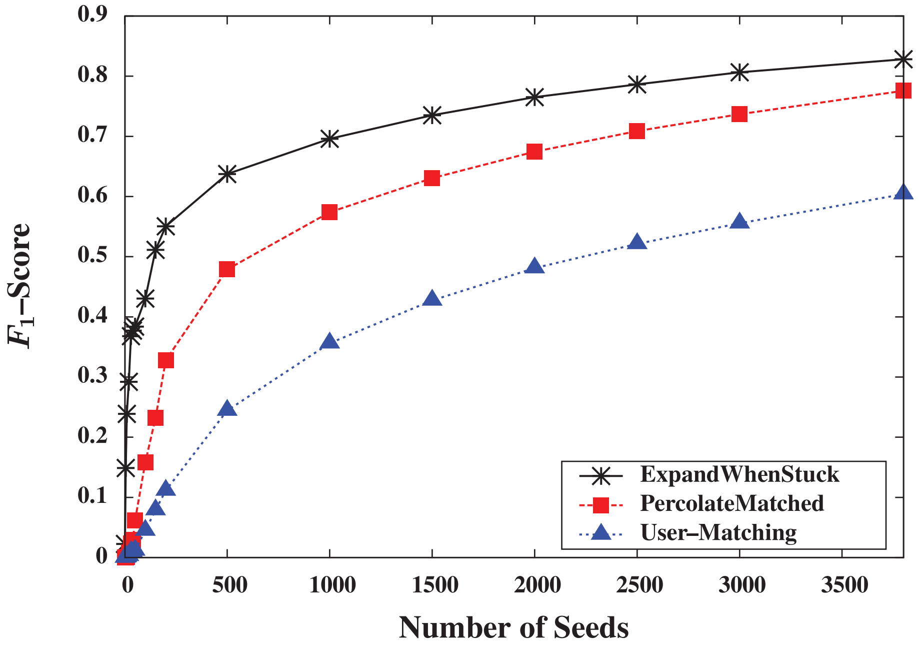


Figure 3.21 – F_1-score (see Equation (3.9)) vs. number of seeds. Simulations are done over Gowalla social network. The number of identifiable nodes, $n_{\text{ident}} = 6634$, is defined as the total number of nodes that are present in both snapshots with degrees greater than five.

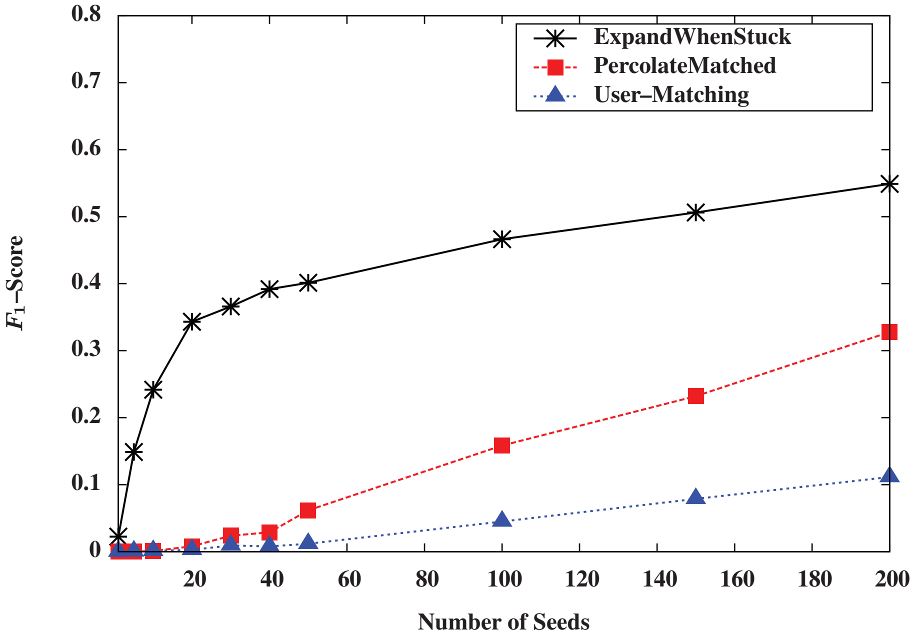


Figure 3.22 – F_1-score (see Equation (3.9)) vs. number of seeds. Simulations are done over Gowalla social network. The number of identifiable nodes, $n_{\text{ident}} = 6634$, is defined as the total number of nodes that are present in both snapshots with degrees greater than five. This figure is the zoomed-in version of Figure 3.21.

graphs is only 0.72. Note that, in the intersection of the Gowalla networks, there are many nodes present in one of the graphs with degree 1. If we consider only the nodes with degrees more than 1 in both graphs, then the overlap reduces to 0.42. As another example, the overlaps for EPFL e-mail networks (see Figure 3.20) are between 0.27 to 0.31. Also, for random graphs with low overlaps, increasing the number of seeds (e.g., only 100 seeds for Chung-Lu graphs with $n = 10^6$, $\beta = 2.5$, $t^2 = 0.49$ and $s^2 = 0.49$) results in good (close to 1) recalls and precisions.

3.4.3 MapReduce implementation

One of the key features of PGM algorithms is their computational simplicity. Nevertheless, for extremely large graphs (100s of millions of nodes or more), the computational and storage overhead for a single machine can still be prohibitive. For this reason, we explored the implementation of a parallelized variant of `ExpandWhenStuck` within the MapReduce framework for scalability; we briefly report the main ideas and results here.

The `ExpandWhenStuck` algorithm cannot be readily parallelized, given the explicitly sequential back-and-forth iterating between spreading marks (lines 9 to 10 in Algorithm 3) and matching new couples (lines 7 to 8 in Algorithm 3). However, we found that without fundamentally affecting the performance of the algorithm, it is possible to reorder these two operations. We can first spread marks from all the eligible couples, then perform the matching of new couples afterwards. More concretely, this approximation of the original `ExpandWhenStuck` algorithm works as follows: (i) We spread marks from the couples in the seed set $\mathcal{A}$; (ii) we add all the couples with at least r marks to the matched set $\mathcal{M}$; (iii) we spread marks from all the new matched couples. The steps (ii) and (iii) are repeated iteratively up to the point that there is no new couple with score at least r; and (iv) at the point the percolation process stops a new set of candidate couples $\mathcal{A}$ is generated from the neighbouring couples of matched couples and the graph matching process continues by returning to step (i).

In this setting, the process of spreading marks can be done independently for all the couples. This enables a parallel implementation of the algorithm through four consecutive MapReduce jobs per iteration. Next we sketch the function of each of these MapReduce jobs, without providing a detailed pseudo-code (in the interest of space):

- The Mapper in the first job spreads out marks from the couples in the candidate set $\mathcal{A}$. The output of Reducer in this job is the set of all the couples with score at least r.
- It is possible for a node to be in several couples with scores above the threshold. The second MapReduce job filters out the nodes that appear in more than one couple with a score of at least r.
- The output of the second MapReduce job is the newly matched couples. These couples are fed to the third MapReduce job to spread their marks and match new couples. The percolation graph-matching process continues by running iteratively the second and third MapReduce jobs.
- When there are no newly matched couples, i.e., the percolation process is stuck, the forth MapReduce job is executed. This job generates a new set of candidate couples $\mathcal{A}$. Provided there are enough seeds, a few iterations of these four MapReduce jobs will correctly match almost all the nodes.

Our MapReduce implementation is able to easily match graphs with millions of nodes. For example, by using a Hadoop cluster with 15 nodes, it took less than twenty minutes to match random graphs with 10 million nodes (starting with 18 seeds); and in under half an hour, the algorithm matches graphs sampled from LiveJournal and Orkut online social networks [116] with 4,847,571 and 3,072,441 nodes, respectively.

3.5 Summary

In this chapter, we have studied the problem of graph matching between two unlabeled graphs when only the structures of the two graphs are available. We characterize the graph-matching problem for graphs with partial-overlapping vertex sets. We give a new percolation graph matching algorithm. We prove that our algorithm correctly matches the nodes that are in the intersection of the two graphs and filters-out the nodes without counterparts in the other

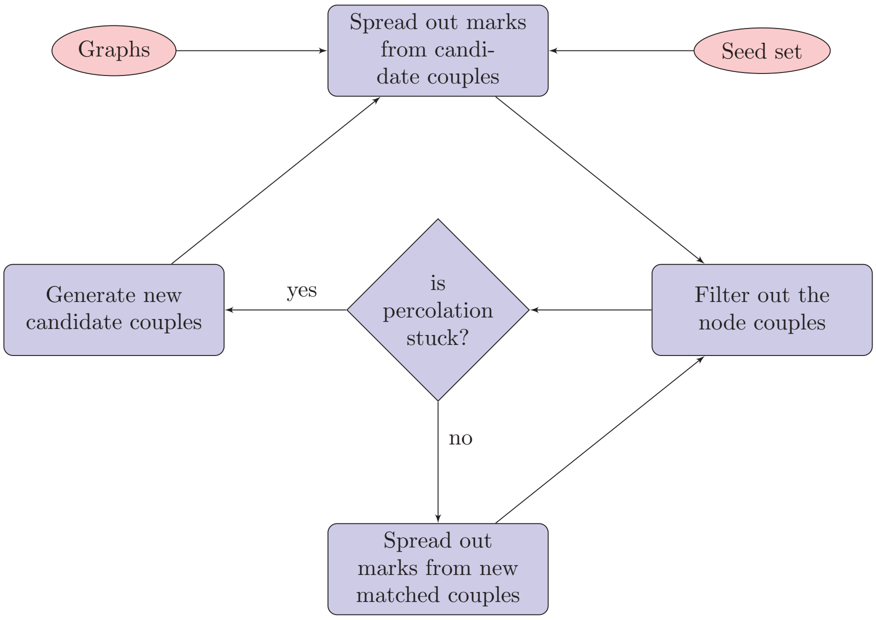

Graphs
Spread out marks from candidate couples
Seed set
Generate new candidate couples
yes
is percolation stuck?
Filter out the node couples
no
Spread out marks from new matched couples

Figure 3.23 – The schematic overview of a variant of the `ExpandWhenStuck` algorithm.

graph. A phase transition in the seed-set size of percolation graph-matching is formally established. Also, we prove that under a wide range of network parameters, our algorithm is robust against a noisy seed-set. As with our algorithmic contribution, we achieve a dramatic reduction in the size of the seed set. We also show the excellent performance in matching several large real social-networks.

Appendix

3.A Table of Notations

Table 3.3 – This table summarizes all the notations used in this chapter.

$(i,j) \in E$	an edge between two nodes i, j in $G(V,E)$
$[i,j]$	a couple of nodes where $i \in V_1$ and $j \in V_2$
$d_{1,i}$	degree of node i in graph G_1
$d_{2,j}$	degree of node j in graph G_2
$\mathcal{A}_0$	initial seed set in Algorithm 1
a_0	size of seed set $\mathcal{A}_0$ in Algorithm 1
$\mathcal{A}'_0$	expanded seed set in Algorithm 2
a'	size of expanded seed set $\mathcal{A}'_0$ in Algorithm 2
$\mathcal{M}_\tau$	set of used couples at time step τ in Algorithm 1
$\mathcal{Z}_\tau$	set of matched couples at time step τ in Algorithm 1
T^*	stopping time of the matching process
$\mathcal{M}^*$	final set of matched couples
$\Lambda(\mathcal{S})$	number of correct couples in a set $\mathcal{S}$ of couples
$\Psi(\mathcal{S})$	number of incorrect couples in a set $\mathcal{S}$ of couples
$V_1(\mathcal{S})$	set of nodes from graph G_1 in a set of couples $\mathcal{S}$
$V_2(\mathcal{S})$	set of nodes from graph G_2 in a set of couples $\mathcal{S}$
τ_{cip}	completion time of the initial phase (cip) of Algorithm 1
$\mathcal{P}_{\ell,\tau}$	set of couples with exactly ℓ marks at time step τ
$\mathcal{P}_{\geq\ell,\tau}$	set of couples with at least ℓ marks at time step τ
$\mathcal{P}_{q,\mathcal{M}^*}$	set of couples that obtain exactly q marks from all the T^* matched couples $\mathcal{M}^* = \mathcal{M}_{T^*}$
$\mathcal{P}_{\geq q,\mathcal{M}^*}$	set of couples that obtain at least q marks from all the T^* matched couples $\mathcal{M}^* = \mathcal{M}_{T^*}$
$\mathcal{C}_{r,\mathcal{M}^*}$	set of all the possible ways of choosing r couples from the set $\mathcal{M}^*$
c_r	a generic element of $\mathcal{C}_{r,\mathcal{M}^*}$

The application perspective Part III

4 Global Pairwise-Network Alignment

A comparative analysis of protein-protein interaction (PPI) networks provides insight into the evolution of organisms and information about the evolutionarily-conserved biological interactions. Network-alignment algorithms are one of the most powerful tools to compare PPI networks. PPI-network alignment has many applications in areas such as the detection of new pathways and of conserved motifs, the prediction of the functions of proteins, orthology detection, drug design, protein-protein interaction prediction and phylogenetic tree reconstruction [111, 175].

PPI-network alignment algorithms use topological (e.g., local and global network structures) and biological (e.g., amino acid sequences of proteins) information to align two (or several) networks. The topological information is more important than sequence information for aligning functionally conserved interactions [51, 129], hence the focus of network-alignment algorithms shifted from using only biological information towards using topological information. Local network-alignment (LNA) and global network-alignment (GNA) methods are the main approaches for aligning PPI networks. Most of the early works on PPI-network alignment, such as PathBLAST [104], NetworkBLAST [172], NetAlign [118], MaWISh [110] and Græmlin [62], study the local network-alignment (LNA) problem. More recent methods, such as IsoRank [119, 176], the GRAAL family [111, 112, 129, 134, 137], MAGNA and its successor MAGNA++ [165, 194], SPINAL [10], PINALOG [154], Netcoffee [80] and BEAMS [12], are examples of global network-alignment (GNA) algorithms.

In this chapter[1], we consider the problem of global pairwise-network alignment. Singh et al. [176] introduced IsoRank as the first GNA algorithm for PPI networks. The IsoRank algorithm is formulated as an eigenvalue problem, where it first computes a pairwise protein similarity metric (as a convex combination of protein-sequence similarities and a structural-similarity score), and then generates the final global alignment between the two networks based on this metric. Bayati et al. [24] developed approximation algorithms for efficient computation of the IsoRank similarities. GHOST [149] aligns two networks according to the similarity of

---

[1] The material of this chapter is based on [101].

spectral signatures of node couples. PINALOG [154] finds the final alignment by matching the communities of the two networks first. The GRAAL (GRAph ALigner) family is a group of GNA methods that use the graphlet-degree signature similarity to align two networks. GRAAL [112] is the first GNA algorithm that uses only structure of the two networks for alignment. It first selects a couple of nodes with high graphlet-degree signature similarity; then, by a seed-and-extend matching procedure, it tries to expand the alignment around this couple in a greedy way. In general, a seed-and-extend algorithm starts the alignment procedure from a set of highly similar couples called seed pairs. Then, it proceeds to align iteratively similar couples among neighbors of the seed pairs. H-GRAAL [137] uses the Hungarian algorithm for improving the quality of alignments produced by GRAAL, at the cost of increased computational complexity. To align two networks, MI-GRAAL [111] integrates several metrics such as graphlet-degree signature similarity, local clustering coefficient differences, degree differences and protein sequence similarity. L-GRAAL [129] is the latest algorithm from the GRAAL family; it directly optimizes both the structural and sequence similarities with a heuristic seed-and-extend strategy based on a Lagrangian relaxation. The SPINAL algorithm [10] iteratively grows an alignment based on an a priori computed coarse-grained node-similarity scores. By using a genetic algorithm, MAGNA [165] tries to optimize the edge conservation between two networks.

In this chapter, we design a new global pairwise-network alignment algorithm for PPI networks; it is built upon our previous results for graph matching (see Chapters 2 and 3). We show the excellent performance of our algorithm (in terms of both accuracy and speed) compared to several state-of-the-art algorithms. We also introduce a new measure for evaluating the performance of algorithms in aligning biological pathways between species. We argue the suitability of our algorithm by analyzing its performance in a bigraph-sampling model of network evolution, similar to the model from Chapter 2. For this random-bigraph model, we use the results of Chapters 2 and 3 to guarantee the performance of our algorithm.

4.1 The PROPER Algorithm: Two Steps

GNA algorithms, by finding a one-to-one mapping of proteins, try to find large conserved sub-networks (as they are indicative of a common ancestor) and network motif[2] among several species [43]. Pairwise-network alignment algorithms align proteins of only two species in order to maximize the biological and topological similarities (these concepts are defined precisely later in the text) between aligned proteins; they have been extensively studied in the literature [10, 57, 59, 154, 176]. In this section, we use the ideas from the PGM class of network-alignment algorithms (mainly from Chapter 3) to design our PROPER (PROtein-protein interaction network alignment based on PERcolatin) algorithm.

A PPI network can be represented by a graph $G(V,E)$, where V is the set of proteins and each

---

[2] A network motif is a small recurrent connected-subgraph that occurs in PPI networks (and other biological networks) significantly more often than in random networks.

edge (u, v) in E is an indicator of interaction between the two proteins u and v. Formally speaking, given two networks $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$, the purpose of global network-alignment is to identify a bijection between the full (or partial) vertex sets of two networks. The network-alignment algorithms use the protein similarities and the network topology. The pairwise similarities between proteins are computed by the well-known basic local-alignment search tool (BLAST) [13] that considers the alignment of amino-acid sequences of those proteins.

In the process of PPI-network alignment by PROPER, initially we have as inputs two PPI networks $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$, the set of pairwise BLAST bit-score similarities (call it $\mathcal{S}$) for couples of proteins in $V_1 \times V_2$, and fixed thresholds $\ell, r > 0$, where ℓ and r are the sequence similarity and the local topological similarity thresholds, respectively. The PROPER algorithm uses the sequence similarities and network structures in a two-stage procedure: (i) At the first step, it uses the sequence similarities to generate a seed set for a PGM algorithm; and (ii) at the second step, to align remaining couples, it uses only the network structure and the seeds generated from the first step as inputs to the PGM algorithm. This is in contrast with many other pairwise algorithms, where they try to simultaneously maximize a function of both sequence and structural similarities. In this section, we first explain the process of generating seed set $\mathcal{A}$ from $\mathcal{S}$ (the `SeedGeneration` algorithm). Then, we explain how to align new couples, starting from the set $\mathcal{A}$ (the `MapPercolation` algorithm).

4.1.1 The First Step: `SeedGeneration`

Initial seeds play an important role in the alignment process. In the PPI setting, the BLAST bit-score is often a good indicator of functional similarities between proteins [92]. In other words, at high levels of sequence similarity it is possible to make a functional inference with an acceptable accuracy [153]. This means that, for couples of proteins with a high sequence-similarity it is very likely that they have similar functions. The main approach in this chapter is to use such couples as a starting point to find a global alignment. Indeed, the seeds to the PROPER algorithm are those couples of proteins with high sequence-similarities. Also, a protein can be aligned with at most one protein from the other species. The degree of similarity between the couples in the seed set $\mathcal{A}$ is controlled by the threshold ℓ.

The seed set $\mathcal{A}$ is generated from the pairwise similarities (the set $\mathcal{S}$) in the following manner: Among all the couples of proteins with BLAST bit-score similarity above ℓ, couples $[i, j]$ are matched in a descending order of sequence similarity, unless i or j is matched already. More precisely, (i) we add the couple $[i, j] \in \mathcal{S}$ with the highest similarity to the seed set and match i to j; (ii) all the couples $[i, j']$ and $[i', j]$ are now forbidden and we remove them from $\mathcal{S}$. We repeat the steps (i) and (ii) until there is no remaining couple in the set $\mathcal{S}$ with BLAST bit-score similarity at least ℓ. Note that, in the process of seed generation, when there are several couples with the same sequence similarity, we randomly pick one of them.

Algorithm 6 describes the `SeedGeneration` algorithm in detail. In this algorithm, for a set of couples $\mathcal{A}$, $V_1(\mathcal{A})$ defines the set of nodes from network G_1 in $\mathcal{A}$, i.e., $V_1(\mathcal{A}) = \{i | \exists j \text{ s.t. } [i, j] \in$

$\mathcal{A}$ for some $j\}$. We define $V_2(\mathcal{A})$ similarly. Also, $BlastBit(i,j)$ denotes the BLAST bit-score similarity between two proteins i and j.

A priori, the probability of biological similarity of a protein couple decreases with a decrease in the sequence similarity. Therefore, there is a trade-off between the number of protein couples with the same biological functions and the accuracy (i.e., the ratio of couples with the same functions over the size of seed set) based on ℓ. Clearly, choosing a high value for ℓ aligns proteins that, with a high probability, have similar functions. However, this can result in removing couples with lower sequence-similarities, but the same functions from the initial seed-set.

Algorithm 4: The `SeedGeneration` Algorithm

```
Input: BLAST bit-score similarities $\mathcal{S}$ and $\ell$
Output: The seed set $\mathcal{A}$
$\mathcal{A} \leftarrow \emptyset$;
for all couples $[i,j] \in \mathcal{S}$ from the highest similarity to the lowest do
    if $i \notin V_1(\mathcal{A})$, $j \notin V_2(\mathcal{A})$ and $BlastBit(i,j) \geq \ell$ then
        add the couple $[i,j]$ to $\mathcal{A}$;
return $\mathcal{A}$;
```

4.1.2 The Second Step: `MapPercolation`

The second step of PROPER (the `MapPercolation` algorithm) starts the alignment process from the seed couples (set $\mathcal{A}$) obtained from the set of pairwise similarities $\mathcal{S}$ (see the `SeedGeneration` algorithm). It then incrementally generates the set π of matched couples among $V_1 \times V_2 \setminus \mathcal{A}$. In the `MapPercolation` step, the PROPER algorithm relies only on the structure of $G_{1,2}$, and it does not use the sequence similarities. In this regard, the seed couples are added to the set of aligned couples π. Then, at each time-step, the goal of the PGM algorithm is to add a new couple to the set π so that structural similarity is maximized.

In the process of the `MapPercolation` algorithm, we look at the neighboring couples of the previously matched couples. We say a couple of proteins $[i',j'] \in V_1 \times V_2$ is a neighbor of another couple $[i,j]$ if and only if $(i,i') \in E_1$ and $(j,j') \in E_2$. To achieve the maximum structural similarity, our algorithm chooses the next couple in a greedy way: it chooses the couple with the maximum number of common neighbors (provided there are at least r) in π and permanently aligns them. Indeed, the evidence for deciding which couple to match (called the score of a couple) is the number of common neighbors each couple has in the set of currently aligned couples. A new couple of proteins can be matched if its score is at least r.

When there are several couples with the maximum score, we tie-break by the minimum degree-difference in the two networks, i.e., we choose the couple $[i,j]$ with the minimum $|d_{1,i} - d_{2,j}|$, where $d_{1,i}$ and $d_{2,j}$ denote the degrees of nodes i and j in the networks G_1 and G_2, respectively.

If there are more than one couples with the minimum degree difference, we choose the couple with the minimum $d_{1,i} + d_{2,j}$. Finally, if there are still several candidate couples, we randomly pick one of them. The process of alignment continues to the point where there is no remaining unmatched couple of proteins (we say a couple $[i, j]$ is unmatched if $i \notin V_1(\pi)$ and $j \notin V_2(\pi)$) with at least r common neighbors, in the current set of aligned proteins. Note that for a given value of r, only nodes with degree at least r can get enough score to be matched. More precisely, `MapPercolation` is not able to align: (i) unmatched nodes with a degree less than r, and (ii) couples that have not gained enough scores. Figure 4.1 presents an example of the second step of PROPER (the `MapPercolation` algorithm). Algorithm 5 describes this algorithm.

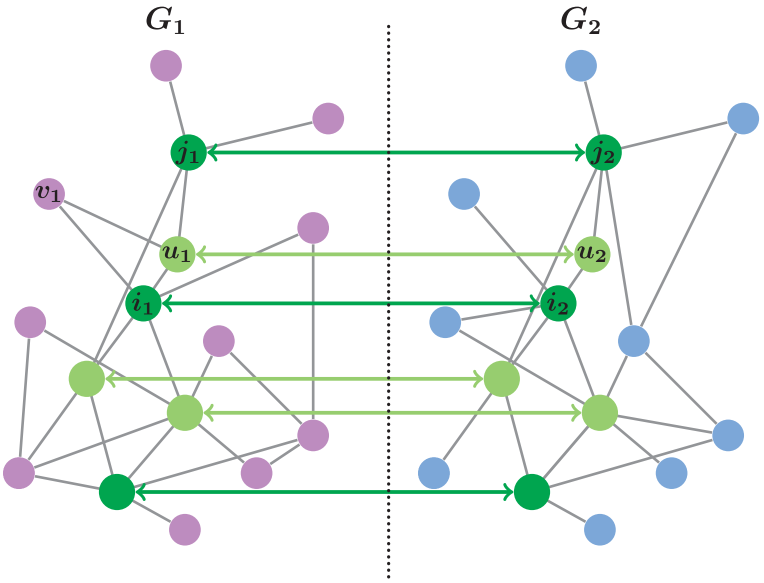


Figure 4.1 – Dark-green nodes correspond the initial seed-set. Couples $[i_1, i_2], [i_1, j_2], [j_1, j_2]$, $[j_1, i_2], [v_1, i_2], [v_1, j_2]$ are neighboring couples of the couple $[u_1, u_2]$. The couples $[i_1, i_2]$ and $[j_1, j_2]$ are the common neighbors of the couple $[u_1, u_1]$ in the set of already matched couples π, i.e., the score of couple $[u_1, u_2]$ is two. Light-green nodes are the nodes that are matched after the first three steps of the `MapPercolation` algorithm. In this example, we set $r = 2$.

4.2 Performance Measures

In this section, we explain the measures used for comparing alignment algorithms. As there is no single standard measure for evaluating the quality of alignments, we use several existing measures [43, 57, 59]). In addition, we introduce a new measure for comparison based on the performance of algorithms in aligning biological pathways.

For better illustration, in this section we assume that, without loss of generality, G_2 has at least as many nodes as G_1, i.e., $|V_1| \leq |V_2|$. Let π denote the mapping produced by an alignment

Algorithm 5: The `MapPercolation` algorithm

```
Input: $G_1(V_1, E_1), G_2(V_2, E_2)$, seed set $\mathcal{A}$ and threshold $r$
Output: The set of aligned couples $\pi$
$\pi \leftarrow \mathcal{A}$;
while there exists an unmatched couple with score at least $r$ do
    among all the couples with the highest score select the unmatched couple $[i, j]$ with the minimum $|d_{1,i} - d_{2,j}|$. If there are more than one couples with the minimum $|d_{1,i} - d_{2,j}|$, select the couple with the minimum $d_{1,i} + d_{2,j}$. Finally, if there are still several candidates, randomly pick one of them;
    add $[i, j]$ to the set $\pi$;
return $\pi$;
```

algorithm. Also, let $G[V]$ denote the induced subgraph of G on the set of vertices V. Assume π maps the nodes $V_1' \subset V_1$ to the nodes $V_2' \subset V_2$. Note that many global alignment algorithms do not match all the nodes from graph G_1 to a node from graph G_2, i.e., they align a large fraction of the nodes but not all of them. We define graph $G_0(V_0, E_0)$ as the intersection of the two graphs G_1 and G_2 under the alignment π, i.e., V_0 is the set of proteins in graph G_1 aligned by π to a protein in graph G_2; and E_0 is the set of interactions in G_1, conserved under the alignment π in the graph G_2. Formally, we have $V_0 = V_1'$ and $E_0 = E_{G_1[V_1']} \cap \pi^{-1}(E_{G_2[V_2']})$.

4.2.1 Structural and Functional Similarity Measures

In this section, we review the measures that are used widely to evaluate the performance of network-alignment algorithms.

(i) Node correctness (NC) of an alignment is defined as the ratio of the number of correctly aligned couples to the number of nodes in the smaller network (i.e., $|V_1|$) [112]. The precision is defined as the ratio of number of correctly aligned couples to the total number of couples $|\pi|$ in the alignment π. These measures are applicable only to synthetic networks, because they can be used only for alignments with known ground-truth [59].

As the true alignment between the proteins of two species is not known completely for real networks, it is not possible to directly calculate the NC and precision of an alignment [43, 57, 59]. To compare the performance of algorithms over real datasets, two different types of measures were introduced in the literature. The first group of measures uses the topological similarity of aligned networks. The second group measures the quality of an alignment by using other biological information.

The following measures are used for evaluating the structural (topological) similarity of aligned networks.

(ii) The number of conserved interactions under the alignment π (call it Δ_π) is one of the measures used to evaluate the quality of algorithms based on the topological similarity [207]. Formally,

$$\Delta_\pi = |\pi(E_1) \cap E_2|.$$

(iii) Edge correctness (EC) is a measure of topological similarity among the aligned networks [112]. EC computes the ratio of edges from graph G_1, i.e., all the edges in the smaller network, which are conserved under the alignment π. Formally,

$$\mathrm{EC} = \frac{|\pi(E_1) \cap E_2|}{|E_1|}.$$

(iv) Recall that the numbers of proteins (nodes) in the two networks are not equal. Therefore, one drawback of the EC measure is that aligning sparse regions of G_1 with dense regions of G_2 can result in high values of EC. The induced conserved-structure score (ICS) measures the structural similarity of aligned networks by penalizing dense regions of G_2 [149]. The ICS score for an alignment π from graph G_1 with graph G_2 is

$$ICS = \frac{|\pi(E_1) \cap E_2|}{|E_{G_2[\pi(V_1)]}|}.$$

(v) The symmetric substructure score (S^3) is defined with respect to both $G_{1,2}$ networks [165]. The S^3 measure penalizes the alignments that map sparse regions of one network to denser regions of the other network. Formally, S^3 is defined as follows.

$$S^3 = \frac{|\pi(E_1) \cap E_2|}{|E_1| + |E_{G_2[\pi(V_1)]}| - |\pi(E_1) \cap E_2|}.$$

Note that $|E_1|$ refers to all the edges in the smaller network.

(vi) The largest connected shared-component (LCSC) is the largest connected subgraph of G_1, which is found to also exist in G_2, i.e., the largest connected component in graph G_0 [43]. Let $|LCSC|$ denote the number of nodes in LCSC. Also, the share of nodes in LCSC is defined as $\frac{|LCSC|}{|V_1|}$ [111].

We now introduce the second group of measures that are used for evaluating the biological quality of alignments by comparing the functional similarity of aligned proteins.

(vii) The gene-ontology consistency (GOC) score measures the functional similarity of aligned proteins. Note that usually more than one gene ontology (GO) terms are assigned to a protein [17]. Also, as the GO datasets are noisy and proteins have diverse functions, it is possible that true ortholog proteins do not have exactly the same set of GO terms. GOC for an aligned couple of proteins $u \in V_1$ and $v \in V_2$ is defined as the Jaccard similarity coefficient between the GO terms of the two proteins [10]. Formally, it is defined as

$$GOC(u,v) = \frac{|GO(u) \cap GO(v)|}{|GO(u) \cup GO(v)|},$$

where $GO(u)$ denotes the set of GO terms associated with the protein u. $GOC(\pi)$ is calculated by summation over the GOC terms of all the aligned couples in π:

$$GOC(\pi) = \sum_{u \in V_1} GOC(u, \pi(u)). \tag{4.1}$$

For ease of notation we refer to $GOC(\pi)$ as GOC score.

(viii) To compare algorithms based on the sequence similarities of aligned proteins, we use a slightly modified version of the average normalized bit–score (ANBS) measure proposed in [169]. ANBS for two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ under the alignment π is defined as follows.

$$ANBS(\pi) = |V_1|^{-1} \sum_{i \in V_1(\pi)} \frac{BlastBit(i, \pi(i))}{\sqrt{BlastBit(i,i) BlastBit(\pi(i), \pi(i))}}.$$

4.2.2 Pathway Comparison Measures

In order to evaluate the performance of algorithms in aligning biological pathways, we introduce a new measure in this section. This measure captures the quality of alignments based on a higher level of functional and structural similarities (beyond the introduced measures such as the similarity of GO terms and the number of conserved interactions).

It is known that there are many biological pathways with similar functions in different species [103]. The KEGG PATHWAY database [3] provides a set of experimentally found biological pathways. In this database, a pathway is called by the name of a species (e.g., hsa for Homo sapiens), followed by a number. The pathways with the same number have the same function in different species. For example, hsa03040, mmu03040, dme03040 and sce03040 are in Homo sapiens (human), Mus musculus (mouse), Drosophila melanogaster (fruit fly) and Saccharomyces cerevisiae (budding yeast), respectively. These pathways have the same functions.[3] Assume $PW_{i,1}$ denotes the set of proteins from a pathway with number i in the PPI network of the first species (i.e., G_1). Similarly, we define $PW_{i,2}$. For pathway i, $\Delta_{\pi,i}$ denotes the number of conserved interactions between the proteins in this pathway under the alignment π, i.e., $\Delta_{\pi,i} = E_{G_1[PW_{i,1}]} \cap \pi^{-1}(E_{G_2[PW_{i,2}]})$. Note that we are looking for pathways that are present in both aligned species.

We say a protein u from a pathway is aligned correctly, if it is mapped to a protein v from a pathway with the same function. For pathway i, we define the number of correctly mapped

---

[3] These pathways are Spliceosome. Spliceosome removes introns from a transcribed pre-mRNA, a type of primary transcript.

proteins as $|PW_{i,1} \cap \pi^{-1}(PW_{i,2})|$. This measure corresponds to the number of proteins that, from pathway i in the first species, are mapped to a protein from the same pathway in the second species. For pathway i, we define the accuracy as

$$acc_{\pi,i} = \frac{2|PW_{i,1} \cap \pi^{-1}(PW_{i,2})|}{|PW_{i,1}| + |PW_{i,2}|}. \tag{4.2}$$

This measure corresponds to the fraction of correctly mapped proteins in pathway i.

We conjecture that a good alignment algorithm should align proteins from pathways with the same functions across species, and many interactions among these proteins are conserved. To quantify this expectation, we set a threshold over the structural similarity of aligned pathways to consider them as a correct alignment. We say that an alignment π successfully aligns a pathway i, if there are at least δ conserved interactions under the alignment π for proteins in that pathway, i.e., if $\Delta_{\pi,i} \geq \delta$. This thresholding guarantees that the structural similarity of aligned pathways are more than a minimum value (here, δ conserved interactions). To evaluate the performance of an algorithm based on this thresholding criterion, we define a set of measures as follows.

1. We consider pathways with at least δ (say $\delta \geq 2$) interactions in each of species. Let "#PW$_\delta$" denote the number of such pathways.
2. Alignment π successfully aligns pathway i, if $\Delta_{\pi,i} \geq \delta$. The variable "#FPW$_\delta$" refers to the number of successfully aligned pathways. We define the recall as

$$recall_{\pi,\delta} = \frac{\#\mathrm{FPW}_\delta}{\#\mathrm{PW}_\delta}. \tag{4.3}$$

3. Again, for a correctly aligned pathway i, we define $acc_{\pi,\delta,i}$ similar to (4.2).

The averages over all i of all the $acc_{\pi,i}$ and $acc_{\pi,\delta,i}$ values are represented by $\overline{acc_\pi}$ and $\overline{acc_{\pi,\delta}}$, respectively. Figure 4.2 provides a toy example of how to calculate the pathway alignment measures.

4.3 Experimental Results

In this section, we compare PROPER with the main state-of-the-art network-alignment algorithms, specifically (i) with L-GRAAL as the most recent member of GRAAL family that takes into account both sequence and structural similarities [129]; (ii) with MAGNA++ that tries to maximize one of the EC, ICS or S^3 measures [165, 194] (In our experiments we run MAGNA++ in two different modes of maximizing S^3, which is the superior mode for MAGNA++ [165], and EC); (iii) with IsoRank [176] as one of the first global PPI-network alignment algorithms; (iv) with PINALOG [154]; and (v) with SPINAL I and II [10] as their performances are reported to be among the best alignment-algorithms [43]. Table 4.1 provides an overview of the arguments

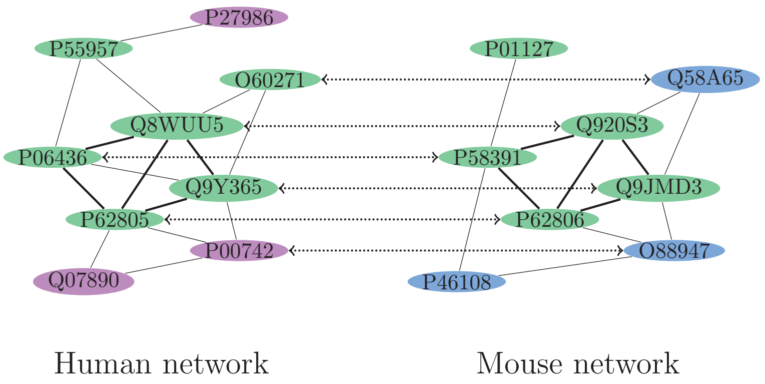


Figure 4.2 – In this figure, two example PPI networks are given. Green nodes are proteins which are in the same pathway (i.e., a pathway with the same number in both species). Dotted lines represent the alignment π between these two networks. Under this alignment, there are five conserved interactions between proteins in this pathway (shown by thick black edges in each network). Also, the number of correctly mapped proteins is four. Therefore, the accuracy of aligning this pathway is $acc_{\pi,i} = \frac{2\times 4}{6+5}$, where there are six and five proteins from this pathway in each species, respectively.

and parameters of the algorithms used in our comparisons. Note that it is recommended to use SPINAL and MAGNA++ in modes I and S^3, respectively. Also, the recommended settings for IsoRank is $\alpha = 0.6$. For the other algorithms, no default setting is provided. We evaluate the performance of PROPER with $r = 1$ (structural similarity threshold) and different values of ℓ (sequence similarity threshold).

Table 4.1 – Algorithms and their parameters

Algorithm	Commandline arguments	Parameters
IsoRank [176]	–K 50 –thresh 1e-5 –alpha α –maxveclen 1000000	$\alpha \in \{0.3, 0.5, 0.6, 0.7\}$
PINALOG [6, 154]	do not require arguments	none
L-GRAAL [129]	-a α -I 50	$\alpha \in \{0.3, 0.5, 0.7\}$
MAGNA++(S^3) [194]	-m S3 -p 1000 -n 15000 -f 5 -a α -t 16	$\alpha \in \{0.3, 0.5, 0.7\}$
MAGNA++(EC) [194]	-m EC -p 1000 -n 15000 -f 5 -a α -t 16	$\alpha \in \{0.3, 0.5, 0.7\}$
SPINAL I [10]	–mode -I –alpha α	$\alpha \in \{0.3, 0.5, 0.7\}$
SPINAL II [10]	–mode -II –alpha α	$\alpha \in \{0.3, 0.5, 0.7\}$

All the algorithms use two sets of data as input: (i) the PPI networks of two species, and (ii) the pairwise BLAST similarities (in form of BLAST bit-score) between proteins from the first species and proteins from the second species. We use two different PPI-network databases for our comparisons. The first one is from IntAct molecular interaction database [1, 76]. This database enables us to compare algorithms based on large and more recent PPI networks. The GO annotation terms are extracted from the Gene Ontology Annotation (UniProt-GOA) Database [5, 22]. For pathway comparisons over these networks we can use data from [3]. The second database is Isobase [148], a common dataset used in comparison of recent algorithms [43, 57]. The results for experiments over Isobase dataset are provided in Appendix 4.A. For further evaluations, we use synthetic networks with a known ground-truth.

4.3.1 Structural and Functional Based Comparisons

Table 4.2 provides a brief description of the PPI networks for five major eukaryotic species, namely C. elegans (ce), D. melanogaster (dm), H. sapiens (hs), M. musculus (mm) and S. cerevisiae (sc); they are extracted from the IntAct database [1, 76]: The last column of Table 4.2 shows the number of pathways of each species from KEGG PATHWAY database [3]. The amino-acid sequences of proteins for each species are extracted in the FASTA format from UniProt database [4, 15]. The BLAST bit-score similarities [13] are calculated using these amino acid sequences.

Table 4.2 – PPI networks of five major eukaryotic species from IntAct molecular interaction database [1, 76].

species	Abbrev.	#nodes	#edges	Avg. deg.	#pathways
C. elegans	ce	4950	11550	4.67	117
D. melanogaster	dm	8532	26289	6.16	127
H. sapiens	hs	19141	83312	8.71	288
M. musculus	mm	10765	22345	4.15	284
S. cerevisiae	sc	6283	76497	24.35	98

Figure 4.3 compares algorithms based on the average ICS versus average GOC score for all the possible 10 pairwise alignments between the species from Table 4.2. We observe that PROPER outperforms the other algorithms in both measures, i.e., the PROPER algorithm finds alignments with higher functional (GOC score) and structural (ICS) similarities. Also, although the other algorithms claim the parameter α controls the contributions of structural and sequence similarities, we observe that, in practice, these algorithms fail to trade-off between these similarity measures. For the detailed comparisons of the algorithms refer to Figures 4.6, 4.7, 4.8 and Appendix 4.A.

Note that many of the GO annotations are based on only sequence similarities, and these annotations could increase the GOC scores artificially. Clark and Kalita [43] (similar to [10]) propose to also compare algorithms by using only the experimentally verified GO terms (along with the comparisons based on all the GO terms) to eliminate the effects of sequence similarities in the GOC evaluations. For this reason, in our next experiment, we consider only GO terms with codes “EXP”, “IDA”, “IMP”, “IGI”,“IEP” and “IPI” (the codes for experimental GO terms), and we exclude the annotations derived from computational methods. Figure 4.4 compares the GOC (based on experimentally verified GO terms) versus EC score. The result of this experiment confirms the superiority of PROPER over the other algorithms.

Figure 4.5 evaluates the performance of algorithms based on S^3 (for structural similarity) and ANBS (for functional similarity) measures. Again, the PROPER algorithm performs the best with respect to the two measures, simultaneously.

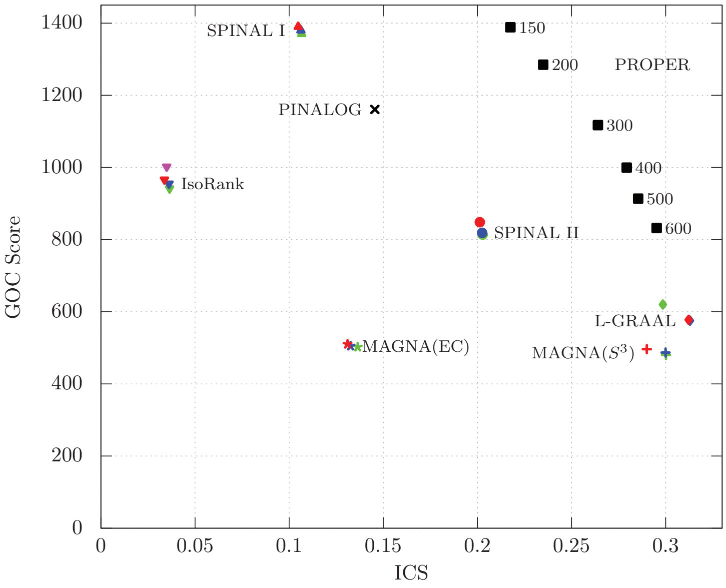


Figure 4.3 – Comparison of different global network-aligners based on the average GO consistency vs. average integrated conserved structure score. For the PROPER algorithm, we set $r = 1$ and each point corresponds to a different value of ℓ. Also, the red, blue, magenta and green points correspond to the parameters $\alpha = 0.3, 0.5, 0.6$ and 0.7, respectively.

Table 4.3 reports the average number of aligned couples and the average of share of nodes in LCSC. We observe that MAGNA++ and IsoRank, irrespectively of the similarity of networks, find alignments with the full coverages, i.e., the size of their alignments is equal to the number of nodes in the smaller network; and PINALOG has the lowest coverage among the algorithms. The size of an alignment alone is not a good indicator of its quality, because an algorithm with a large coverage might find alignments with low functional-similarities and structural-similarities. Instead, we can consider the sum of functional similarities of aligned proteins. To address this point, for example, GOC score (4.1) captures the total functional similarity, by summation over all the couples in π (see Figures 4.3 and 4.4). We can also consider the size of shared structure between networks. To address this second point, we use LCSC. A larger LCSC implies that we have found a larger amount of shared structure between the two PPI networks [111]. From Table 4.3, we observe that PROPER, L-GRAAL and SPINAL II outperform the other algorithms (with huge margins), based on the share of nodes in LCSC.

Figure 4.6 provides a detailed comparison between the algorithms based on their performance in aligning H. sapiens with S. cerevisiae. Also, detailed comparisons between C. elegans and D. melanogaster, and M. musculus and S. cerevisiae are provided in Figures 4.7 and 4.8, respectively. Note that in Figures 4.6, 4.7 and 4.8, the values for each measure are normalized

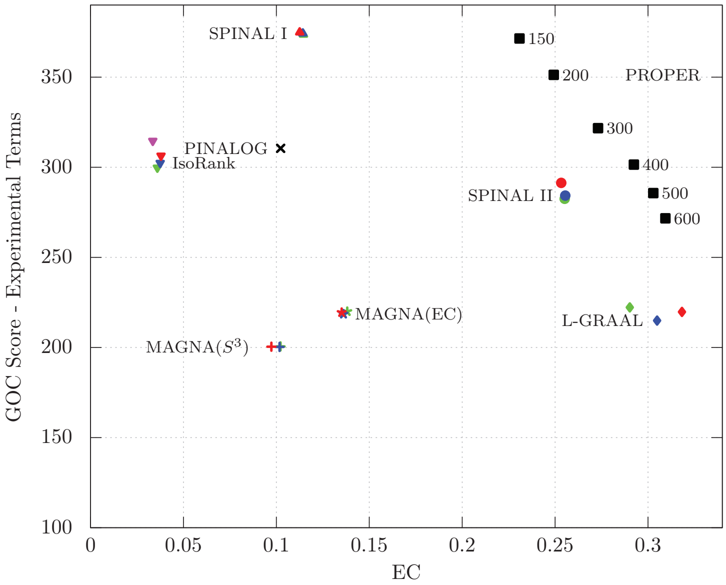


Figure 4.4 – Comparison of different global network-aligners based on the average GO consistency vs. EC score. For the PROPER algorithm, we set $r = 1$ and each point corresponds to a different value of ℓ. Also, the red, blue, magenta and green points correspond to the parameters $\alpha = 0.3, 0.5, 0.6$ and 0.7, respectively.

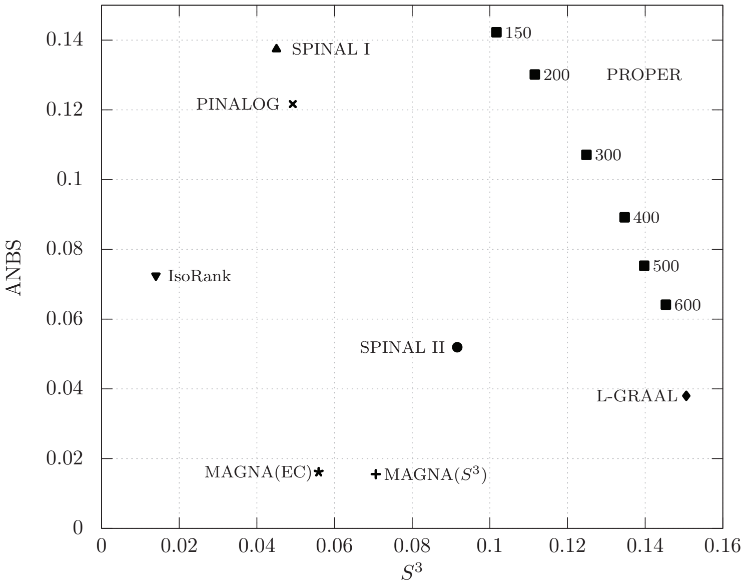


Figure 4.5 – Comparison of different global network-aligners based on the average ANBS vs. average S^3 score. For the PROPER algorithm, we set $r = 1$ and each point corresponds to a different value of ℓ. The parameter α is 0.7.

Table 4.3 – This table reports the average number of aligned couples (i.e., $|\pi|$) and the average of share of nodes in LCSC (i.e., $|LCSC|/|V_1|$). We use $\alpha = 0.7$ for SPINAL, IsoRank, MAGNA and L-GRAAL, and $r = 1$ for PROPER.

Algorithms	$\|\pi\|$	$\|LCSC\|/\|V_1\|$
PROPER ($\ell = 150$)	5521.2	0.528
PROPER ($\ell = 600$)	5347.4	**0.728**
SPINAL I	6364.3	0.219
SPINAL II	6433.4	0.720
PINALOG	3740.9	0.233
L-GRAAL	5616.4	0.726
MAGNA++(S^3)	**6647.8**	0.292
MAGNA++(EC)	**6647.8**	0.353
IsoRank	**6647.8**	0.051

to the highest value, i.e., for each measure, in these figures, the maximum is 1 for the best algorithm and values for the other algorithms are normalized with respect to the maximum. We observe that PROPER outperforms the other algorithms in terms of most of GOC, ANBS, ICS, S^3, EC and LCSC measures.

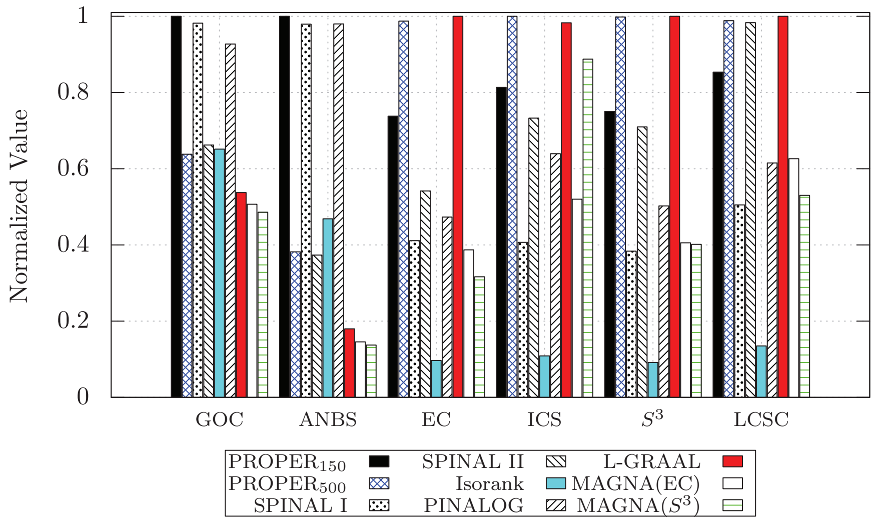


Figure 4.6 – Comparison of different global network-aligners on aligning H. sapiens and S. cerevisiae based on six different measures. For the PROPER algorithm, we set $r = 1$ and $\ell \in \{150, 500\}$. The parameter α is 0.7.

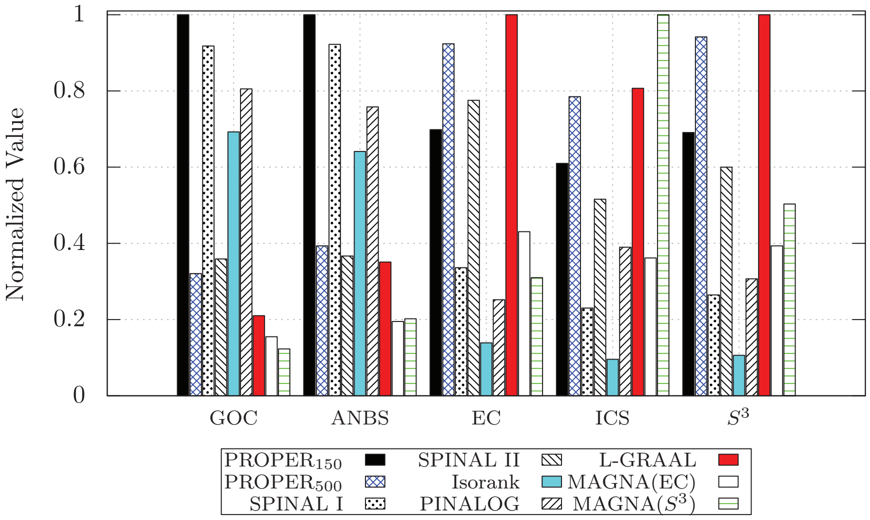


Figure 4.7 – Comparison of different global network-aligners on aligning C. elegans and D. melanogaster based on six different measures. For the PROPER algorithm, we set $r = 1$ and $\ell \in \{150, 500\}$. The parameter α is 0.7.

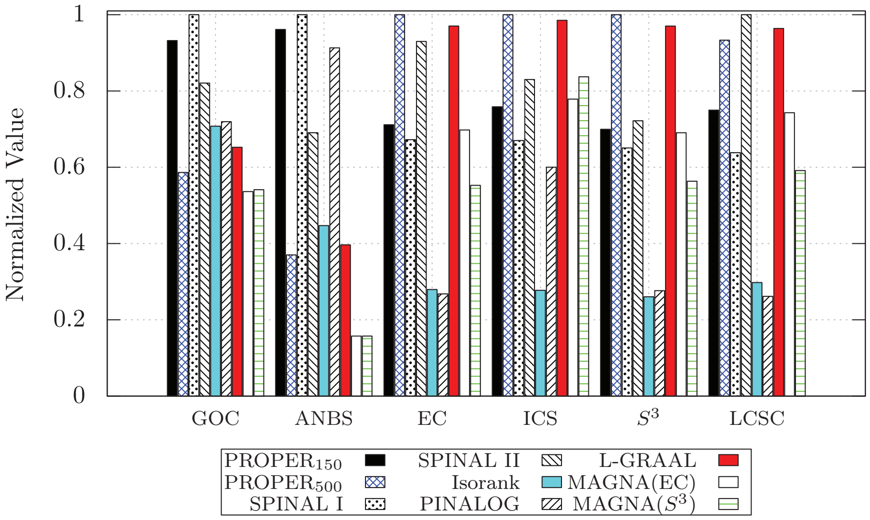


Figure 4.8 – Comparison of different global network-aligners on aligning M. musculus and S. cerevisiae based on six different measures. For the PROPER algorithm, we set $r = 1$ and $\ell \in \{150, 500\}$. The parameter α is 0.7.

4.3.2 The MapPercolation Algorithm and r

The PROPER algorithm has two main steps: (i) `SeedGeneration` and (ii) `MapPercolation`. The number of aligned couples in the first and second steps are functions of ℓ and r, respectively. In Table 4.5, we report the average number of aligned couples (i.e., $|\pi|$) in the first and second steps of PROPER for different values of ℓ and $r \in \{1,2\}$. We observe that by increasing the value of ℓ, the number of aligned couples in the first step decreases. This is because the number of couples with BLAST bit-score of at least ℓ has an inverse relationship with ℓ. In the second step, $|\pi|$ increases by a factor of 2.5 to 7.6 for $\ell \in \{150, 200, 300, 400, 500, 600\}$ with $r = 1$. For the detailed experimental result of PROPER with $r \in \{1,2\}$ refer to Table 4.4.

Table 4.4 – The experimental results for PROPER with different values of $r \in \{1,2\}$ and $\ell \in \{150, 200, 300, 400, 500, 600\}$. The results are the average of 10 pairwise alignments between species from Table 4.2.

r, ℓ	$\vert\pi\vert$	GOC (all)	GOC (exp.)	EC	ICS	S^3	LCSC
1,150	**5521.2**	**1388.562**	**371.460**	0.231	0.218	0.102	3345.7
1,200	5471.4	1284.853	351.285	0.249	0.235	0.112	3610.7
1,300	5432.9	1117.425	321.699	0.273	0.264	0.125	4081.9
1,400	5416.4	999.517	301.476	0.292	0.279	0.135	4397.3
1,500	5347.4	913.508	285.664	0.303	0.285	0.140	4533.4
1,600	5320.5	832.233	271.684	**0.309**	**0.295**	**0.145**	**4669.8**
2,150	3116.1	1224.103	299.481	0.114	0.185	0.060	1375.7
2,200	2900.3	1104.756	275.392	0.122	0.205	0.066	1433.6
2,300	2618.4	920.392	239.140	0.134	0.247	0.075	1566.2
2,400	2408.7	791.523	212.464	0.143	0.269	0.082	1617.8
2,500	2216.1	687.839	191.209	0.147	0.280	0.086	1602.8
2,600	2094.0	603.080	173.923	0.148	0.296	0.089	1596.8

Choosing smaller values of r reduces the required structural similarity for aligning a couple. This explains why the number of aligned couples for $r = 1$ is larger than for $r = 2$ in Table 4.5. Note that the `MapPercolation` algorithm, for a given value of r, cannot align nodes with degrees less than r. From Figure 4.9, which reports the degree distribution of different networks, we observe that there are many nodes with degree one, e.g, almost half of nodes for C. elegans and M. musculus. These nodes of degree one cannot be aligned with $r = 2$, and this is the reason we choose $r = 1$ for our experiments. In general, the value of r controls the strength of the structural evidence required before we decide to align a couple and a larger r makes errors less likely. We believe that by the increasing size of PPI networks over time, which consequently results in the decrease of number of low-degree nodes, a larger value of r will generate alignments with higher qualities.

Synthetic Networks

In this section, we compare algorithms based on their performance over synthetic networks. For this, we consider the high-confidence yeast Saccharomyces cerevisiae PPI network with 1004 nodes and 8323 edges [46, 165]; this network serves as our "ground-truth". For this experiment, a noisy version of the yeast network is generated by sampling each of its nodes

Table 4.5 – The average number of aligned couples when running (i) only the first step of PROPER (i.e., the `SeedGeneration` algorithm), and (ii,iii) PROPER with $r = \{1,2\}$ with different values of ℓ.

ℓ	`SeedGeneration`	$r = 2$	$r = 1$
150	2198.4	3116.1	5521.2
200	1875.6	2900.3	5471.4
300	1393.9	2618.4	5432.9
400	1083.1	2408.7	5416.4
500	861.0	2216.1	5347.4
600	696.4	2094.0	5320.5

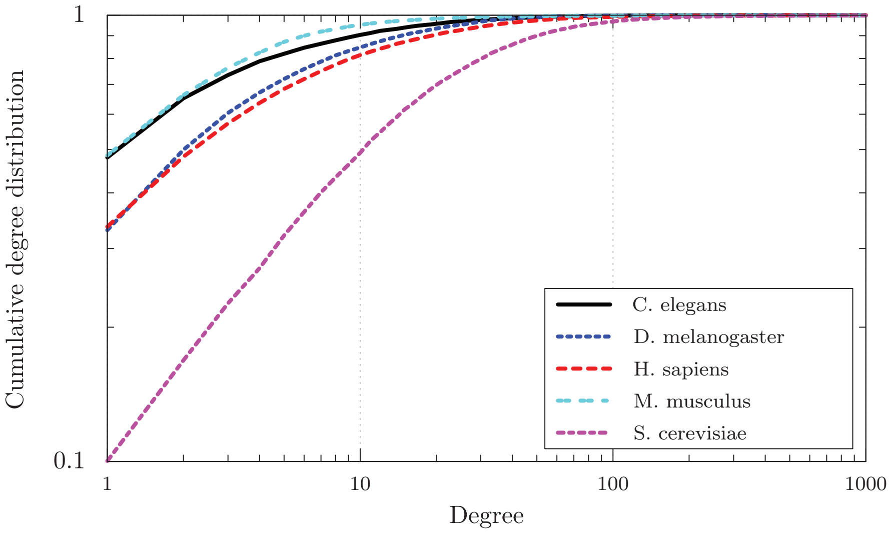


Figure 4.9 – Cumulative degree distribution for all the networks from Table 4.2.

and interactions with a probability s. Here, s controls the similarity of a sampled network with the original network, and we take $1-s$ as the “level of noise”. Also, the sequence similarity for a subset of randomly chosen proteins is provided as a side information. In this experiment, the ground-truth node mapping is known by design, which enables us to calculate NC and precision. Note that in order to account for the randomness of our experiments, we provide the average of 50 different alignments for each level of noise and available sequence similarity.

In the first experiment, we align the original network with five networks that are generated by different levels of noise $1-s \in \{5\%, 10\%, 15\%, 20\%, 25\%\}$. Also, the sequence similarity for 50% of randomly chosen proteins is provided. Figure 4.10 provides NC comparison over these synthetic networks for different levels of noise. From Figure 4.10, for example, we observe that PROPER aligns networks which are sampled with the noise level $1-s = 15\%$ with NC=0.86. Note that the average number of nodes for different noise levels (from 5% to 25%) is

946.48, 893.24, 832.54, 780.4 and 730.96, respectively. This means that PROPER correctly aligns $0.86 \cdot 832.54 \approx 716$ couples. Figure 4.11 compares algorithms based on precision. From the result of this experiment, we observe that for a low level of noise ($1-s = 5\%$) L-GRAAL has the best performance and PROPER comes second. With increasing level of noise, the performance of PROPER remains almost unaffected, whereas the quality of the other alignments decreases quite markedly.

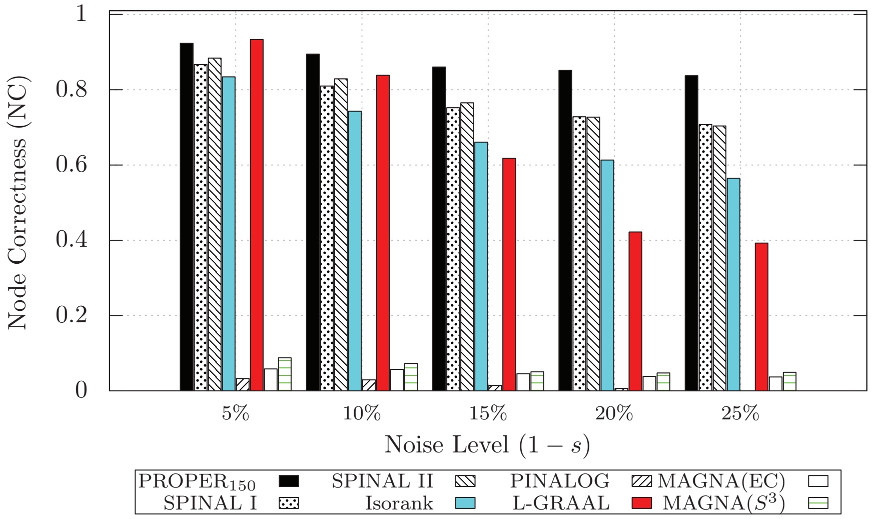


Figure 4.10 – Comparison of different global network-aligners over synthetic networks based on node correctness (NC). The sequence similarity for 50% of randomly chosen proteins is provided. For the PROPER algorithm, we set $r = 1$ and $\ell = 150$. The parameter α is 0.7.

In the second experiment, we investigate the effect of available sequence similarity on the performance of algorithms. We consider different amounts of available sequence similarity and fix the level of noise to $1-s = 20\%$. Figure 4.12 compares algorithms when the sequence similarities for $20\%, 30\%, 40\%, 50\%, 60\%$ and 70% of randomly chosen proteins are provided. Figure 4.13 compares algorithms based on precision. We observe that PROPER outperform the other algorithms for different amounts of available sequence similarity.

These two experiments confirm the success of the PROPER algorithm in aligning synthetic networks and its robustness to high levels of noise.

4.3.3 Aligning Biological Pathways

In this section, we compare algorithms based on their performance in aligning biological pathways. We use $\alpha = 0.7$ for SPINAL, IsoRank, MAGNA and L-GRAAL, and $r = 1, l = 150$ for PROPER. We use the measures introduced in Section 4.2.2. For our comparisons, we consider the alignment of H. sapiens with the other four species from Table 4.2. We know that there are several proteins that belong to more than one pathway, because some proteins could be

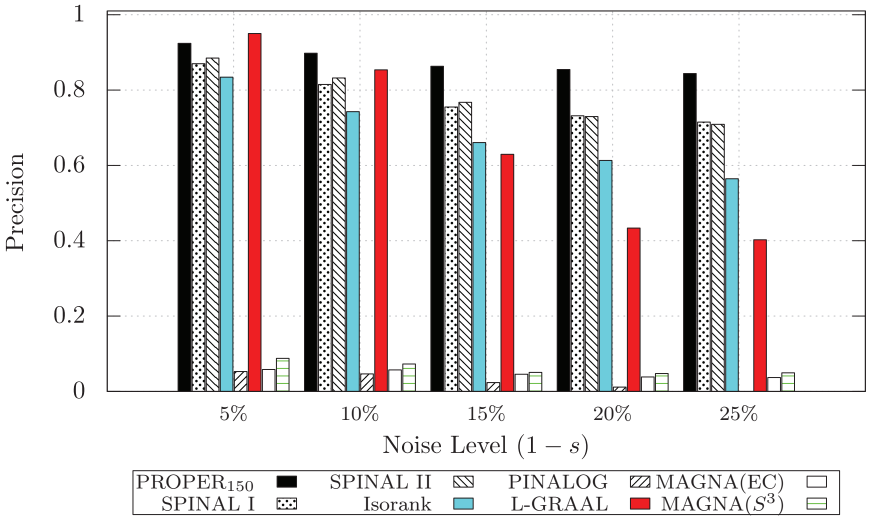


Figure 4.11 – Comparison of different global network-aligners over synthetic networks based on precision. The sequence similarity for 50% of randomly chosen proteins is provided. For the PROPER algorithm, we set $r = 1$ and $\ell = 150$. The parameter α is 0.7.

involved in different biological processes. For this reason, along the results for all the pathways, we consider a subset of non-overlapping pathways for each pair of species. Table 4.6 reports the number of common KEGG pathways between different pairs of species, where we consider (i) all the pathways, (ii) pathways with at least $\delta = 4$ interactions in each of the species, and (iii) a subset of non-overlapping pathways.

Table 4.6 – Number of common KEGG pathways between different pairs of species.

Pair of species	#PW	#PW($\delta = 4$)	#PW (no-overlap)
hs-ce	116	19	37
hs-dm	122	31	40
hs-mm	283	152	49
hs-sc	98	32	34

For the first experiment, we do not consider the topological similarities of aligned pathways. The result for alignments of pathways from different algorithms is provided in Table 4.7. We observe that PROPER outperforms the other algorithms in terms of accuracy. In the second experiment, for each algorithm we consider only the pathways with at least $\delta = 4$ conserved interactions across species (i.e., $\Delta_{\pi,i} \geq 4$). Table 4.8 provides the results for this case. Again, we observe that the PROPER algorithm outperforms the other algorithms, i.e., on average it aligns more pathways with a higher accuracy. MAGNA++ performs very poorly in this experiment and we omit it from Table 4.8.

For many pathways, the PROPER algorithm, compared to other algorithms, returns alignments

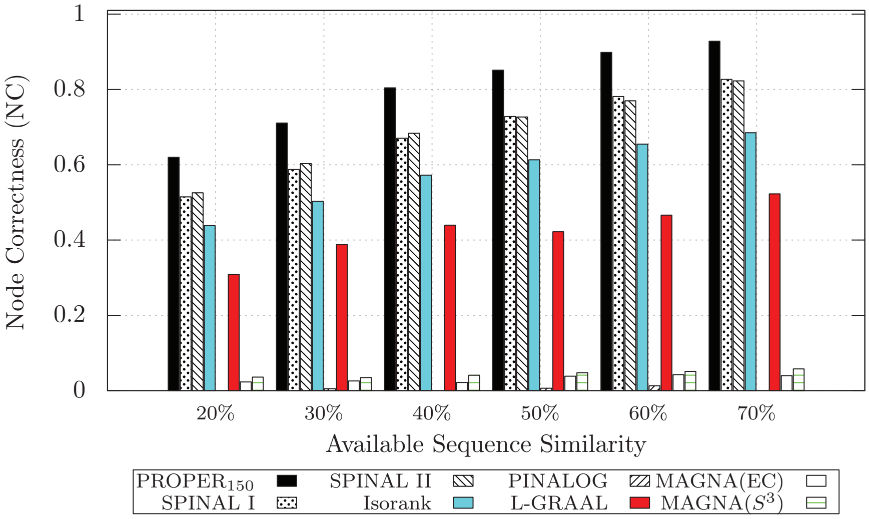


Figure 4.12 – Comparison of different global network-aligners over synthetic networks based on node correctness (NC). The level of noise is set to $1 - s = 20\%$. For the PROPER algorithm, we set $r = 1$ and $\ell = 150$. The parameter α is 0.7.

Table 4.7 – Comparison of algorithms based on aligning biological pathways. This table reports the average value of $\overline{acc_\pi}$ for pairwise alignments between Home sapiens and the four other species from Table 4.2.

Algorithms	$\overline{acc_\pi}$	$\overline{acc_\pi}$ (no-overlap)
PROPER	**0.471**	**0.442**
SPINAL I	0.447	0.426
SPINAL II	0.115	0.134
PINALOG	0.409	0.397
L-GRAAL	0.232	0.218
MAGNA++(S^3)	0.016	0.020
MAGNA++(EC)	0.017	0.020
IsoRank	0.202	0.195

with a larger portion of connected conserved subgraphs. For example, Figure 4.14 shows the connected conserved subgraph of pathways hsa05200 and mmu05200 between human and mouse.[4] The connected subgraph of this pathway has 37 nodes and 42 edges, which is larger than alignments by the other algorithms (see Appendix 4.B).

---

[4] The connected subgraph of this pathway has 37 nodes and 42 edges, which is larger than alignments by the other algorithms (see Appendix 4.B).

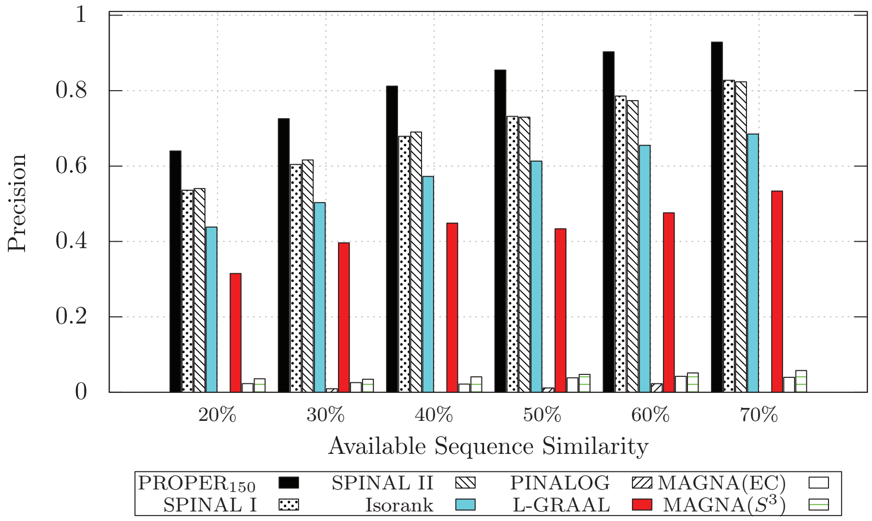


Figure 4.13 – Comparison of different global network-aligners over synthetic networks based on precision. The level of noise is set to $1 - s = 20\%$. For the PROPER algorithm, we set $r = 1$ and $\ell = 150$. The parameter α is 0.7.

Table 4.8 – Comparison of algorithms based on pathway alignment measures for $\delta = 4$ (i.e., $\Delta_{\pi,i} \geq 4$). This table reports the average value of measures for pairwise alignments between Home sapiens and the four other species from Table 4.2.

Algorithms	#FPW	$\overline{acc_{\pi,\delta}}$	$recall_{\pi}$
PROPER	**42.5**	**0.585**	**0.584**
SPINAL I	38.75	0.554	0.536
SPINAL II	9.0	0.223	0.102
PINALOG	39.75	0.497	0.547
L-GRAAL	25.5	0.320	0.235
IsoRank	18.5	0.356	0.225

4.3.4 Execution Time

A fast and scalable alignment algorithm is needed with the growing size of PPI networks. One of the key features of the PROPER algorithm is its low computational complexity and scalability. PROPER is able to align synthesis networks with millions of nodes in less than a hour. In fact, the complexity of our algorithm is $O\left((|E_1| + |E_2|) \min(D_1, D_2)\right)$, where $D_{1,2}$ are the maximum degrees in the two networks. Table 4.9 provides the total execution time of algorithms for 10 pairwise alignments between the five species from Table 4.2. All computations are done on the same Linux machine with 16 GB of memory and 8 Intel Xeon E3-1270 CPUs working at clock speeds 3.50 GHz. We observe that PROPER runs much faster than the other algorithms.

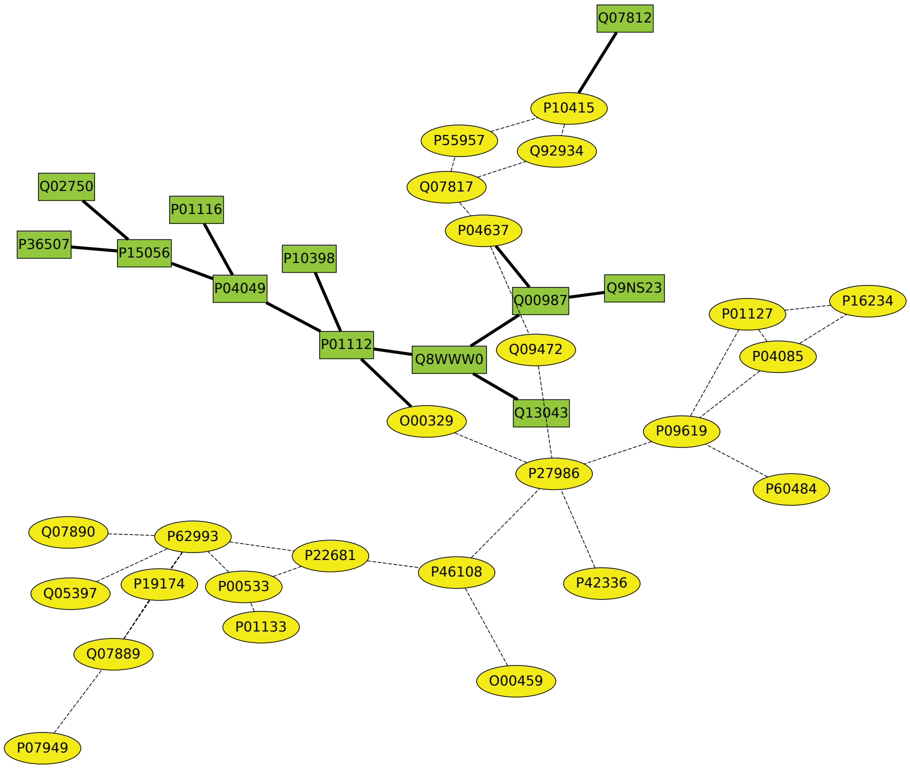


Figure 4.14 – The connected subgraph of hsa05200 and mmu05200 pathways in human and mouse from the PROPER algorithm, with conserved interactions in both species. This connected subgraph has 37 nodes and 42 edges. The PINALOG algorithm returns the second largest connected subgraph. The rectangular nodes and solid edges are the proteins and interactions among them that are found only by the PROPER algorithm.

Table 4.9 – The total execution time of algorithms for 10 pairwise alignments between the five species from Table 4.2.

Aligner	Time
PROPER	317 seconds
L-GRAAL	4 hours and 2 minutes
MAGNA++(S^3)	7 hours and 47 minutes
MAGNA++(EC)	7 hours and 41 minutes
PINALOG	2 days, 5 hours and 26 minutes
SPINAL I	10 hours and 51 minutes
SPINAL II	11 hours and 56 minutes
IsoRank	12 hours and 43 minutes

4.4 Discussion

The purpose of network-alignment algorithms is to find functional and structural similarities between PPI networks of different species [59]. Most of the works in the literature model global network-alignment as an optimization problem over the convex combination of sequence and structural similarities between two networks [112, 176, 207]. This class of algorithms aims to maximize a cost function in order to increase the following two quantities simultaneously: (i) the pairwise similarities between aligned proteins (e.g., by maximizing the summation over all the BLAST similarities of aligned proteins), and (ii) the structural similarity between the two graphs, (e.g., by maximizing the conserved PPIs under the alignment) [43].

It appears that this particular formulation of the optimization problem precludes these algorithms from making good alignments by using both similarities jointly [43]. For example, in Chapter 6 we show that in the IsoRank algorithm for the structure-only ($\alpha = 1$) alignment, the similarity of two nodes is only a function of their degrees. Our results in that chapter explicate the poor performance of IsoRank in finding alignments with good structural similarities. Also, our experimental results confirm the trade-off between structural and functional similarities in most of the state-of-the-art network-alignment algorithms. We observe that each of the five algorithms evaluated here, namely L-GRAAL, MAGNA++, IsoRank, PINALOG and SPINAL, covers only a small portion of the trade-off frontier (see Figures 4.3 and 4.5). In summary, we believe that these observations make it necessary to study the PPI network alignment problem under rigorous mathematical models.

The PROPER algorithm, in comparison, shows less compromise between the functional similarities among aligned proteins and the topological similarity. Figures 4.3, 4.4 and 4.5 show that our algorithm sweeps the frontier (i.e., has the best trade-off between both measures) more robustly than the other algorithms. In addition, large conserved subgraphs with the same function are aligned with PROPER. The PROPER algorithm not only aligns proteins and their corresponding interactions from two different species better than other algorithms, it also aligns the conserved pathways between the species with higher accuracy. This shows that instead of finding conserved single pairwise PPIs, PROPER represents a more biologically realistic performance by detecting sub-networks of conserved interactions from pathways with the same function among species.

In addition to its superior accuracy, PROPER performs better in terms of memory usage and speed, because the alignment process of PROPER is a very simple local propagation method.

4.4.1 Why the PROPER Algorithm?

In the following, we explicate the two reasons PROPER performs well in terms of all the cost functions considered.

The first reason is that a high BLAST bit-score is a reliable indicator of a match, whereas a low BLAST bit-score is very unreliable for many functional characteristics [52]. As a consequence, rather than optimizing a convex combination of functional similarity with structural similarity, it is advantageous to ascribe high confidence to the sparse set of high-BLAST couples, and to completely ignore low BLAST bit-scores. This is what PROPER does, by generating an initial seed-set of high BLAST couples, and then by propagating outwards from this seed set as a purely structure-driven process. Note that as the PGM class of algorithms are shown to be robust against noise in the seed set [99], PROPER is not sensitive to the sequence similarity threshold ℓ for aligning new couples of proteins.

The second reason is more speculative and has to do with the statistical structure of the two networks being matched. Computational biology postulates evolutionary models to explain the difference between PPI networks. Studies have identified gene duplication and the gain or loss of genes and their interactions as the key evolutionary events in forming biological networks [126, 161, 187]. Several evolutionary models for regulatory networks and protein–interaction networks have been introduced based on these observed evolutionary processes [28, 162, 208].

Percolation-based methods for network alignment are well-suited for network pairs whose structural differences arise from the random deletions of nodes and edges. Specifically, in Chapters 2 and 3, we define the $G(n,p;t,s)$ random bigraph model for generating two correlated networks $G_{1,2}$ that rely on node and edge sampling processes. The two parameters t and s control the node and edge similarity of the generated graphs. Although the analysis in Chapter 3 is for a different algorithm within the PGM class, we believe the main concepts carry over to PROPER.

More specifically, for the sake of simplicity, we assume that the evolutionary process can only delete proteins and interactions among proteins. We call this model $Evolve(G,t,s)$, where we postulate an ancestor network $G(V,E)$, from which both observable networks $G_{1,2}$ derive through independent evolutionary processes. The parameter t is the probability that a protein in G survives in $G_{1,2}$ (proteins are lost with probability $1-t$); and parameter s is the probability that an interaction between proteins, i.e., an edge in G, survives in $G_{1,2}$ (interactions are lost with probability $1-s$). With the additional assumption that the ancestor network G is an Erdős-Rényi [58] random graph (i.e., a $G(n,p)$ graph with n nodes, where each of the $\binom{n}{2}$ possible edges occurs independently with probability $0<p<1$) this evolutionary model is equivalent to the $G(n,p;t,s)$ model studied in the literature [49, 100, 150].

By using this model, conditions for the success of PGM-based network alignment have been established. In particular, a sharp phase-transition in terms of the seed-set size have been shown: If the seed-set size is above some threshold (which depends on the network parameters n, p, t, and s), PGM-based alignment can correctly match, with high probability, almost all the node couples by using a purely structural process. Also, from the result of [100], we know that under a similar random bigraph model, the correct alignment maximizes the number of conserved interactions between the two networks. This simple parsimonious evolutionary model provides guarantees for the performance of the PROPER algorithm over random graphs similar to [99]. Note that, in practice, these algorithms are able to successfully align large real-networks, as well as many types of random graphs. In conclusion, it seems that mapping a (small) subset of nodes through a seed-generation step and matching the rest, by using only structure of the two graphs, works very well under an evolutionary model.

4.5 Summary

In this chapter, we have introduced a new global pairwise-network alignment algorithm called PROPER. We have compared our algorithm with the state-of-the-art algorithms. We have shown that PROPER outperforms the other algorithms in both accuracy and speed. Also, we have shown that the PROPER algorithm can detect large conserved subnetworks between species. The PROPER algorithm is publicly available at http://proper.epfl.ch.

Our results suggest that network-evolutionary models could be beneficial in designing network-alignment algorithms. We believe that, for future work, considering a model that also takes into account gene duplication, network motifs, clustering within networks and modularity of biological networks (e.g., [141]) would increase the accuracy of global network-alignments. Finally, to find biological pathways and protein complexes using the PROPER algorithm, the next step would be to design methods that can detect sub-networks as potential pathways or complexes (similar to the method used in [103, 104]).

Appendix

4.A IsoBase: Experimental Results

Isobase is a collection of PPI networks of five major eukaryotic species [2, 148]. This database also contains information about (i) gene ontology (GO) and KEGG categories associated to the proteins, and (ii) functionally related orthologs. In addition, for complementary comparisons, we use the experimentally verified GO terms from [10]. We consider a subset of four species from IsoBase. As the PPI network of M. musculus is very sparse (with average degree 1.867) we omit if from our comparisons. Table 4.10 represents the name of these species and the number of proteins and interactions in their PPI networks.

Table 4.10 – PPI networks of four eukaryotic species from IsoBase [148].

species	#nodes	#edges	Avg. deg.
C. elegans	2974	4827	3.246
D. melanogaster	7387	24937	6.752
H. sapiens	10296	54654	10.617
S. cerevisiae	5523	82656	29.932

In this appendix, we compare different alignment algorithms. The alignments are done over six pairs of species: ce-dm, ce-hs, ce-sc, dm-hs, dm-sc and hs-sc (for abbreviations and information about the networks refer to Table 4.10).

Figure 4.15 compares algorithms based on the average of EC versus GOC score for all the possible 6 pairwise alignments between the species. The complementary result for experimentally verified GO terms is shown in Figure 4.16.

Kyoto Encyclopedia of Genes and Genomes (KEGG) database [95] provides another classification of proteins. We define KEGG consistency similar to the GOC using KEGG categories. Figure 4.17 compares algorithms based on average KEGG consistency versus average LCSC. We observe that proper finds alignments with high KEGG consistency and LCSC scores.

Figures 4.18, 4.19, 4.20, 4.21, 4.22, 4.23 and 4.24 report the detailed results of algorithms for each one of the alignments over all pairs of species.

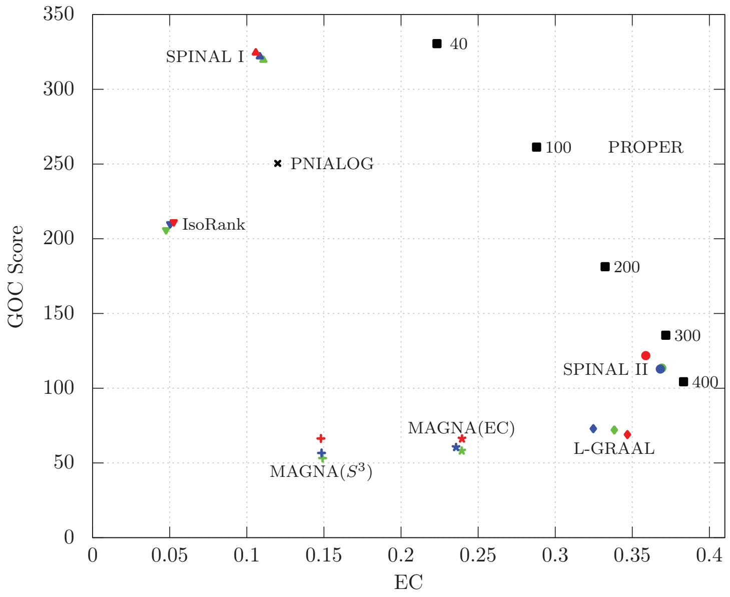


Figure 4.15 – Comparison of different global network-aligners based on the average GO consistency vs. average EC. For the PROPER algorithm, we set $r = 1$ and each point corresponds to a different value of ℓ. The red, blue and green points correspond to the parameters $\alpha = 0.3, 0.5$ and 0.7, respectively.

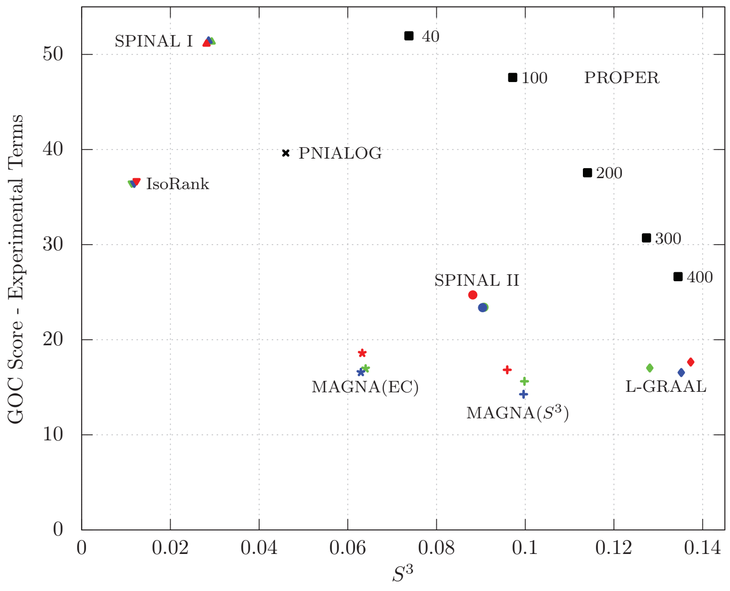


Figure 4.16 – Comparison of different global network-aligners based on the average GO consistency vs. average S^3. For the PROPER algorithm, we set $r = 1$ and each point corresponds to a different value of ℓ. The red, blue and green points correspond to the parameters $\alpha = 0.3, 0.5$ and 0.7, respectively.

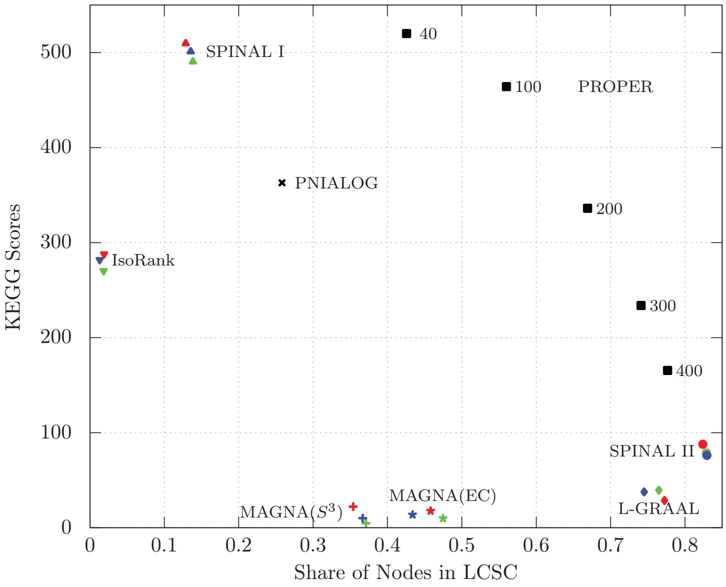


Figure 4.17 – Comparison of different global network-aligners based on the average LCSC vs. average KEGG consistency. For the PROPER algorithm, we set $r = 1$ and each point corresponds to a different value of ℓ. The red, blue and green points correspond to the parameters $\alpha = 0.3, 0.5$ and 0.7, respectively.

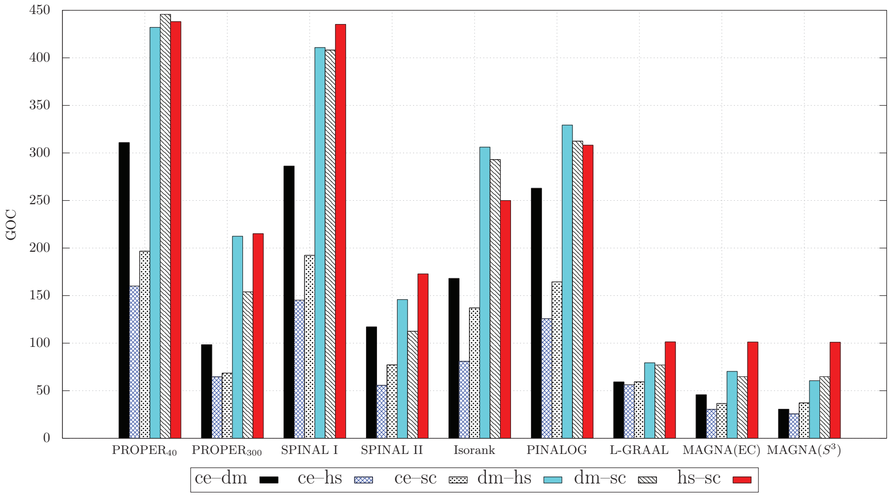


Figure 4.18 – GO Consistency scores for each alignment of the IsoBase PPI-networks. For the PROPER algorithm, we set $r = 1$ and $\ell \in \{40, 300\}$. The parameter α is 0.7.

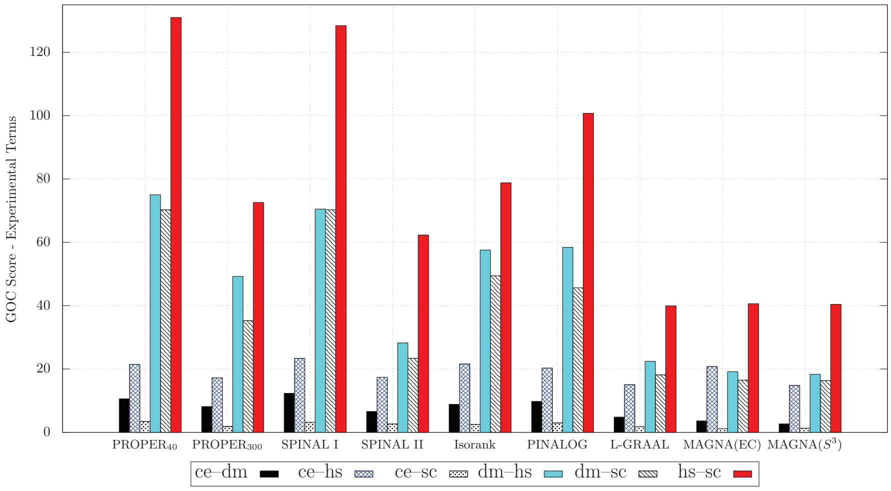


Figure 4.19 – GO Consistency scores (considering only experimental terms) for each alignment of the IsoBase PPI-networks. For the PROPER algorithm, we set $r = 1$ and $\ell \in \{40, 300\}$. The parameter α is 0.7.

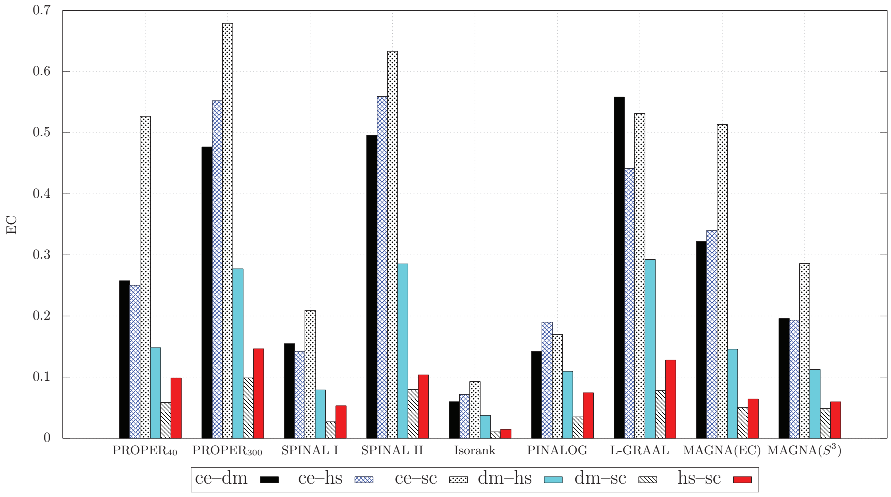


Figure 4.20 – Edge correctness (EC) scores for each of the pairwise alignment for the IsoBase PPI-networks. For the PROPER algorithm, we set $r = 1$ and $\ell \in \{40, 300\}$. The parameter α is 0.7.

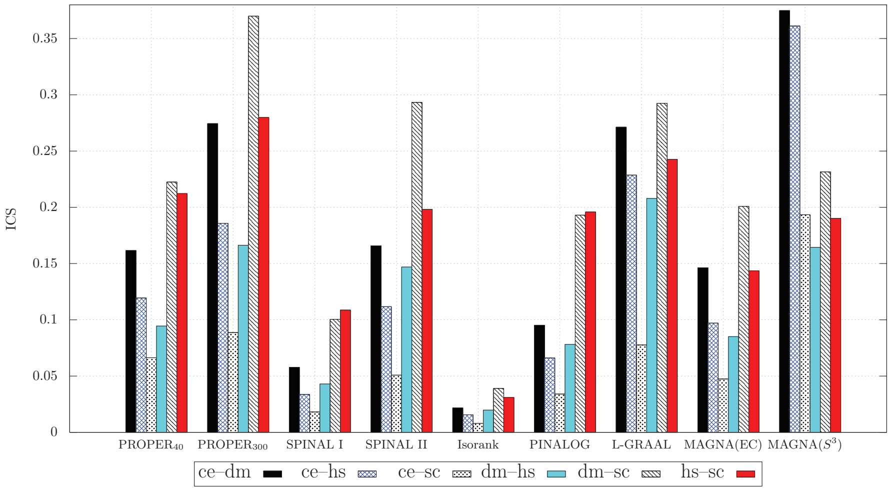


Figure 4.21 – Induced conserved structure (ICS) scores for each of the pairwise alignment for the IsoBase PPI-networks. For the PROPER algorithm, we set $r = 1$ and $\ell \in \{40, 300\}$. The parameter α is 0.7.

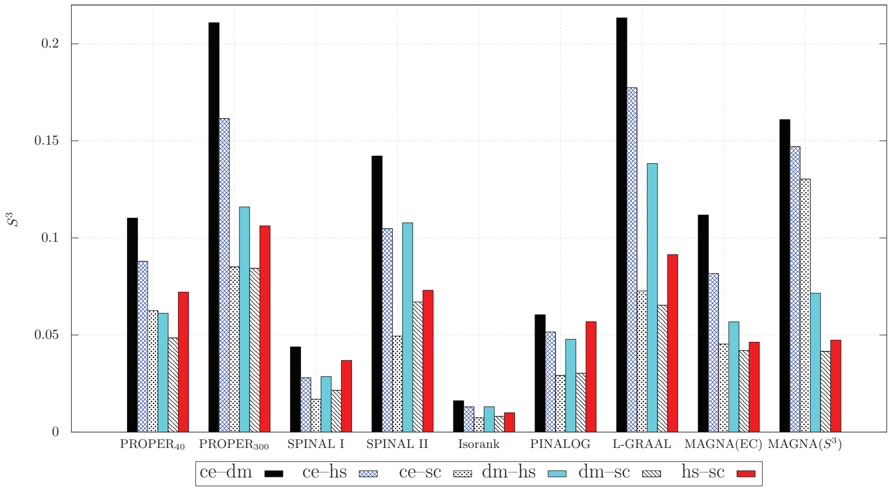


Figure 4.22 – Symmetric substructure score (S^3) scores for each of the pairwise alignment for the IsoBase PPI-networks. For the PROPER algorithm, we set $r = 1$ and $\ell \in \{40, 300\}$. The parameter α is 0.7.

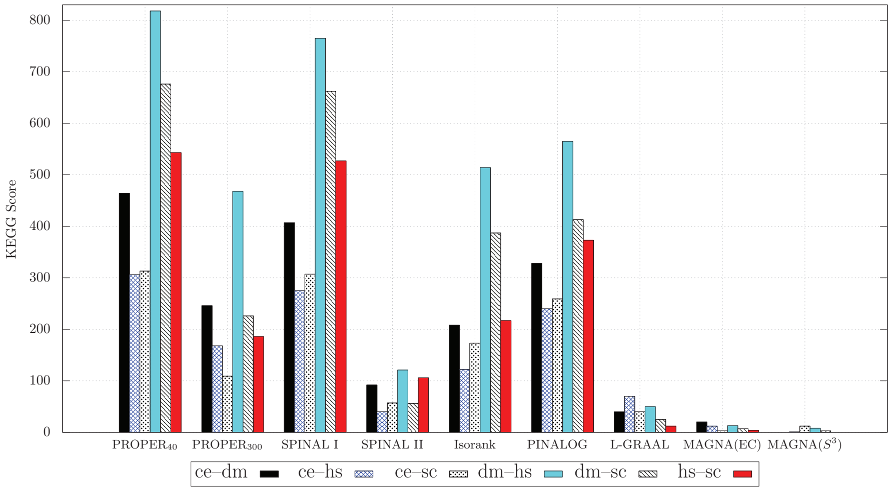


Figure 4.23 – KEGG scores for each alignment of the IsoBase PPI-networks. For the PROPER algorithm, we set $r = 1$ and $\ell \in \{40, 300\}$. The parameter α is 0.7.

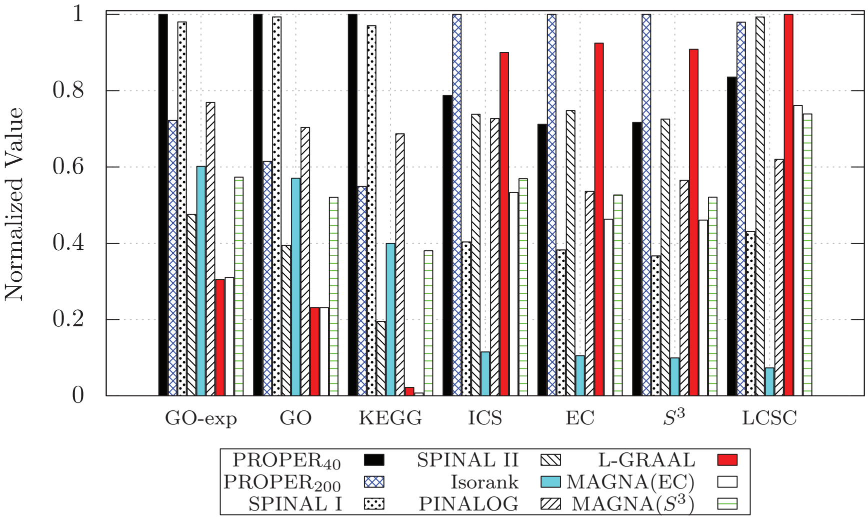


Figure 4.24 – Comparison of different global network-aligners on aligning H. sapiens and S. cerevisiae based on six different measures. For the PROPER algorithm, we set $r = 1$ and $\ell \in \{40, 200\}$. The parameter α is 0.7.

4.B Pathways: Experimental Results

In this appendix, we provide the results of our experiments for aligning biological pathways in more details. The results for each one of the algorithms are reported in Tables 4.11, 4.12, 4.13, 4.14, 4.15 and 4.16.

Table 4.11 – The PROPER algorithm: pathways with at least four conserved edges ($\delta = 4$) in the intersection graph G_0.

species	#PW$_\delta$	#FPW$_\delta$	$\overline{acc_{\pi,\delta}}$	$recall_\pi$
ce-hs	19	8	0.455	0.421
dm-hs	31	12	0.611	0.387
hs-mm	152	128	0.788	0.842
sc-hs	32	22	0.486	0.688
Average	58.5	42.5	0.585	0.584

Table 4.12 – SPINAL I algorithm: pathways with at least four conserved edges ($\delta = 4$) in the intersection graph G_0.

species	#PW$_\delta$	#FPW$_\delta$	$\overline{acc_{\pi,\delta}}$	$recall_\pi$
ce-hs	19	7	0.472	0.368
dm-hs	31	11	0.564	0.355
hs-mm	152	116	0.711	0.763
sc-hs	32	21	0.469	0.656
Average	58.5	38.75	0.554	0.536

Table 4.13 – SPINAL II algorithm: pathways with at least four conserved edges ($\delta = 4$) in the intersection graph G_0.

species	#PW$_\delta$	#FPW$_\delta$	$\overline{acc_{\pi,\delta}}$	$recall_\pi$
ce-hs	19	1	0.301	0.053
dm-hs	31	1	0.081	0.032
hs-mm	152	30	0.261	0.197
sc-hs	32	4	0.247	0.125
Average	58.5	9	0.223	0.102

The pathways hsa05200 and mmu05200 are the pathways in the class cancer Homo sapiens (human). The largest connected subgraphs of hsa05200 and mmu05200 pathways in the intersection graph G_0 from all the algorithms are shown in Figures 4.14, 4.26, 4.27, 4.28, 4.29 and 4.30. Next, we consider another pathway. The largest connected subgraphs of hsa04510 and mmu04510 pathways in the intersection graph G_0 from all the algorithms are shown in Figures 4.31, 4.32, 4.33, 4.34, 4.35 and 4.36. We observe that, again, PROPER returns alignments with a larger portion of connected conserved subgraphs compared to other algorithms.

Table 4.14 – PINALOG algorithm: pathways with at least four conserved edges ($\delta = 4$) in the intersection graph G_0.

species	#PW$_\delta$	#FPW$_\delta$	$\overline{acc_{\pi,\delta}}$	$recall_\pi$
ce-hs	19	8	0.409	0.421
dm-hs	31	9	0.459	0.290
hs-mm	152	120	0.629	0.789
sc-hs	32	22	0.492	0.687
Average	58.5	39.75	0.497	0.547

Table 4.15 – L-GRAAL algorithm: pathways with at least four conserved edges ($\delta = 4$) in the intersection graph G_0.

species	#PW$_\delta$	#FPW$_\delta$	$\overline{acc_{\pi,\delta}}$	$recall_\pi$
ce-hs	19	1	0.211	0.053
dm-hs	31	1	0.312	0.032
hs-mm	152	92	0.414	0.605
sc-hs	32	8	0.344	0.250
Average	58.5	25.5	0.320	0.235

Table 4.16 – IsoRank algorithm: pathways with at least four edges conserved ($\delta = 4$) in the intersection graph G_0.

species	#PW$_\delta$	#FPW$_\delta$	$\overline{acc_{\pi,\delta}}$	$recall_\pi$
ce-hs	19	3	0.341	0.158
dm-hs	31	5	0.442	0.161
hs-mm	152	60	0.407	0.395
sc-hs	32	6	0.234	0.186
Average	58.5	18.5	0.356	0.225

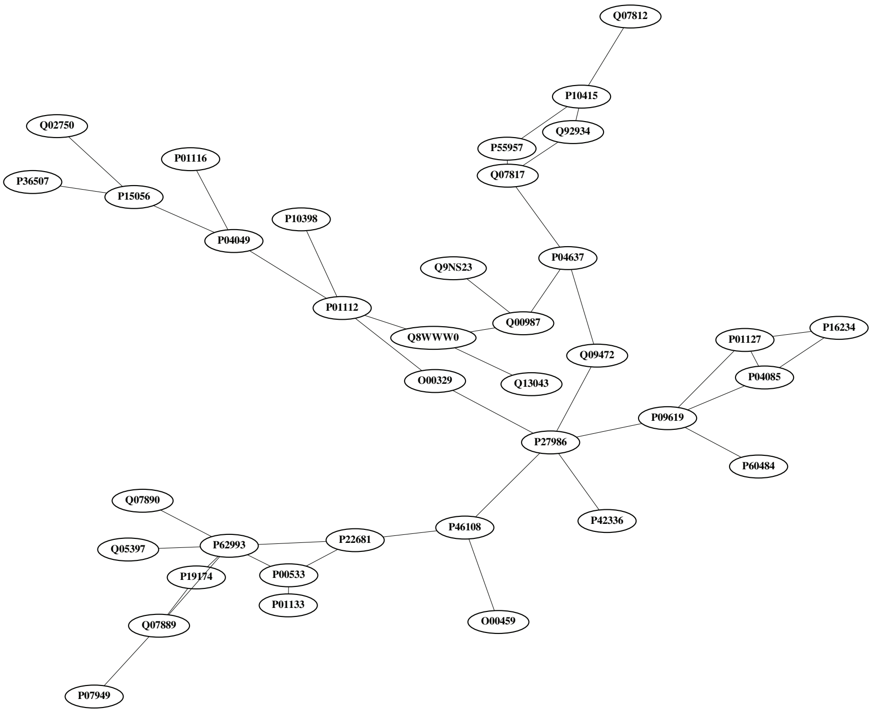


Figure 4.25 – PROPER: pathways hsa05200 and mmu05200. Subgraph is preserved in hs and mm. This connected subgraph of the pathway has 37 nodes and 42 edges.

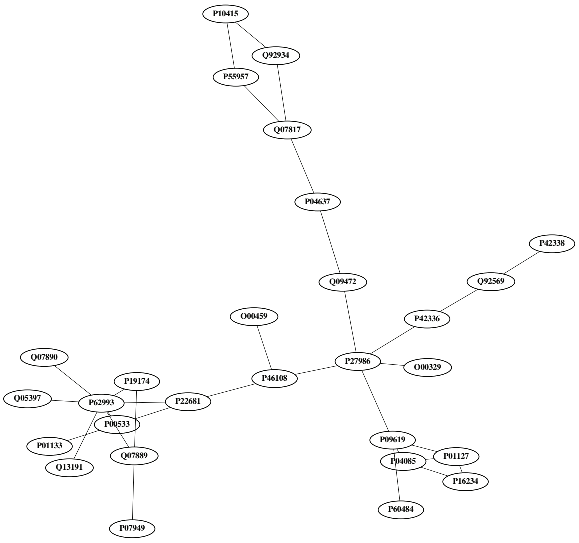


Figure 4.26 – PINALOG: pathways hsa05200 and mmu05200. Subgraph is preserved in hs and mm. This connected subgraph of the pathway has 28 nodes and 32 edges.

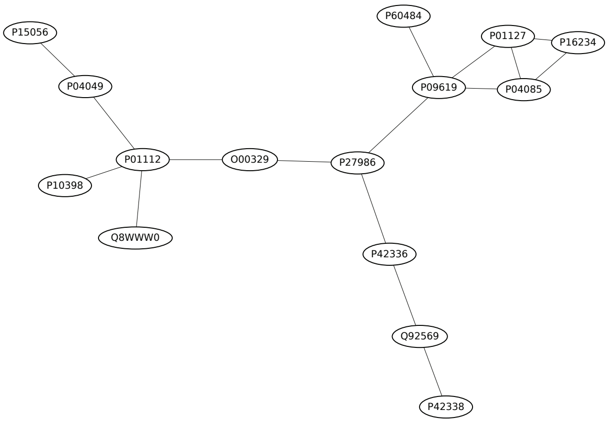


Figure 4.27 – L-GRAAL: pathways hsa05200 and mmu05200. Subgraph is preserved in hs and mm. This connected subgraph of the pathway has 15 nodes and 16 edges.

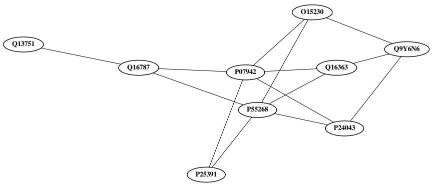


Figure 4.28 – SPINAL I: pathways hsa05200 and mmu05200. Subgraph is preserved in hs and mm. This connected subgraph of the pathway has 9 nodes and 14 edges.

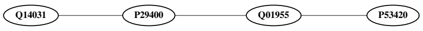


Figure 4.29 – SPINAL II: pathways hsa05200 and mmu05200. Subgraph is preserved in hs and mm. This connected subgraph of the pathway has 4 nodes and 3 edges.

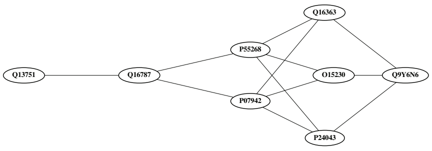


Figure 4.30 – IsoRank graph matching: pathways hsa05200 and mmu05200. Subgraph is preserved in hs and mm. This connected subgraph of the pathway has 8 nodes and 12 edges.

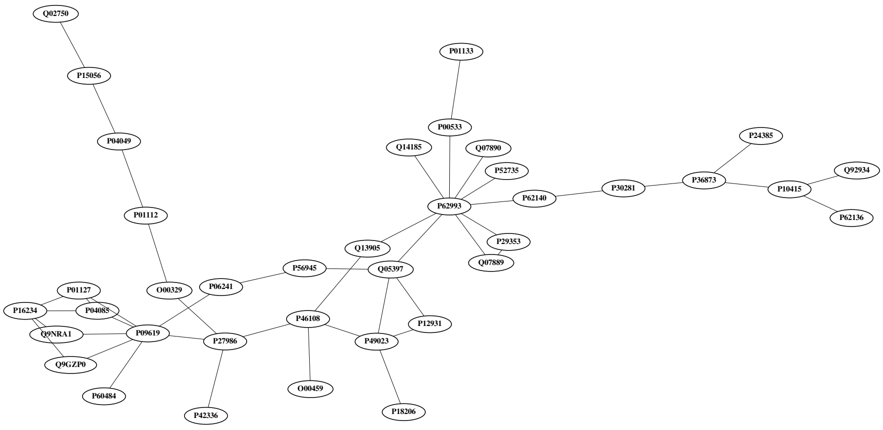


Figure 4.31 – PROPER: pathways hsa04510 and mmu04510. Subgraph is preserved in hs and mm. This connected subgraph of the pathway has 38 nodes and 45 edges.

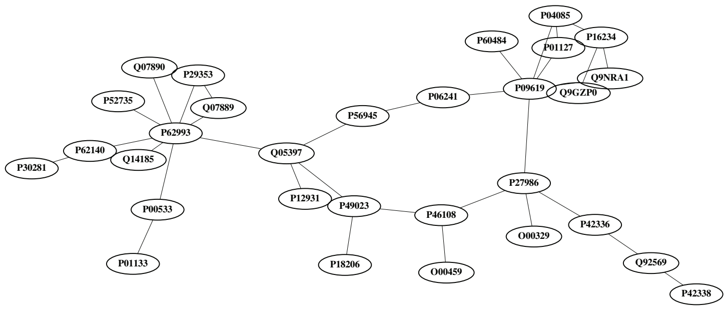


Figure 4.32 – PINALOG: pathways hsa04510 and mmu04510. Subgraph is preserved in hs and mm. This connected subgraph of the pathway has 30 nodes and 36 edges.

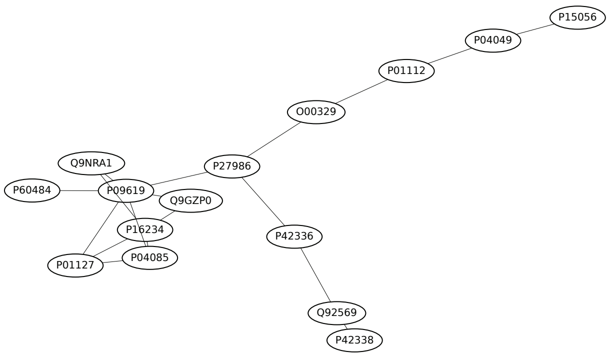


Figure 4.33 – L-GRAAL: pathways hsa04510and mmu04510. Subgraph is preserved in hs and mm. This connected subgraph of the pathway has 15 nodes and 18 edges.

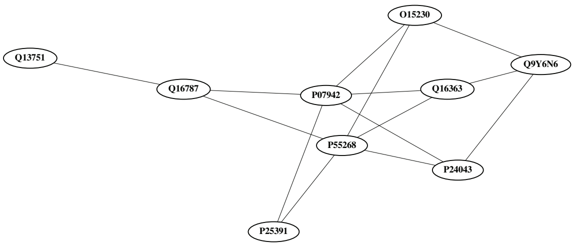


Figure 4.34 – SPINAL I: pathways hsa04510and mmu04510. Subgraph is preserved in hs and mm. This connected subgraph of the pathway has 9 nodes and 14 edges.

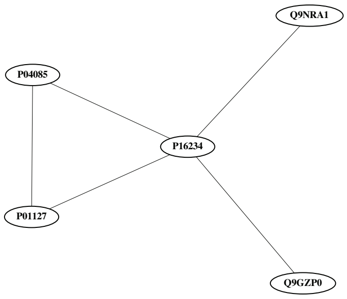


Figure 4.35 – SPINAL II: pathways hsa04510 and mmu04510. Subgraph is preserved in hs and mm. This connected subgraph of the pathway has 5 nodes and 5 edges.

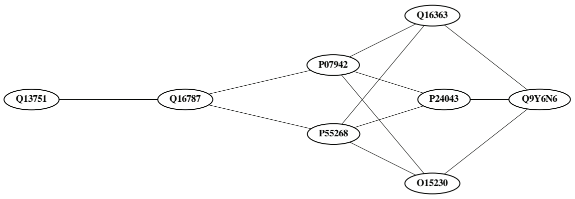


Figure 4.36 – IsoRank: pathways hsa04510 and mmu04510. Subgraph is preserved in hs and mm. This connected subgraph of the pathway has 8 nodes and 12 edges.

5 Global Multiple-Network Alignment

The advance of high-throughput methods for detecting protein interactions has made the PPI networks of many organisms available to researchers. With the huge amounts of biological network data and increasing number of known PPI networks, the problem of multiple-network alignment (MNA) is gaining more attention in the systems-biology studies. We believe that a good MNA algorithm leads us to a deeper understanding of biological networks (compared to pairwise-network alignment methods), because they capture the knowledge related to several species.

MNA methods produce alignments consisting of aligned clusters (or tuples) with nodes from several networks. MNA algorithms are classified into two categories of one-to-one and many-to-many algorithms. In the first category, each node from a network can be aligned to at most one node from another network. In the many-to-many category, one or several nodes from a network can be aligned with one or several nodes from another network.

Several MNA algorithms were proposed in past few years: NetworkBlast-M, a many-to-many local MNA algorithm, begins the alignment process with a set of high-scoring sub-networks (as seeds). It then expands them in a greedy fashion [94, 173]. Graemlin [62] is a local MNA algorithm that finds alignments by successively performing alignments between pairs of networks, by using information from their phylogenetic relationship. IsoRankN [119] is the first global MNA algorithm that uses both pairwise sequence similarities and network topology, to generate many-to-many alignments. SMETANA [163], another many-to-many global MNA algorithm, tries to find aligned node-clusters by using a semi-Markov random-walk model. This random-walk model is used for computing pairwise similarity scores. CSRW [87], a modified version of SMETANA, uses a context-sensitive random-walk model. NetCoffee [80] uses a triplet approach, similar to T-Coffee [144], to produce a one-to-one global alignment. GEDEVO-M [82] is a heuristic one-to-one global MNA algorithm that uses only topological information. To generate multiple alignments, GEDEVO-M minimizes a generalized graph edit distance measure. NH [156] is a many-to-many global MNA heuristic algorithm that uses only network structures. Alkan and Erten [12] designed a many-to-many global heuristic method based on a backbone extraction and merge strategy (BEAMS). The BEAMS algorithm, given k

networks, constructs a k-partite pairwise similarity graph. It then builds an alignment, in a greedy manner, by finding a set of disjoint cliques over the k-partite graph. Gligorijević et al. [66] introduced FUSE, another one-to-one global MNA algorithm. FUSE first applies a non-negative matrix tri-factorization method to compute pairwise scores from protein-sequence similarities and network structure. Then it uses an approximate k-partite matching algorithm to produce the final alignment.

In this chapter, we introduce a new scalable and accurate one-to-one global multiple-network alignment algorithm called MPROPER. This algorithm is an extension of the PROPER algorithm (see Chapter 4). MPROPER has two main steps. In the first step (`SeedGeneration`), to generate an initial set of seed clusters (or tuples), it uses only protein sequence similarities. In the second step (`MultiplePercolation`), to align remaining unmatched nodes, it uses network structures and the seed tuples generated from the first step. We compare MPROPER with several state-of-the-art algorithms. We show that MPROPER outperforms the other algorithms, with respect to different evaluation criteria. Also, we provide experimental evidence for the good performance of the `SeedGeneration` algorithm. Finally, we study the performance of the `MultiplePercolation` algorithm, by using a stochastic graph-sampling model.

5.1 Problem Definition

The goal of a one-to-one global MNA algorithm is to find an alignment between proteins from k different species (networks), where a protein from a species can be aligned to at most one unique protein from another species, in a way such that (i) the clusters (or tuples) of aligned proteins have similar biological functions, and (ii) the aligned networks are structurally similar, e.g., they share many conserved interactions among different clusters. To be more precise, a one-to-one global alignment π between k networks $G_i = (V_i, E_i), 1 \le i \le k$, is the partition of all (or most of) the nodes $V = \cup_{i=i}^{k} V_i$ into clusters $\{T_1, T_2, \cdots, T_{|\pi|}\}$ of size at least two (i.e., they should have nodes from at least two networks), where a cluster T_i has at most one node from each network. Note that any two clusters T_i and T_j are disjoint, i.e., $T_i \cap T_j = \emptyset$.

In the global MNA problem, to align the proteins from $k > 2$ species, PPI-networks and protein sequence similarities are used as inputs. Formally, we are given the PPI networks of k different species: the networks are represented by $G_1(V_1, E_1), G_2(V_2, E_2), \cdots, G_k(V_k, E_k)$. Also, the BLAST sequence similarity of the couples of proteins in all the $\binom{k}{2}$ pairs of species is provided as additional side information. Let $\mathcal{S}_{i,j}$ denote the set of BLAST bit-score sequence similarity of the couples in $V_i \times V_j$, i.e.,

$$\mathcal{S}_{i,j} = \{([u, v], BlastBit(u, v)) \,|\, [u, v] \in V_i \times V_j\}.$$

In the next section, we introduce our proposed global MNA algorithm.

5.2 The MPROPER Algorithm

In this section, we propose the MPROPER algorithm for aligning multiple PPI-networks. The MPROPER algorithm has two main steps:

- In the first step, it uses only the sequence similarities to find a set of initial seed-tuples. These seed tuples have nodes from at least two networks.
- In the second step, by using the network structure and the seed-tuples (generated from the first step), MPROPER aligns the remaining unmatched nodes with a percolation graph-matching (PGM) algorithm. By using structural evidence, to generate larger tuples in the second step, it is possible to add new nodes to the initial seed-tuples.

5.2.1 First Step: SeedGeneration

We now explain how to generate the seed-tuples $\mathcal{A} = \{T_1, T_2, \cdots, T_{|\mathcal{A}|}\}$, by using only sequence similarities. We first define an ℓ-consistent tuple as a natural candidate for seed set. Then, to find these ℓ-consistent tuples, we introduce a heuristic algorithm, called `SeedGeneration`.

A tuple $T = [p_1, p_2, \cdots, p_d]$ is ℓ-consistent, if for every $p_i \in T$ there is at least one other protein $p_j \in T$, such that $BlastBit(p_i, p_j) \geq \ell$. In Section 5.5, we argue that it is reasonable to assume that the BLAST bit-score similarities among real proteins are (pseudo) transitive. Also, we show that proteins with high sequence-similarities, often share many experimentally verified GO terms. Based on these two canonical observations, we argue that, often all the proteins of an ℓ-consistent tuple (with a large enough ℓ) have some experimental GO terms in common, i.e., they have common biological functions. This idea is supported by the two following statements: (i) The pseudo transitivity property of the BLAST bit-scores guarantees that, in an ℓ-consistent tuple T, almost all the $\binom{|T|}{2}$ pairwise couples have high sequence-similarities; and (ii) proteins with high sequence-similarities, often have similar biological functions. Therefore, it is likely that all the proteins in an ℓ-consistent tuple share many biological functions.

Assume $\mathcal{S}$ is the set of all pairwise sequence-similarities, i.e., $\mathcal{S} = \{\mathcal{S}_{1,2}, \cdots, \mathcal{S}_{1,k}, \mathcal{S}_{2,3}, \cdots, \mathcal{S}_{k-1,k}\}$. Also, let $\mathcal{S}_{\geq \ell}$ denote the set of couples with BLAST bit-score similarity of at least ℓ, i.e, $\mathcal{S}_{\geq \ell} = \{[u, v] \in \mathcal{S} \mid BlastBit(u, v) \geq \ell\}$. In `SeedGeneration`, we consider only those couples with BLAST bit-score similarity of at least ℓ. The `SeedGeneration` algorithm, by processing the protein couples from the highest BLAST bit-score similarity to the lowest, fills in the seed-tuples with proteins from several species in a sequential and iterative procedure. At a given step of `SeedGeneration`, assume $[u, v]$ is the next couple that we are going to process, where u and v are from the ith and jth networks, respectively. To add this couple to the seed-tuples $\mathcal{A}$, we consider the following cases:

- Neither u nor v belongs to a tuple in $\mathcal{A}$: we add both nodes to a new tuple, i.e, add $T = [u, v]$ to $\mathcal{A}$.

- Only one of u or v belongs to a tuple in $\mathcal{A}$: assume, without loss of generality, u belongs to a tuple T_u. If the tuple T_u does not have a protein from the network of the other node in this couple (i.e., the jth network), then the node v is added to T_u. This step adds one protein to one existing tuple.
- Both u and v, respectively, belong to tuples T_u and T_v in $\mathcal{A}$: If T_u and T_v do not have a node from, the jth and ith networks, respectively, then we merge the two tuples by the `MergeTuples` algorithm. The goal of `MergeTuples` is to combine the two tuples in order to generate a larger tuple that has nodes from more networks. In this merging algorithm, it is possible to have another (small) tuple as a leftover.

Algorithm 6 describes the `SeedGeneration` algorithm. Also, `MergeTuples` is described in Algorithm 7. For the notations used in these two algorithms, refer to Table 5.2. Furthermore, Example 22 provides an example of the `SeedGeneration` algorithm.

Algorithm 6: The `SeedGeneration` algorithm

Input: Pairwise BLAST bit score similarities $\mathcal{S}$ between k species and ℓ
Output: The seed set $\mathcal{A}$ of tuples

```
$\mathcal{S}_{\geq\ell}$ ← All the couples from the set $\mathcal{S}$ with BLAST bit-score similarity at least $\ell$;
for for all pairs $[u, v]$ in $\mathcal{S}$ from the most similar to the lowest do
    Assume $u \in V_i$ and $v \in V_j$;
    if $T_{\mathcal{A}}(u) = -1$ and $T_{\mathcal{A}}(v) = -1$ then
        Add $T = [u, v]$ to $\mathcal{A}$;
    else if $T_{\mathcal{A}}(u) \neq -1$ and $T_{\mathcal{A}}(v) = -1$ then
        if $V_j(T_{\mathcal{A}}(u)) = -1$ then
            add $v$ to the tuple $T_{\mathcal{A}}(u)$;
    else if $T_{\mathcal{A}}(u) = -1$ and $T_{\mathcal{A}}(v) \neq -1$ then
        if $V_i(T_{\mathcal{A}}(v)) = -1$ then
            add $u$ to the tuple $T_{\mathcal{A}}(v)$;
    else
        if $V_j(T_{\mathcal{A}}(u)) = -1$ and $V_i(T_{\mathcal{A}}(v)) = -1$ then
            MergeTuples($T_{\mathcal{A}}(u), T_{\mathcal{A}}(v)$);
return $\mathcal{A}$;
```

Example 22. Table 5.1 provides an example of the `SeedGeneration` algorithm. This algorithm uses the set of pairwise sequence similarities; this set is sorted from the highest BLAST bit-score to ℓ (an input parameter to the algorithm). In this example, the couple [hs1, mm8] (i.e., the couple of proteins with the highest sequence similarity) generates the first tuple in the seed set. At the third step, one of the nodes from the third couple, i.e., hs1, is already in the tuple T_1 =[hs1, mm8]. Because T_1 does not have any node from the network of ce, the node ce4 is added to T_1. At the eight step, as the two nodes from [ce6, hs9] belong to two different tuples, their corresponding tuples are merged.

Algorithm 7: The `MergeTuples` algorithm

```
Input: Two tuples $T_1$ and $T_2$
Output: The modified tuples $T_1$ and $T_2$
Assume $T_1$ is the tuple that contains the couple with the highest sequence similarity;
for $i$ = 1 to $k$ do
    if $V_i(T_1) = -1$ and $V_i(T_2) \neq -1$ then
        move node $V_i(T_2)$ from $T_2$ to $T_1$;
if $|T_2| = 1$ then
    Delete the tuple $T_2$ ;
```

Table 5.1 – An example of the `SeedGeneration` algorithm. Inputs to this algorithm are the set of pairwise sequence-similarities and a fixed threshold ℓ. The sequence similarities are sorted from the highest BLAST bit-score to ℓ. The seed-tuples $\mathcal{A}$ are generated from the pairwise similarities.

#	Couples	BLAST	Seed-tuples $\mathcal{A}$
1	[hs1, mm8]	1308	[hs1, mm8]
2	[ce6, sc9]	909	[hs1, mm8] and [ce6, sc9]
3	[ce4, hs1]	813	[ce4, hs1, mm8] and [ce6, sc9]
4	[dm15, mm8]	797	[ce4, dm15, hs1, mm8] and [ce6, sc9]
5	[ce654, mm8]	603	[ce4, dm15, hs1, mm8] and [ce6, sc9]
6	[dm15, sc12]	414	[ce4, dm15, hs1, mm8, sc12] and [ce6, sc9]
7	[dm7, hs9]	334	[ce4, dm15, hs1, mm8, sc12], [ce6, sc9] and [dm7, hs9]
8	[ce6, hs9]	282	[ce4, dm15, hs1, mm8, sc12] and [ce6, dm7, hs9, sc9]
9	[dm7, sc63]	101	[ce4, dm15, hs1, mm8, sc12] and [ce6, dm7, hs9, sc9]

5.2.2 Second Step: MultiplePercolation

In the second step of MPROPER, a new PGM algorithm, called `MultiplePercolation`, uses the network structures and the generated seed-tuples from the first step, to align the remaining unmatched nodes. This PGM algorithm uses structural similarities of couples as the only evidence for matching new nodes. The `MultiplePercolation` algorithm adds new tuples in a greedy way, in order to maximize the number of conserved interactions among networks. In `MultiplePercolation`, network structure provides evidence for similarities of unmatched node-couples, and a couple with enough structural similarity is matched. New node-tuples are generated by merging matched couples. Also, if there is enough structural similarity between two nodes from different tuples, the two tuples are merged. In the `MultiplePercolation` algorithm, we look for tuples that contain nodes from more networks, i.e., a tuple that has nodes from more networks is more valuable. Next, we explain the `MultiplePercolation` algorithm in detail.

Assume π is the set of aligned tuples at a given time step of the `MultiplePercolation` algorithm. Note that we have initially $\pi = \mathcal{A}$, where $\mathcal{A}$ is the output of `SeedGeneration`. Let $\pi_{i,j}$ denote the set of pairwise alignments between nodes from the ith and jth networks: A couple $[u, v]$, where $u \in V_i$ and $v \in V_j$, belongs to the set $\pi_{i,j}$, if and only if there is a tuple $T \in \pi$ such that both u and v are in that tuple, i.e., $T_\pi(u) = T_\pi(v) \neq -1$. The set $\pi_{i,j}$ is defined as

$$\pi_{i,j} = \{[u, v] | u \in V_i \text{ and } v \in V_j \text{ s.t. there exists } T \in \pi \text{ where } u, v \in T\}.$$

The score of a couple of nodes is the number of their common neighbours in the set of previously aligned tuples. Formally, we define the score of a couple $[u, v]$, $u \in V_i$ and $v \in V_j$ as

$$score([u, v]) = |\{[p_i, p_j] \in \pi_{i,j} \text{ s.t. } (u, p_i) \in E_i \text{ and } (v, p_j) \in E_j\}|. \tag{5.1}$$

The score of a couple is equal to the number of interactions that remain conserved if this couple is added as a new tuple to the set of currently aligned tuples. Alternatively, it is possible to interpret the score of a couple as follows: All the couples $[p_i, p_j] \in \pi_{i,j}$ provide marks for their neighboring couples, i.e., the couples in $N_i(p_i) \times N_j(p_j)$ receive one mark from $[p_i, p_j]$. The score of a couple is the number of marks it has received from the previously aligned couples (note that aligned couples are subsets of aligned tuples).

In the `MultiplePercolation` algorithm, the initial seed-tuples provide structural evidence for the other unmatched couples. More precisely, for a tuple $T = [p_1, p_2, \cdots, p_d]$, all the $\binom{d}{2}$ possible couples $[p_i, p_j]$, which are subset of the tuple T, spread marks to their neighboring couples in the networks $V(p_i)$ and $V(p_j)$. After this step, the couple $[u, v]$ with the highest number of marks (but at least r) is the next candidate to get matched. The alignment process, similar to `SeedGeneration`, is as follows:

- If $T_\pi(u) = -1$ and $T_\pi(u) = -1$, then we add a new tuple $T = [u, v]$ to the set of aligned tuples π.
- If exactly one of the two nodes u or v belongs to a tuple $T \in \pi$, by adding the other node to T (if it is possible[1]), we generate a tuple with nodes from one more network.
- If both u and v belong to different tuples of π, by merging these two tuples (again, if possible), we make a larger tuple.

After the alignment process, $[u, v]$ spread out marks to the other couples, because it is a newly matched couple. Then, recursively new couples are matched and added to the set of aligned tuples. The alignment process continues to the point that there is no couple with a score of at least r. Algorithm 8 describes `MultiplePercolation`. For the notations refer to Table 5.2. An example of the `MultiplePercolation` algorithm is provided in Example 23.

Example 23. Figure 5.1 provides an example of the `MultiplePercolation` algorithm over graphs $G_{1,2,3}$. Dark-green nodes are the initial seed-tuples. The tuple $[x_1, x_2, x_3]$ is an example

---

[1] Refer to Algorithm 8.

Algorithm 8: The `MultiplePercolation` algorithm

```
Input: $G_1(V_1, E_1), G_2(V_2, E_2), \cdots, G_k(V_k, E_k)$ seed tuples $\mathcal{A}$ and the threshold $r$
Output: The set of aligned tuples $\pi$
$\pi \leftarrow \mathcal{A}$;
while there exists a couple with score at least $r$ do
    $[u, v] \leftarrow$ the couple with the highest score, where $u \in V_i$ and $v \in V_j$;
    if $T_\pi(u) = -1$ and $T_\pi(v) = -1$ then
        Add $T = [u, v]$ to $\pi$;
    else if $T_\pi(u) \neq -1$ and $T_\pi(v) = -1$ then
        if $V_j(T_\pi(u)) = -1$ then
            add $v$ to $T_\pi(u)$;
    else if $T_\pi(u) = -1$ and $T_\pi(v) \neq -1$ then
        if $V_i(T_\pi(v)) = -1$ then
            add $u$ to $T_\pi(v)$th tuple;
    else
        if $V(T_\pi(u)) \cap V(T_\pi(p_j)) = \emptyset$ then
            Merge the two tuples $T_\pi(u)$ and $T_\pi(v)$ into one tuple;
return $\pi$;
```

of a seed tuple that contains nodes from all the three networks. $[y_1, y_2]$ is a seed couple between networks G_1 and G_2. All the pairwise couples, which are subsets of the initial seed-tuples, provide structural evidence for the other nodes. In this example, after that initial seed-tuples spread out marks to other couples, the couples $[w_1, w_2]$ and $[u_2, u_3]$ have the highest score (their score is three). Hence we align them first. Among the couples with score two, $[w_1, u_3]$ is not a valid alignment; because the nodes w_1 and u_3 are matched to different nodes in G_2 (also, this true for w_2 and u_2). The set of aligned tuples is $\{[u_1, u_2, u_3], [v_1, v_2, v_3], [w_1, w_2], [z_2, z_3]\}$. Here, there is not enough information to match v_1 and v_3 directly, but as they both are matched to v_2, we can align them through transitivity of the alignments. Furthermore, if we continue the percolation process, it is possible to match the couples $[i_1, i_2]$ and $[i_1, i_3]$; it results in the tuple $[i_1, i_2, i_3]$. Note that, by aligning all the networks at the same time, we have access to more structural information. For example, although the pairwise alignment of G_1 and G_3 does not provide enough evidence to align $[v_1, v_3]$, it is possible to align this couple by using the side information we can get through G_2.

5.3 Performance Measures

Comparing global MNA algorithms is a challenging task for several reasons. Firstly, it is not possible to directly evaluate the performance of algorithms, because the true node-mappings for real biological-networks is not known. Secondly, algorithms can return tuples of differ-

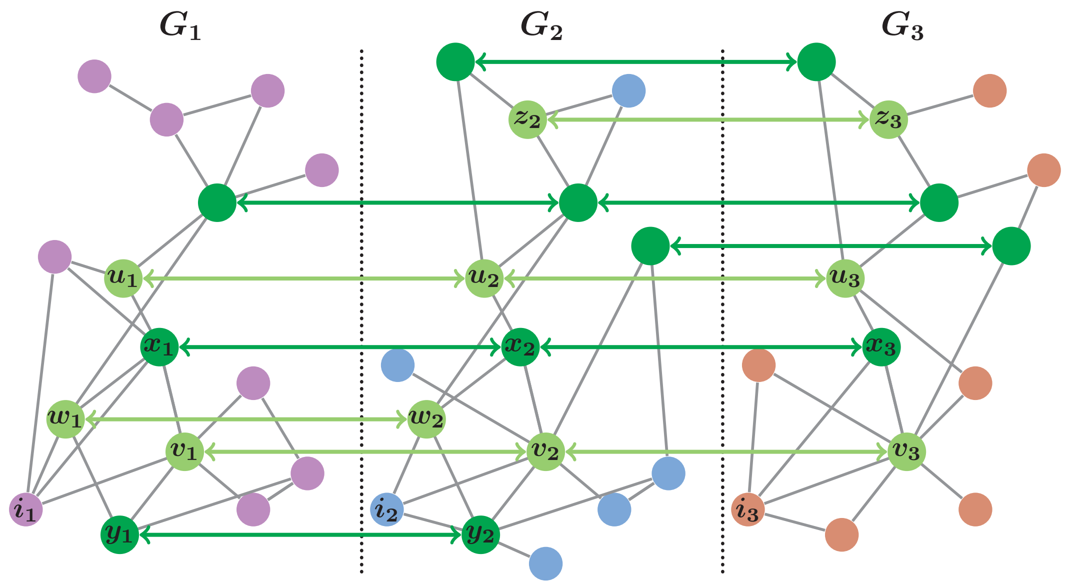


Figure 5.1 – An example of the `MultiplePercolation` algorithm. The alignment is performed over graphs G_1, G_2 and G_3. Dark-green nodes are the initial seed-tuples. Light-green nodes are tuples that are matched in the PGM process.

ent sizes. Although the fundamental goal of a global MNA algorithm is to find tuples with nodes from many different networks, some algorithms tend to return tuples of smaller sizes. Therefore, tuples of different sizes make the comparison more difficult. For these reasons, we use several measures from the literature. In addition, we introduce a new measure, using the information content of aligned tuples.

We first compare global MNA algorithms based on their performance in generating tuples that cover nodes from more networks. The best tuples are those that contain nodes from all the k networks, whereas tuples with nodes from only two networks are the worst. The d-coverage of clusters denotes the number of clusters with nodes from exactly d networks [66]. Note that, for many-to-many alignment algorithms, it is possible to have more than d nodes in a cluster with nodes from d networks. Therefore, for the number of proteins in clusters with different d-coverages, we also consider the total number of nodes in those clusters [66].

A major group of measures evaluate the performance of algorithms, using the functional similarity of aligned proteins. A cluster is annotated, if it has at least two proteins that are annotated by at least one GO term. An annotated cluster is consistent, if all of the annotated proteins in that cluster share at least one GO term. We define $\#AC$ as the total number of annotated clusters. Furthermore, $\#AC_d$ represent the number of annotated clusters with a coverage d. For the number of consistent clusters, we define $\#CC$ and $\#CC_d$ similarly. Also, the number of proteins, in a consistent cluster with a coverage d, is shown by $\#CP_d$. The specificity of an alignment is defined as the ratio of the number of consistent clusters to the

Table 5.2 – Table of notations

$G_i(V_i, E_i)$	A network with vertex set V_i and edge set E_i.
$N_i(u)$	The set of neighbors of node u in G_i.
$BlastBit(u, v)$	BLAST bit-score similarity of two proteins u and v
$[u, v]$	A couple of proteins u and v.
T	A cluster or tuple.
$\mathcal{A}$	Initial seed-tuples.
π	The final alignment.
$\|T\|$	Number of nodes in T.
$V(T)$	The set of networks such that have a node in the tuple T.
$K(u)$	The $K(u)$th network such that $u \in V_{K(u)}$.
$V_i(T)$	Returns the node u in T such that $u \in V_i$. If there is not such tuple, we define $V_i(T) = -1$
$\mathcal{S}$	The set of all pairwise BLAST bit-score similarities.
$\mathcal{S}_{\geq \ell}$	The set of all pairwise BLAST bit-score similarities that are at least ℓ.
$T_\pi(u)$	Returns the tuple $T \in \pi$ such that $u \in T$. If there is no such tuple, we define $T_\pi(u) = -1$.
E_{T_i,T_j}	The set of all the interactions between nodes from the two tuples T_i and T_j, i.e., $E_{T_i,T_j} = \{e = (u, v) \| u \in T_i, v \in T_j\}$.
$V(E_{T_i,T_j})$	The set of networks such that have an edge in E_{T_i,T_j}.
$C(\pi)$	The set of consistent clusters in an alignment π.

number of annotated clusters:

$$Spec. = \frac{\#CC}{\#AC} \text{ and } Spec._d = \frac{\#CC_d}{\#AC_d}. \tag{5.2}$$

Mean entropy (ME) and mean normalized entropy (MNE) are two other measures that calculate the consistency of aligned proteins by using GO terms. The entropy (E) of a tuple $T = [p_1, p_2, \cdots, p_d]$, with the set of GO terms $GO(T) = \{GO_1, GO_2, \cdots, GO_m\}$, is defined as

$$E(T) = -\sum_{i=1}^{m} g_i \log p_i, \tag{5.3}$$

where g_i is the fraction of proteins in T that are annotated with the GO term GO_i. ME is defined as the average of $E(T)$ over all the annotated clusters. Normalized entropy (NE) is defined as

$$NE(T) = \frac{1}{\log m} E(T), \tag{5.4}$$

where m is the number of different GO terms in T. Similarly, MNE is defined as the average of $NE(T)$ over all the annotated clusters.

To avoid the *shallow annotation problem*, Alkan and Erten [12] and Gligorijević et al. [66]

suggest to restrict the protein annotations to the fifth level of the GO directed acyclic graph (DAG): (i) by ignoring the higher level GO annotations, and (ii) by replacing the deeper-level GO annotations with their ancestors at the fifth level. For the specificity and entropy evaluations, we use the same restriction method in our comparisons.

The way we deal with the GO terms can greatly affect the comparison results. Indeed, there are serious drawbacks with the restriction of the GO annotations to a specific level. Firstly, the depth of a GO term is not an indicator of its specificity. The GO terms that are at the same level do not have the same semantic precision, and a GO term at a higher level might be more specific than a term at a lower level [158]. Also, it is known that the depth of a GO term reflects mostly the vagaries of biological knowledge, rather than anything intrinsic about the terms [124]. Secondly, there is no explanation (e.g., in [12, 66]) about why we should restrict the GO terms to the fifth level. Also, the notion of consistency for a cluster (i.e., sharing at least one GO term) is very general and does not say anything about how specific are the shared GO terms. Furthermore, from our experimental studies, we observe that two random proteins share at least one experimentally verified GO term with probability 0.21, whereas five proteins share at least one GO term with a very low probability of 0.002.[2] To overcome these limitations, we define the *semantic similarity* (SS_p) measure for a cluster of proteins. This is the generalization of a measure that is used for semantic similarity of two proteins [158, 168]. For an annotated tuple T, we define SS_p as follows.

Assume $|annot(t_i)|$ is the number of proteins that are annotated to the GO term t_i. The frequency of t_i is defined as

$$freq(t_i) = |annot(t_i)| + \sum_{s \in successors(t_i)} |annot(s)|, \tag{5.5}$$

where $successors(t_i)$ is the successors of the term t_i in its corresponding gene-annotation DAG. The relative frequency $p(t_i)$ for a GO term t_i is defined as

$$p(t_i) = \frac{freq(t_i)}{freq(root)}. \tag{5.6}$$

The *information content* (IC) [158] for a term t_i is defined as

$$IC(t_i) = -\log(p(t_i)). \tag{5.7}$$

The semantic similarity between the d terms $\{t_1, t_2, \cdots, t_d\}$ is defined as

$$SS(t_1, t_2, \cdots, t_d) = IC(LCA(t_1, t_2, \cdots, t_d)), \tag{5.8}$$

where $LCA(t_1, t_2, \cdots, t_d)$ is the lowest common ancestor of terms t_i in DAG. For proteins

---

[2] For more information refer to Appendix 5.A.

$p_1, p_2, \cdots, p_d$, we define semantic similarity $SS_p(p_1, p_2, \cdots, p_d)$ as

$$SS_p(p_1, p_2, \cdots, p_d) = \max_{t_1 \in GO(p_2), t_2 \in GO(p_2), \cdots, t_d \in GO(p_d)} IC(LCA(t_1, t_2, \cdots, t_d)), \tag{5.9}$$

where $GO(p_i)$ are the GO annotations of p_i. The sum of SS_p values for all clusters in an alignment π is shown by $SS_p(\pi)$. Let $\overline{SS_p}(\pi)$ denote the average of SS_p values, i.e., $\overline{SS_p}(\pi) = \frac{SS_p(\pi)}{|\pi|}$. Note that, algorithms with higher values of $SS_p(\pi)$ and $\overline{SS_p}(\pi)$, result in alignments with higher qualities, because these alignments contain clusters with more specific functional similarity among their proteins.

The second group of measures evaluate the performance of global MNA algorithms based on the structural similarity of aligned networks. We define edge correctness (EC) as a generalization of the introduced measures in [112, 149]. EC is a measure of edge conservation between consistent clusters under a multiple alignment π. For two tuples T_i and T_j, let E_{T_i,T_j} denote the set of all the interactions between nodes from these two tuples, i.e., $E_{T_i,T_j} = \{e = (u, v) | u \in T_i, v \in T_j\}$. The set of networks such that have an edge in E_{T_i,T_j} is defined by $V(E_{T_i,T_j})$. Theoretically, we can have a conserved interaction between two clusters T_i and T_j, if they have nodes from at least two similar networks, i.e., $|V(T_i) \cap V(T_j)| \geq 2$. The interaction between two clusters T_i and T_j is conserved, if there are at least two edges from two different networks between these clusters, i.e., $|V(E_{T_i,T_j})| \geq 2$. The EC measure is defined

$$EC(\pi) = \frac{\Delta(\pi)}{E(\pi)}, \tag{5.10}$$

where $E(\pi)$ is the total number edges between all the consistent clusters T_i and T_j, such that $|V(T_i) \cap V(T_j)| \geq 2$. Also, $\Delta(\pi)$ is the total number of edges between those clusters with $|V(E_{T_i,T_j})| \geq 2$.

Cluster interaction quality (CIQ) measures the structural similarity as a function of the conserved interactions between different tuples [12]. The conservation score $cs(T_i, T_j)$ is defined as

$$cs(T_i, T_j) = \begin{cases} 0 & \text{if } |V(T_i) \cap V(T_j)| = 0 \text{ or } |V(E_{T_i,T_j})| = 1 \\ \dfrac{|V(E_{T_i,T_j})|}{|V(T_i) \cap V(T_j)|} & \text{otherwise,} \end{cases} \tag{5.11}$$

where $|V(T_i) \cap V(T_j)|$ and $|V(E_{T_i,T_j})|$ are the number of distinct networks with nodes in both $T_{i,j}$ and with edges in E_{T_i,T_j}, respectively. $CIQ(\pi)$ is defined as follows:

$$CIQ(\pi) = \frac{\sum_{\forall T_i, T_j \in \pi} |E_{T_i,T_j}| \times cs(T_i, T_j)}{\sum_{\forall T_i, T_j \in \pi} |E_{T_i,T_j}|}. \tag{5.12}$$

We can interpret CIQ as a generalization of S^3 [165], a measure for evaluating the structural similarity of two networks.

5.4 Experimental Results

In this section, we compare MPROPER with several state-of-the-art global MNA algorithms [59]: FUSE (F) [66], BEAMS (B) [12], SMETANA (S) [163] and CSRW (C) [87]. Also, we compare our algorithm with IsoRankN (I) [119], which is one of the very first global MNA algorithms for PPI networks.

Table 5.3 provides a brief description of the PPI networks for five major eukaryotic species that are extracted from the IntAct database [1, 76]. The amino-acid sequences of proteins are extracted in the FASTA format from UniProt database [4, 15]. The Blast bit-score similarities [13] are calculated using these amino-acid sequences. We consider only experimentally verified GO terms, in order to avoid biases induced by annotations from computational methods (mainly from sequence similarities).[3] More precisely, we consider the GO terms with codes "EXP", "IDA", "IMP", "IGI" and "IEP", and we exclude the annotations derived from computational methods and protein-protein interaction experiments.

Table 5.3 – PPI networks of five major eukaryotic species from IntAct molecular interaction database [1, 76].

species	Abbrev.	#nodes	#edges	Avg. deg.
C. elegans	ce	4950	11550	4.67
D. melanogaster	dm	8532	26289	6.16
H. sapiens	hs	19141	83312	8.71
M. musculus	mm	10765	22345	4.15
S. cerevisiae	sc	6283	76497	24.35

5.4.1 Comparisons

We first investigate the optimality of `SeedGeneration` in generating seed-tuples from sequence similarities. To have an upper-bound on the number of proteins in the set of seed-tuples $\mathcal{A}$, we look at the maximum bipartite graph matching between all pairwise species, i.e., all the proteins in all the possible $\binom{k}{2}$ matchings. The total number of nodes that are matched in at least one of these bipartite matchings, provide an upper-bound for the number of matchable nodes. Figure 5.2 compares `SeedGeneration`, the proposed upper-bound and MPROPER for different values of ℓ, and the other algorithms based on the total number of aligned proteins. In Figures 5.3 and 5.4, we compare algorithms based on different d-coverages. We observe that MPROPER finds the most number of clusters with 5-coverage among all the algorithms. Furthermore, we observe that MPROPER has the best overall coverage (for clusters of size five to two). For example, we also observe that, for $\ell = 40$, the `SeedGeneration` algorithm aligns 28608 proteins (compared to 30820 proteins that we found as an upper-bound) in 1366, 1933, 2342 and 3510 clusters of size 5, 4, 3 and 2, respectively. The second step of MPROPER

---

[3] We obtained GO terms from http://www.ebi.ac.uk/GOA/downloads.

(i.e., `MultiplePercolation`) extends the initial seed tuples to 40566 proteins aligned in 3076, 2719, 2502 and 3402 clusters of size 5, 4, 3 and 2, respectively.

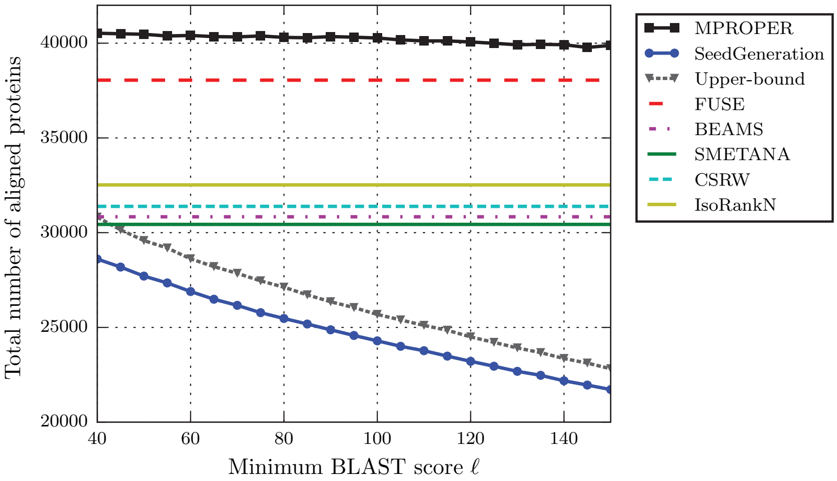


Figure 5.2 – Total number of aligned proteins. For MPROPER, we set $r = 1$. We observe that MPROPER aligns the most number of proteins. Also, it is clear that the number of aligned nodes, in the first step of MPROPER (i.e., `SeedGeneration`), is close to the values found by the proposed upper-bound.

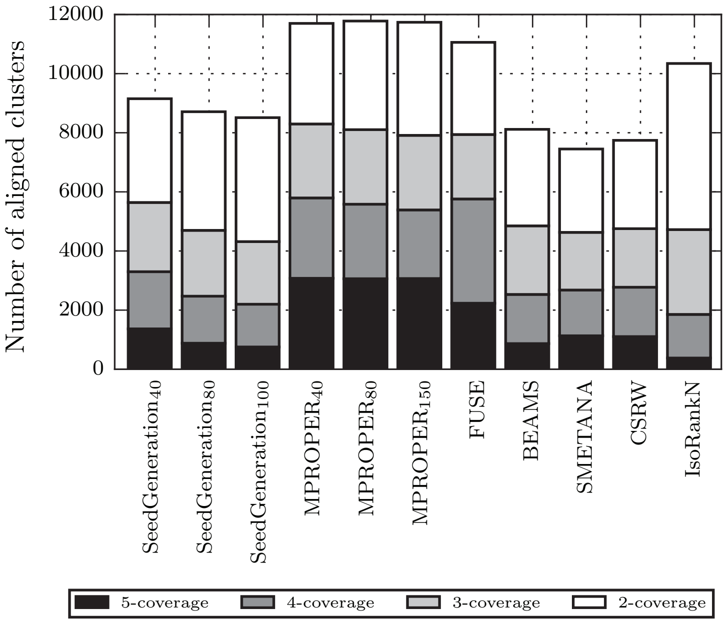


Figure 5.3 – The coverage of alignments from different algorithms. The results are for clusters with nodes from five, four, there and two networks. For MPROPER, we set $r = 1$ and $\ell \in \{40, 80, 100\}$. We observe that MPROPER finds the most number of clusters. Also, our proposed algorithm finds the most number of clusters with nodes from all the five networks.

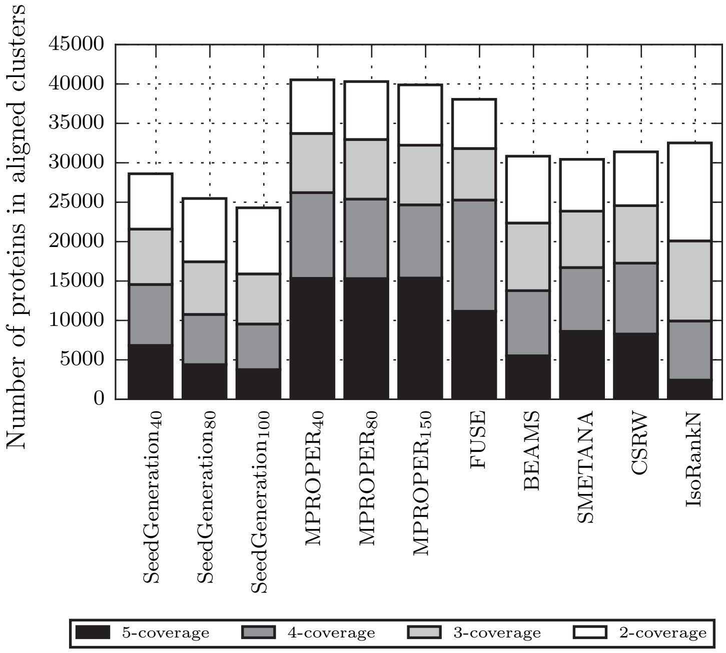


Figure 5.4 – Number of proteins in clusters with different d-coverages. The results are for clusters with nodes from five, four, there and two networks. For MPROPER, we set $r = 1$ and $\ell \in \{40, 80, 100\}$.

An algorithm with a good d-coverage does not necessarily generate high-quality clusters (in terms of functional similarity of proteins). For this reason, we look at the number of consistent clusters. For example, although IsoRankN generates the maximum number of clusters with proteins from two species (see Figure 5.3), only a small fraction of these clusters are consistent (see Figure 5.5). Also, in Figure 5.5, we observe that MPROPER returns the largest number of consistent clusters with proteins from five different species. Tables 5.4, 5.5, 5.6 and 5.7 provide detailed comparisons for clusters with different coverages.

Table 5.4 compares algorithms over clusters with nodes from five networks. The second step of MPROPER (i.e., `MultiplePercolation`) uses PPI networks to generate 3076 clusters out of initial seed-tuples. We observe that MPROPER (for $\ell = 40$) finds an alignment with the maximum d-coverage, $\#CC_5$, $\#CP_5$ and $SS_p(\pi)$. In addition, the first step of MPROPER (i.e., `SeedGeneration`) has the best performance on $Spec._5$, $\overline{SS_p}(\pi)$ and MNE. This was expected, because `MultiplePercolation` uses only network structure, a less reliable source of information for functional similarity in comparison to sequence similarities, to align new nodes. From this table, it is clear that MPROPER outperforms the other algorithms with respect to all the measures.

In Figure 5.7, we compare algorithms based on the EC measure. We observe that MPROPER (for values of ℓ close to 150) and SMETANA find alignments with the highest EC score. In Figure 5.8, to calculate EC, we consider only the edges between consistent clusters. We observe

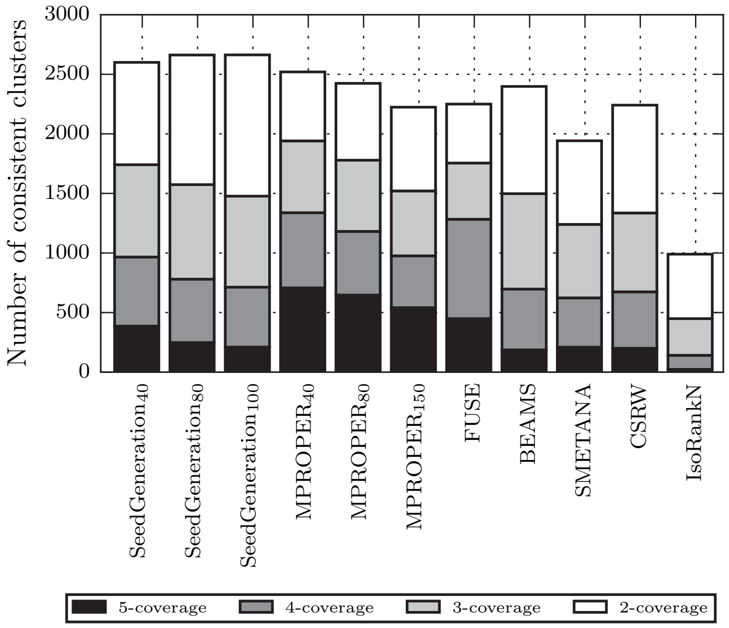


Figure 5.5 – Number of consistent clusters. The results are for clusters with nodes from five, four, there and two networks. For MPROPER, we set $r = 1$ and $\ell \in \{40, 80, 100\}$. We observe that `SeedGeneration` and MROPER find the most number of consistent clusters and consistent cluster with nodes from all the five networks, respectively.

that MPROPER has the best performance among all the algorithms. This shows that MPROPER finds alignments where (i) many of the aligned clusters are consistent and (ii) there are many conserved interactions among these consistent clusters. CIQ is another measure, based on structural similarity of aligned networks, for further evaluating the performance of algorithms. In Figure 5.9, we observe that again MPROPER and SMETANA find alignments with the best CIQ score.

Table 5.4 – Comparison results for clusters of size five. For MPROPER we set $r = 1$.

	SeedGeneration (ℓ)			MPROPER (ℓ)			F	B	S	C	I
	40	80	150	40	80	150					
d-coverage	1366	880	568	**3076**	3062	3068	2233	867	1132	1104	379
$\#CC_5$	386	248	159	**707**	647	541	449	187	209	200	23
$\#CP_5$	1930	1240	795	**3535**	3235	2705	2245	1082	1279	1234	126
$Spec._5$	**0.291**	0.286	0.284	0.244	0.222	0.184	0.21	0.22	0.187	0.185	0.063
$SS_p(\pi)$	5294	3519	2251	**10659**	10002	9285	7078	2788	3315	3097	554
$\overline{SS_p}(\pi)$	3.928	**4.018**	3.993	3.55	3.326	3.074	3.22	3.239	2.944	2.818	1.482
MNE	**2.927**	2.943	3.049	3.008	3.071	3.144	3.014	3.162	3.312	3.071	3.469

Table 5.5 – Comparison results for clusters of size four. For MPROPER we set $r = 1$.

	SeedGeneration (ℓ)			MPROPER (ℓ)			F	B	S	C	I
	40	80	150	40	80	150					
d-coverage	1933	1591	1133	2719	2520	2321	**3527**	1663	1547	1670	1475
$\#CC_4$	580	532	392	631	534	435	**834**	510	414	474	118
$\#CP_4$	2320	2128	1568	2524	2136	1740	**3336**	2398	1903	2272	560
$Spec._4$	0.339	0.366	**0.369**	0.277	0.256	0.224	0.299	0.329	0.291	0.306	0.092
$SS_p(\pi)$	8309	7335	5465	10449	9087	7902	**12829**	7043	5863	6522	2953
$\overline{SS_p}(\pi)$	4.591	4.814	**4.982**	4.213	3.984	3.726	4.095	4.34	3.922	4.044	2.156
MNE	**2.565**	2.648	2.717	2.571	2.621	2.668	2.597	2.733	2.664	2.73	3.168

Table 5.6 – Comparison results for clusters of size three. For MPROPER we set $r = 1$.

	SeedGeneration (ℓ)			MPROPER (ℓ)			F	B	S	C	I
	40	80	150	40	80	150					
d-coverage	2342	2227	1842	2502	2522	2523	2180	2320	1951	1981	**2869**
$\#CC_3$	775	794	692	603	598	545	472	**801**	617	662	308
$\#CP_3$	2325	2382	2076	1809	1794	1635	1416	**2886**	2132	2352	1027
$Spec._3$	0.462	0.486	**0.5**	0.384	0.382	0.349	0.35	0.437	0.417	0.436	0.153
$SS_p(\pi)$	11509	**11672**	9988	10040	10070	9430	8197	11509	9064	9526	7463
$\overline{SS_p}(\pi)$	6.007	6.319	**6.394**	5.263	5.348	4.995	4.956	5.706	5.441	5.587	3.224
MNE	**2.239**	2.29	2.372	2.312	2.336	2.35	2.348	2.276	2.31	2.264	2.83

Table 5.7 – Comparison results for clusters of size two. For MPROPER we set $r = 1$.

	SeedGeneration (ℓ)			MPROPER (ℓ)			F	B	S	C	I
	40	80	150	40	80	150					
d-coverage	3510	4013	4411	3402	3675	3825	3118	3265	2820	2988	**5620**
$\#CC_2$	859	1088	**1357**	579	645	703	495	900	702	905	541
$\#CP_2$	1718	2176	**2714**	1158	1290	1406	990	2309	1644	2073	1224
$Spec._2$	0.611	0.619	**0.646**	0.527	0.519	0.529	0.462	0.557	0.558	0.618	0.231
$SS_p(\pi)$	15049	18777	**22849**	10664	12032	12918	9035	14749	12003	14898	12853
$\overline{SS_p}(\pi)$	8.157	8.286	**8.551**	7.025	7.153	7.161	6.118	7.375	7.378	8.124	4.21
MNE	1.946	**1.944**	1.968	2.01	2.03	2.002	2.163	1.987	1.951	1.987	2.464

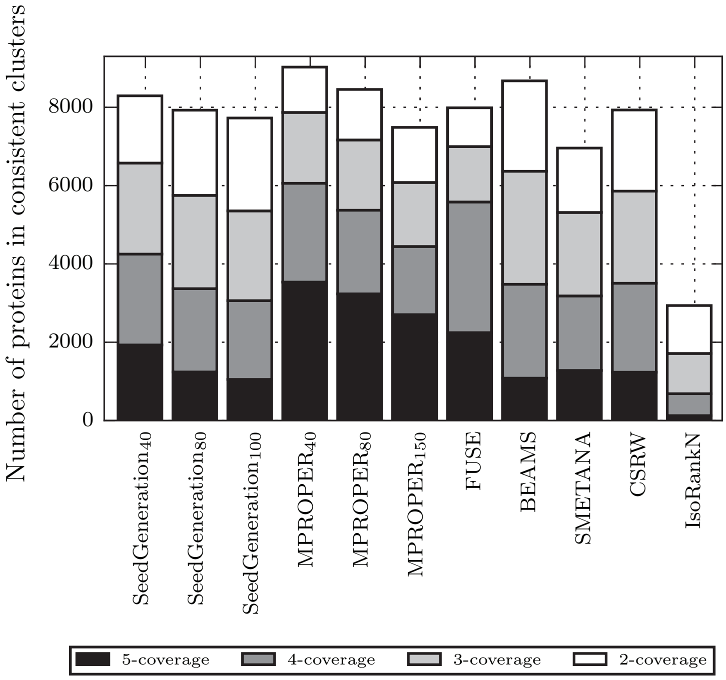


Figure 5.6 – Number of proteins in consistent clusters with different d-coverages. The results are for clusters with nodes from five, four, there and two networks. For MPROPER, we set $r = 1$ and $\ell \in \{40, 80, 100\}$. We observe that MROPER finds the most number of proteins in consistent clusters and consistent cluster with nodes from all the five networks.

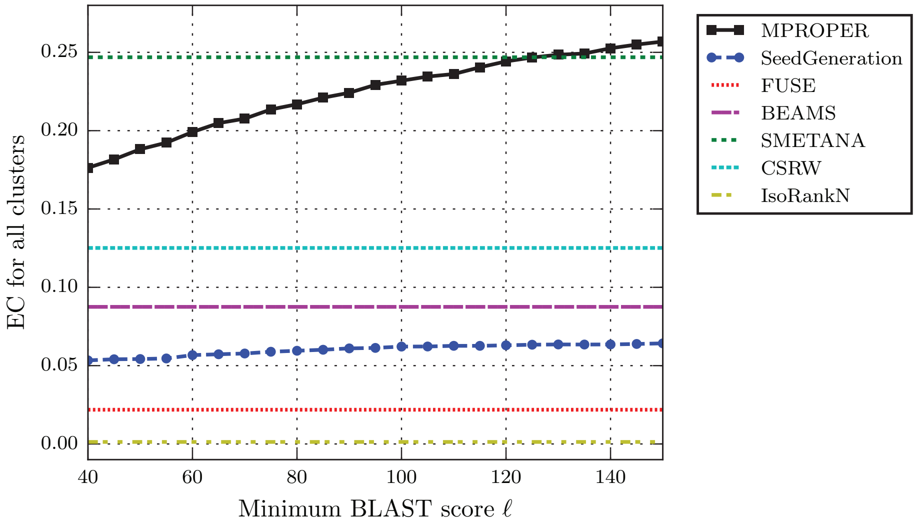


Figure 5.7 – Comparison based on the EC measure for all clusters. The results for MPROPER are presented for $r = 1$ and different values of ℓ. We observe that MPROPER and SMETANA find alignments with the highest EC score.

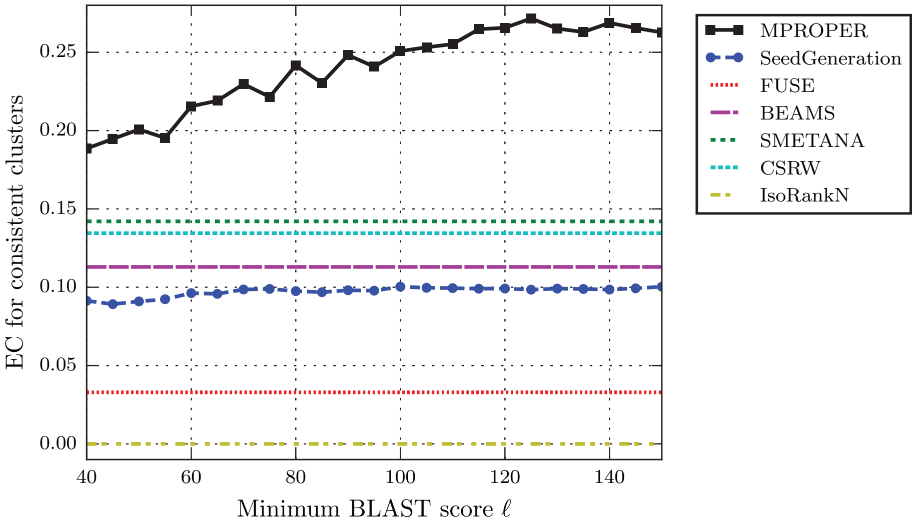


Figure 5.8 – Comparison based on the EC measure for consistent clusters. The results for MPROPER are presented for $r = 1$ and different values of ℓ. We observe that MPROPER has the best performance among all the algorithms.

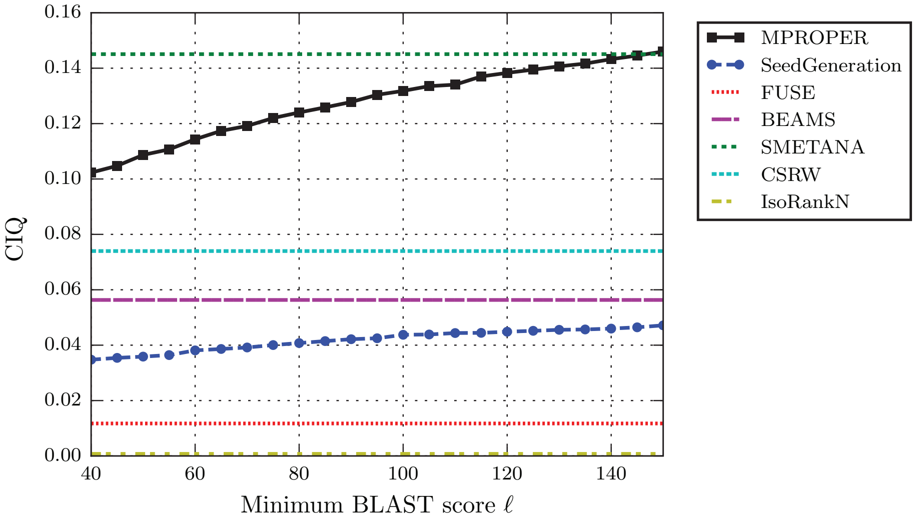


Figure 5.9 – Comparison based on the CIQ measure. The results for MPROPER are presented for $r = 1$ and different values of ℓ. We observe that MPROPER and SMETANA find alignments with the highest CIQ score.

5.4.2 Computational Complexity

The computational complexity of the `SeedGeneration` algorithm is $O\left(|\mathcal{S}_{\geq\ell}|\log|\mathcal{S}_{\geq\ell}|\right)$; it includes (i) sorting all the sequence similarities from the highest to the lowest, and (ii) processing them. The computational complexity of the `MultiplePercolation` algorithm is $O\left(k^2\left(|E_1|+|E_2|\right)\min\left(D_1, D_2\right)\right)$, where $D_{1,2}$ are the maximum degrees in the two networks. To have a scalable algorithm, for very large networks, we can use the MapReduce implementation of `MultiplePercolation`.

5.5 Why MPROPER?

One simple solution to the global MNA problem is to first perform all the pairwise alignments between different networks. Then to find the final multiple alignment by merging all these pairwise alignments. The main drawback of this approach is that the collection of these pairwise alignments might be inconsistent. For example, for nodes $u_{1,2,3} \in V_{1,2,3}$, if u_1 is matched to u_2 and u_2 to u_3, but u_1 is matched to another node from G_3, then it is not possible to generate a consistent one-to-one global MNA from these pairwise alignments. In contrast to the idea of merging different pairwise alignments, our approach has three main advantages:

- It aligns all the k networks at the same time. Therefore, it will always end up with a consistent one-to-one global MNA.
- It uses the structural information from all networks simultaneously.
- The `SeedGeneration` algorithm gives more weight to the pairs of species that are evolutionarily closer to each other. For example, as H. sapiens and M. musculus are very close, (i) many couples from these two species are matched first, and (ii) there are more couples of proteins with high-sequence similarities from these two species. Hence there are more tuples that contain proteins from both H. sapiens and M. musculus.

In rest of this section, we provide experimental evidence and theoretical results that support the good performance of the MPROPER algorithm.

5.5.1 Why SeedGeneration?

The first step of MROPER (`SeedGeneration`) is a heuristic algorithm that generates seed-tuples. `SeedGeneration` is designed based on the following observations.

- First, we argue that proteins with high BLAST bit-score similarities share GO terms with a high probability. To provide experimental evidence for our hypothesis, we look at the biological similarity of protein couples versus their BLAST bit-score similarities. For this reason, we define a gene-ontology consistency (GOC) measure (based on the measure introduced in [139]) to evaluate the relationship between BLAST bit-scores and the experimentally verified GO terms. This measure represents the percentage of pairs of proteins with BLAST bit-score similarity of at least ℓ, such that they share at least one GO term. Formally, we define

$$goc_{\geq \ell} = \frac{|\{[p_i, p_j] | BLAST(p_i, p_j) \geq \ell \text{ and } go(p_i) \cap go(p_j) \neq \emptyset\}|}{|\{[p_i, p_j] | BLAST(p_i, p_j) \geq \ell, go(p_i) \neq \emptyset \text{ and } go(p_j) \neq \emptyset\}|}. \tag{5.13}$$

In this section, we consider only experimentally verified GO terms. Figure 5.10 shows the $goc_{\geq\ell}$ measure for couples of proteins among five eukaryotic species, namely C. elegans (ce), D. melanogaster (dm), H. sapiens (hs), M. musculus (mm) and S. cerevisiae (sc). In this figure, the results are provided for cases, where we consider (i) all the experimental GO terms, (ii) cellular component (CC) annotations, (iii) molecular function (MF) annotations, and (iv) biological process (BP) annotations. For further experiments, we look at the average of semantic similarity SS_p (5.9) between couples of proteins with BLAST bit-score similarity of at least ℓ. Figure 5.11 shows the SS_p for couples of proteins with BLAST bit-score similarities of at least ℓ. We observe that, for couples of proteins with higher BLAST bit-score similarities, the average of SS_p measure increases.

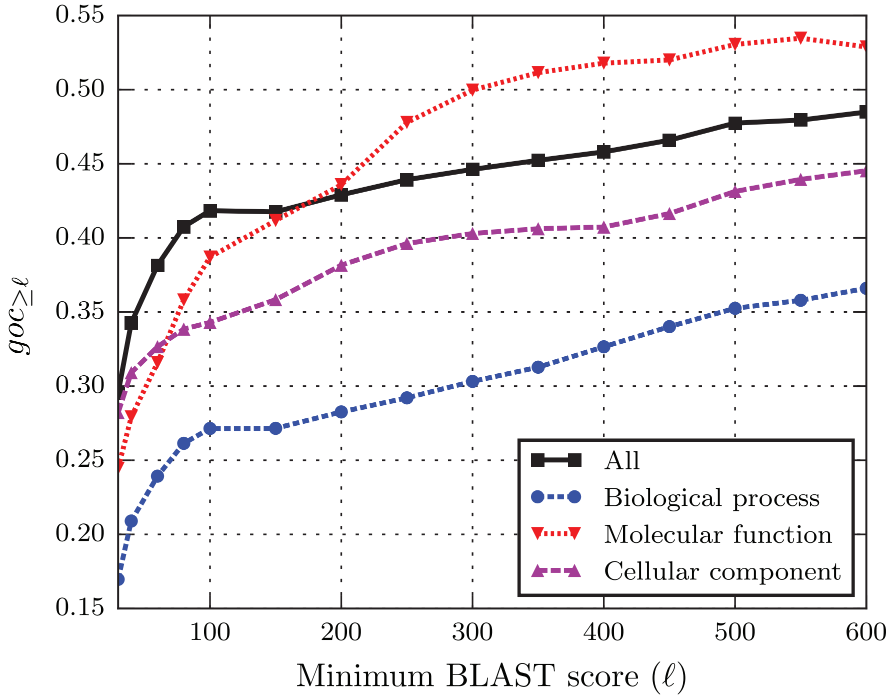


Figure 5.10 – The $goc_{\geq\ell}$ measure for couples of proteins with BLAST bit-score similarities of at least ℓ.

- Second, we look at the transitivity of BLAST bit-score similarities for real proteins. In general, the BLAST similarity is not a transitive measure, i.e., for proteins p_1, p_2 and p_3 given that couples $[p_1, p_2]$ and $[p_2, p_3]$ are similar, we can not always conclude that the two proteins p_1 and p_3 are similar. But real proteins cover a small portion of the space of possible amino-acid sequences, and it might be safe to assume a (pseudo) transitivity property for them.

Example 24. Assume, we have the three toy proteins p_1, p_2 and p_3 with amino-acid sequences $[MMMMMM]$, $[MMMMMMVVVVVV]$ and $[VVVVVV]$, respectively. In this example, p_2 is similar to both p_1 and p_3, where p_1 is not similar to p_3. Indeed, we

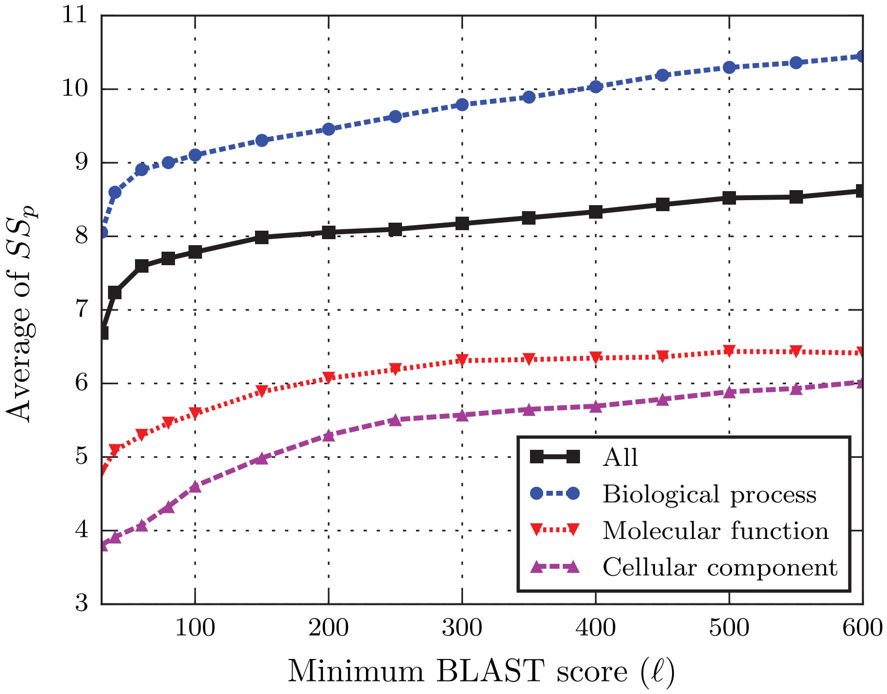


Figure 5.11 – Average of SS_p for couples of proteins with BLAST bit-score similarities of at least ℓ. We observe that, for couples of proteins with higher BLAST bit-score similarities, the average of SS_p measure increases.

have $BlastBit(p_1, p_2) = 11.2$, $BlastBit(p_2, p_3) = 10.0$ and $BlastBit(p_1, p_3) = 0$.

To empirically evaluate the transitivity of BLAST bit-scores, we define a new measure for an estimation of the BLAST bit-score similarity of two proteins p_1 and p_3, when we know that there is a protein p_2, such that BLAST bit-score similarities between p_2 and both p_1, p_3 are at least ℓ. Formally, we define $\alpha_{\ell,\beta}$ as

$$\alpha_{\ell,\beta} = \operatorname*{argmax}_{\alpha} \left[\mathbb{P}[BLAST(i,k) \geq \alpha \times \ell \mid BLAST(i,j) \geq \ell, BLAST(j,k) \geq \ell] \geq \beta \right].$$

A value of $\alpha_{\ell,\beta}$, which is close to one, is an indicator of a high level of transitivity (with a probability of β) between the sequence similarities of protein couples. In Figure 5.12, we study the transitivity of BLAST bit-scores for different levels of confidence β. For example, in this figure, we observe that for two couples $[p_1, p_2]$ and $[p_2, p_3]$ with BLAST bit-score similarities of at least 100, the similarity of the couple $[p_1, p_3]$ is at least 91 with a probability of 0.80. In general, based on these experimental evidence, it seems reasonable to assume that there is a *pseudo transitive* relationship between the sequence similarities of real proteins.

The two main observations about (i) the relationship between sequence similarity and biological functions of protein couples, and (ii) the transitivity of BLAST bit-scores help us to design

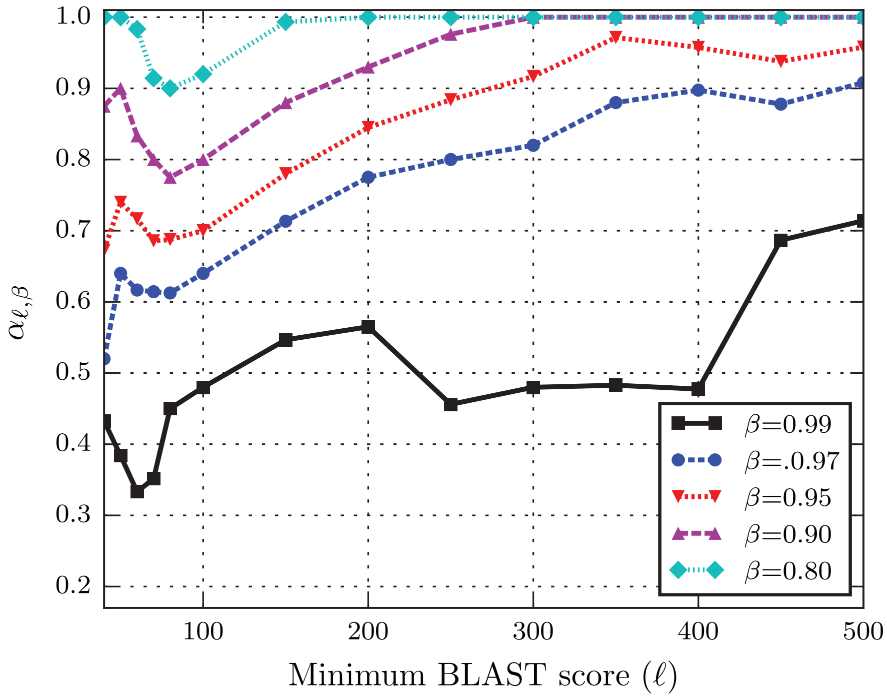


Figure 5.12 – The transitivity of BLAST bit-score similarities for real proteins. The $\alpha_{\ell,\beta}$ measure is calculated for different values of ℓ and β.

a heuristic algorithm for generating high-quality clusters (ℓ-consistent tuples) from sequence similarities.

5.5.2 Why MultiplePercolation?

The general class of PGM algorithms has been shown to be very powerful for global pairwise-network alignment problems. For example, PROPER is a state-of-the-art algorithm that uses PGM-based methods to align two networks (see Chapter 4). There are several works on the theoretical and practical aspects of PGM algorithms [40, 100, 140, 202]. In this chapter, we introduce a global MNA algorithm, as a new member of the PGM class. In this section, by using a parsimonious k-graph sampling model (as a generalization of the model from [100]), we prove that `MultiplePercolation` aligns all the nodes correctly, if initially enough number of seed-tuples are provided. We first explain the model. Then we state the main theorem. Finally, we present experimental evaluations of `MultiplePercolation` over random graphs that are generated based on our k-graph sampling model.

A Multi-graph Sampling Model

Assume that all the k networks $G_i(V_i, E_i)$ are evolved from an ancestor network $G(V,E)$ through node sampling (to model gene or protein deletion) and edge sampling (to model loss of protein-protein interactions) processes.

Definition 25 (The $Multi(G, \boldsymbol{t}, \boldsymbol{s}, k)$ sampling model)**.** Assume we have $\boldsymbol{t} = [t_1, t_2, \cdots, t_k]$ and $\boldsymbol{s} = [s_1, s_2, \cdots, s_k]$, $0 < t_i, s_i \le 1$. The network $G_i(V_i, E_i)$ is sampled from $G(V, E)$ in the following way: First the nodes V_i are sampled from V independently with probability t_i; then the edges E_i are sampled from those edges of graph G, whose both endpoints are sampled in V_i, by independent edge sampling processes with probability s_i. We define $t_{i,j} = \sqrt{t_i t_j}$ and $s_{i,j} = \sqrt{s_i s_j}$.

Definition 26 (A correctly matched tuple)**.** A tuple T is a correctly matched tuple, if and only if all the nodes in T are the same (say a node u), i.e., they are samples of a same node from the ancestor network G.

Definition 27 (A completely correctly matched tuple)**.** A correctly matched tuple T, which contains different sample of a node u, is complete if and only if for all the vertex sets $V_i, 1 \le i \le k$, if $u \in V_i$ then $V_i(T) = u$

Assume the k networks $G_i(V_i, E_i)$ are sampled from a $G(n, p)$ random graph with n nodes and average degrees of np. Now we state two main theorems that guarantee the performance of `MultiplePercolation` over the $Multi(G(n.p), \boldsymbol{t}, \boldsymbol{s}, k)$ sampling model. We first define two parameters $b_{t,s,r}$ and $a_{t,s,r}$:

$$b_{t,s,r} = \left[\frac{(r-1)!}{nt^2(ps^2)^r}\right]^{\frac{1}{r-1}} \text{ and } a_{t,s,r} = (1 - \frac{1}{r})b_{t,s,r}. \tag{5.14}$$

Theorem 28. *For $r \ge 2$ and an arbitrarily small but fixed $\frac{1}{6} > \epsilon > 0$, assume that $n^{-1} \ll p \le n^{-\frac{5}{6}-\epsilon}$. For an initial set of seed tuple $\mathcal{A}$, if $|\mathcal{A}_{i,j}| > (1+\epsilon)a_{t_{i,j},s_{i,j},r}$ for every $1 \le i, j \le k, i \ne j$, then with high probability the* `MultiplePercolation` *algorithm percolates and for the final alignment π, we have $|\pi_{i,j}| = nt_{i,j}^2 \pm o(n)$, where almost all the tuples are completely correctly matched tuples.*

Theorem 29. *For $r \ge 2$ and an arbitrarily small but fixed $\frac{1}{6} > \epsilon > 0$, assume that $n^{-1} \ll p \le n^{-\frac{5}{6}-\epsilon}$. For an initial set of seed tuple $\mathcal{A}$, if for every $1 \le i \le k$ there at least c set of $\mathcal{A}_{i,j}$, $1 \le j \le k$ and $i \ne j$, such that $|\mathcal{A}_{i,j}| > (1+\epsilon)a_{t_{i,j},s_{i,j},r}$, then with high probability the* `MultiplePercolation` *algorithm percolates and for the final alignment π, we have:*

- *Almost all the tuples $T \in \pi$ are correctly matched tuples.*
- *For a correctly matched tuple T, which contains the node u, if there are at least $k-c+1$ networks $G_i(V_i, E_1)$ such that $u \in V_i$, then T is a completely correctly matched tuple*

Note that Theorem 28 is the special case of Theorem 29 for $c = k-1$. The proofs of Theorems 28 and 29 follow from generalization of the ideas that are used to prove Theorem 16 in Chapter 3.

Experimental Results

To evaluate the performance of our algorithm by using synthetic networks, we consider $k \in \{3,4,5\}$ randomly generated networks from the $Multi(G, \boldsymbol{t}, \boldsymbol{s}, k)$ model. In these experiments, we assume that a priori a set of seed-tuples $\mathcal{A}$ ($|\mathcal{A}| = a$), with nodes from all the k networks, are given and the `MultiplePercolation` algorithm starts the alignment process from these tuples. Figures 5.13, 5.14 and 5.15 show the simulation results for these experiments. We use $r = 2$ for the `MultiplePercolation` algorithm. For each $k \in \{3,4,5\}$, the total number of correctly aligned tuples is provided. We observe that when there are enough number of tuples in the seed set, `MultiplePercolation` aligns correctly most of the nodes. We also see the sharp phase-transitions predicted in Theorems 28 and 29. According to (5.14), we need $a_{t,s,r} = 236$ correct seed-tuples to find the complete alignments for the model parameters of $n = 10^5, p = 20/n, t = 0.9$ and $s = 0.9$. We observe that the phase transitions take place very close to $a_{t,s,r} = 236$. For example, if $k = 5$, in expectation there are $nt^5 = 59049$ nodes that are present in all the five networks. From Figures 5.13 (the black curve), it is clear that `MultiplePercolation` aligns correctly almost all these nodes. Also, in expectation, there are $\binom{5}{3} nt^3(1-t)^2 = 7290$ nodes that are present in exactly three networks. Again, from Figures 5.13 (the red curve), we observe that `MultiplePercolation` correctly aligns them .

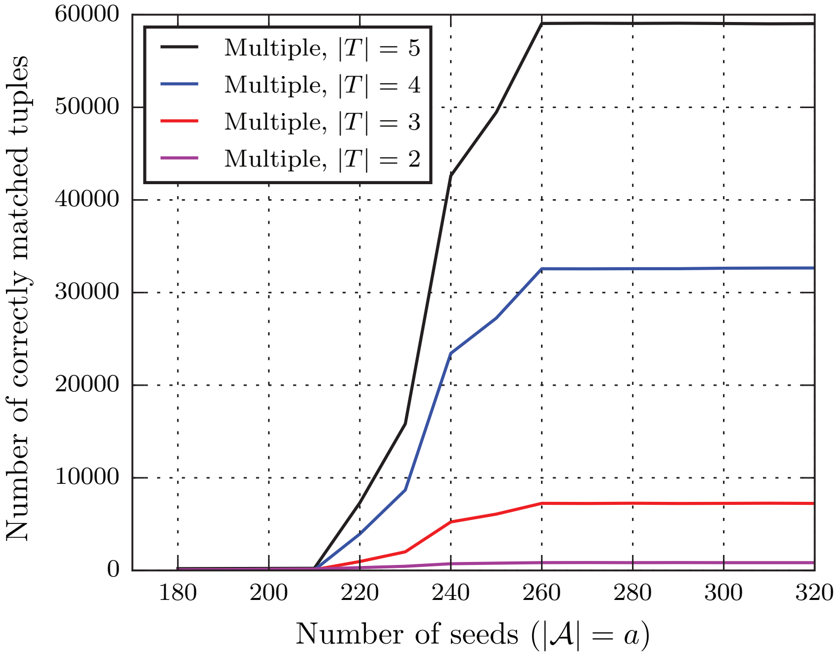


Figure 5.13 – Multiple network alignment for graphs sampled from $Multi(G, \boldsymbol{t}, \boldsymbol{s}, k)$ with parameters $k = 5, n = 10^5, p = 20/n, t = 0.9$ and $s = 0.9$. We set $r = 2$ for `MultiplePercolation`.

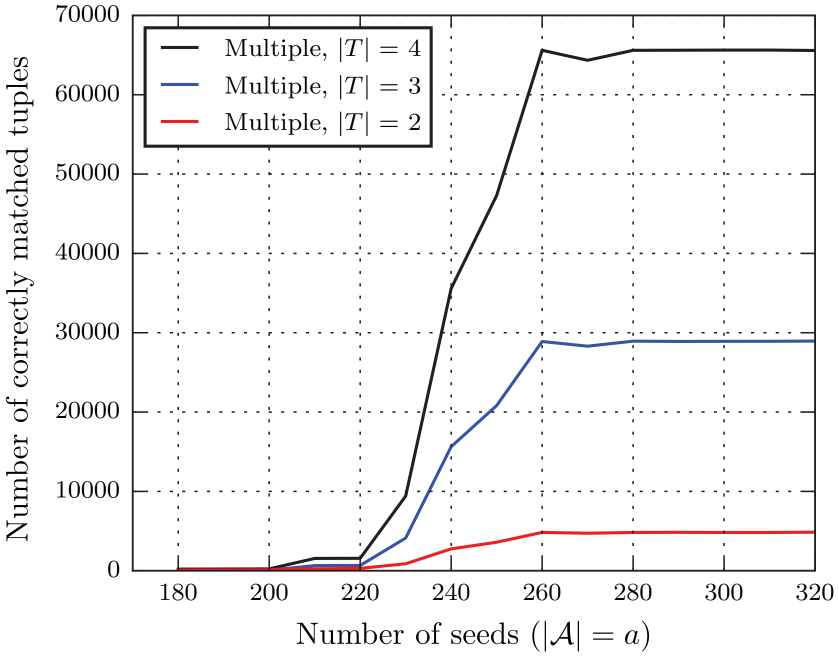


Figure 5.14 – Multiple network alignment for graphs sampled from $Multi(G, t, s, k)$ with parameters $k = 4, n = 10^5, p = 20/n, t = 0.9$ and $s = 0.9$. We set $r = 2$ for `MultiplePercolation`.

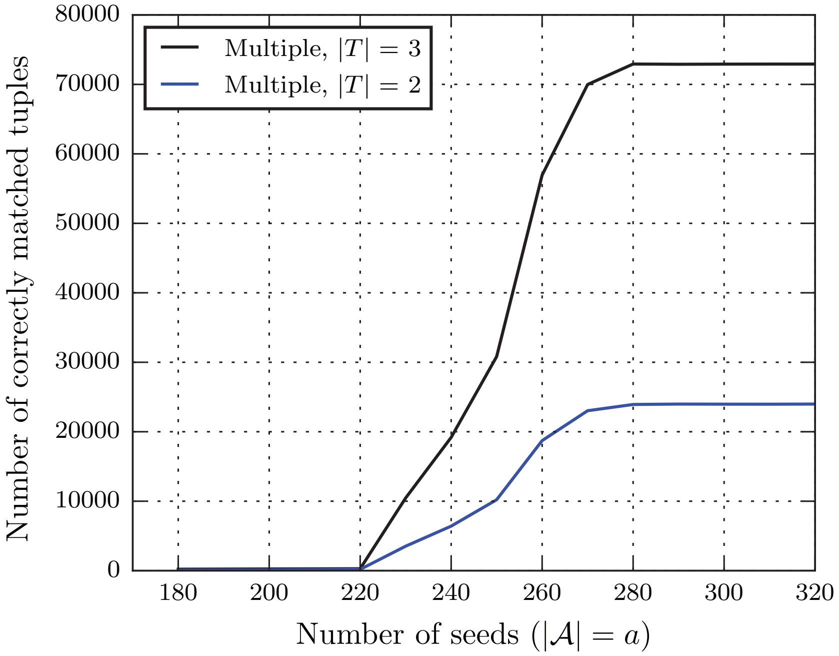


Figure 5.15 – Multiple network alignment for graphs sampled from $Multi(G, t, s, k)$ with parameters $k = 3, n = 10^5, p = 20/n, t = 0.9$ and $s = 0.9$. We set $r = 2$ for `MultiplePercolation`.

5.6 Summary

In this chapter, we have introduced a new one-to-one global multiple-network alignment algorithm, called MPROPER. The MPROPER algorithm has two main steps. In the first step (`SeedGeneration`), it uses protein sequence-similarities to generate an initial seed-set of tuples. In the second step (`MultiplePercolation`), MPROPER applies a percolation-based graph-matching algorithm to align the remaining unmatched proteins, by using only the structure of networks and the seed tuples from the first step. We have compared MPROPER with several state-of-the-art methods. We observe that MPROPER outperforms the other algorithms with respect to several measures. More specifically, MPROPER finds many consistent clusters with high d-coverage (mainly for $d = k$). Also, it outputs alignments with high structural similarity between networks, i.e., many interactions are conserved among aligned clusters.

We have studied the transitivity of sequence similarities for real proteins and have found that it is reasonable to assume a pseudo transitive relationship among them. We argue, based on this pseudo transitivity property, that the `SeedGeneration` heuristic is able to find seed tuples with high functional-similarities. In addition, we present a random-sampling model to generate k correlated networks. By using this model, we prove that `MultiplePercolation` aligns correctly (almost) all the nodes, if initially enough seed tuples are provided.

Appendix

5.A GO Annotation: Statistics

In this appendix, we look at a few statistics regarding GO annotations. GO annotations comprises three orthogonal taxonomies for a gene product: molecular-function, biological-process and cellular-component. This information is captured in three different directed acyclic graphs (DAGs). The roots (the most general annotations for each category) of these DAGs are:

- GO:0003674 for molecular function annotations
- GO:0008150 for biological process annotations
- GO:0005575 for cellular component annotations

For information content of each GO term, we use the SWISS-PROT-Human proteins, and counted the number of times each concept occurs. Information content is calculated based on the following information:

- Number of GO terms in the dataset is 26831.
- Number of annotated proteins in the dataset is 38264085.
- Number of experimental GO terms in the dataset is 24017.
- Number of experimentally annotated proteins in the dataset is 102499.

Table 5.8 provides information related to different categories of GO annotations for the five networks we used in our experiments.

Next we report the number of experimentally annotated proteins (at the cut-off level 5 of DAGs) in each network:

- C. elegans: 1544 out of 4950 proteins (31.2 %).

Table 5.8 – Statistics for experimental GO annotations.

GO type	#GO	#proteins	Avg. #GO
All	20738	28896	49.47
Biological process	14876	20723	48.21
Molecular function	3938	21670	7.84
Cellular component	1924	21099	12.35

- D. melanogaster: 4653 out of 8532 proteins (54.5 %).
- H. sapiens: 10929 out of 19141 proteins (57.1 %).
- M. musculus: 7150 out of 10765 proteins (66.4 %).
- S. cerevisiae: 4819 out of 6283 proteins (76.7 %).

The probabilities of sharing at least one GO term (at the cut-off level 5) for clusters of size two to five, when all the proteins of a cluster are annotated, are as follows:

- clusters of size 2: 0.215
- clusters of size 3: 0.042
- clusters of size 4: 0.009
- clusters of size 5: 0.002

Also, the probabilities of sharing at least one GO term (at the cut-off level 5) for clusters of size two to five, when at least two proteins from each cluster are annotated, are as follows:

- clusters of size 2: 0.215
- clusters of size 3: 0.167
- clusters of size 4: 0.120
- clusters of size 5: 0.081

In Figure 5.16, the total number of annotated proteins, at different cut-off levels, are shown. Also, the number of GO terms and the average number of GO terms for each annotated protein, at different cut-off levels, are shown in Figures 5.17 and 5.18, receptively.

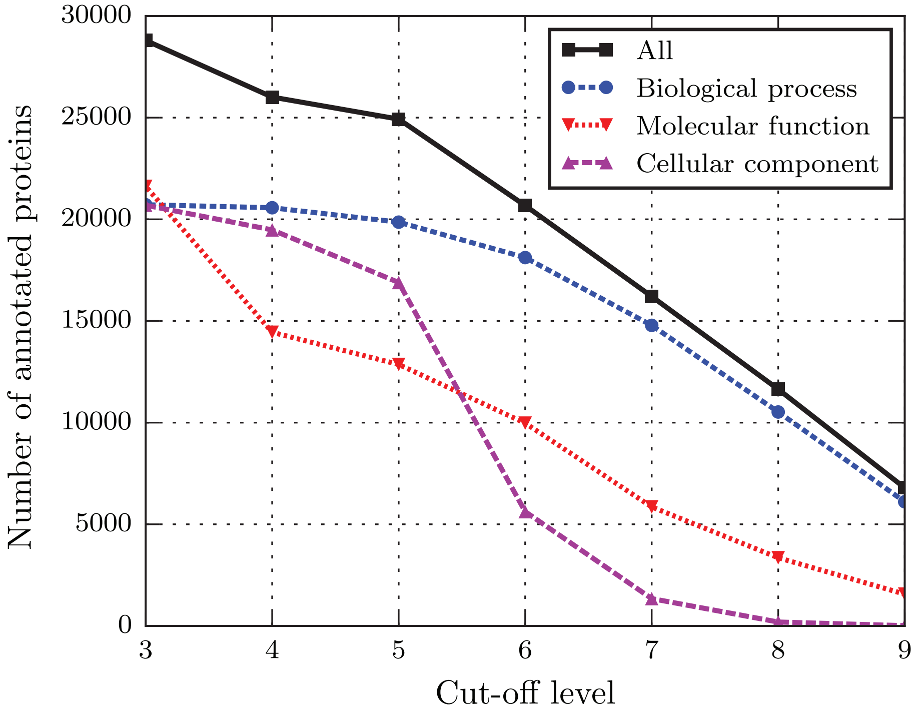


Figure 5.16 – Number of annotated proteins for different cut-off levels

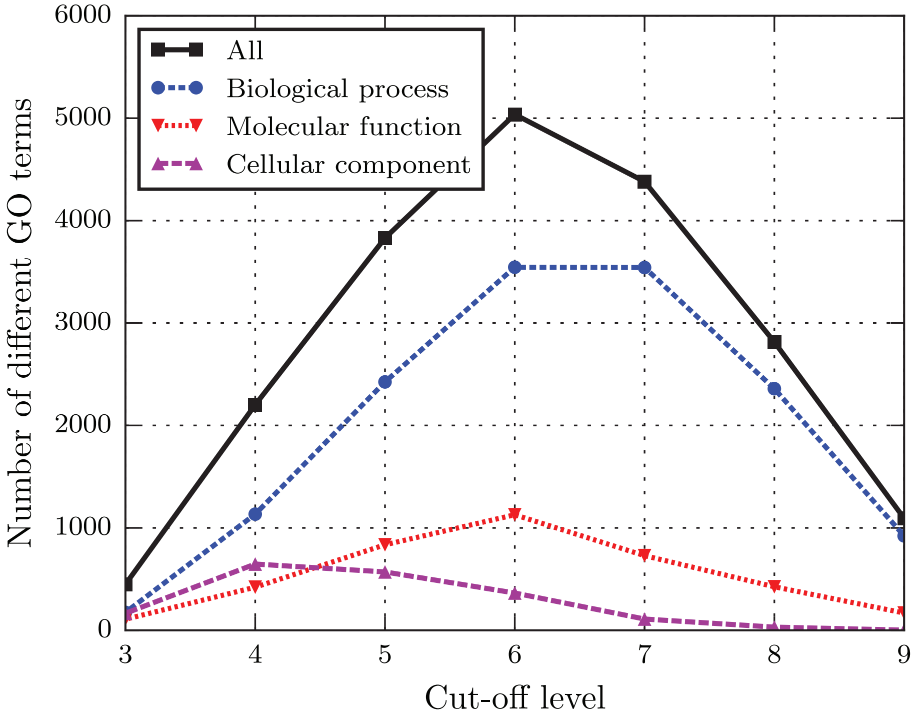


Figure 5.17 – Number of different GO terms for different cut-off levels

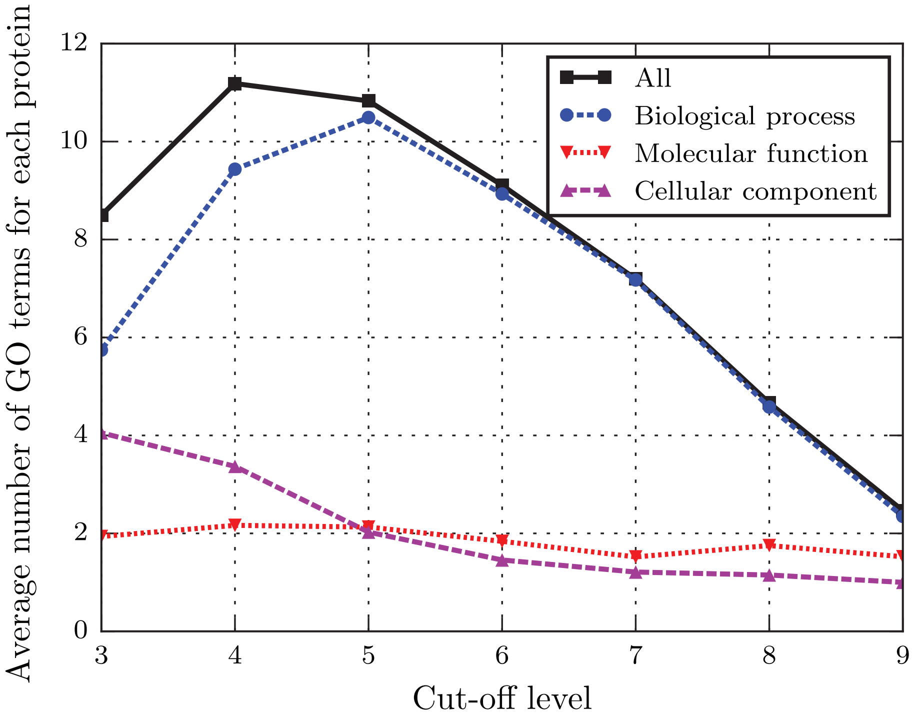


Figure 5.18 – Average number of GO terms for each annotated protein for different cut-off levels

6 IsoRank Node Similarities

As we discussed in the previous chapters, the alignment of protein-protein interaction (PPI) networks has many applications, such as the detection of conserved biological network motifs, the prediction of protein interactions, and the reconstruction of phylogenetic trees [57, 59, 111]. IsoRank is one of the first global network-alignment algorithms [119, 175, 176], where the goal is to match all (or most) of the nodes of two PPI networks. The IsoRank algorithm first computes a pairwise node-similarity metric; it then, based on this metric, generates an alignment between the two node sets. The metric is a convex combination of a structural-similarity score (with weight α) and an extraneous amino-acid sequence-similarity score for two proteins (with weight $1-\alpha$).

In this chapter[1], we make two contributions. First, we show that when IsoRank similarity depends only on network structure ($\alpha = 1$), the similarity of two nodes is only a function of their degrees. In other words, IsoRank similarity is invariant to any network rewiring that does not affect the node degrees. This result suggests a reason for the poor performance of IsoRank in structure-only ($\alpha = 1$) alignment. Second, using ideas from [14, 71], we develop an approximation algorithm that outperforms IsoRank (including recent versions with better scaling, e.g., [24]) by several orders of magnitude in time and memory complexity, despite only a negligible loss in precision.

6.1 Problem Definition

We first define the IsoRank algorithm as given in [176]. Assume we are given two networks $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ with $|V_i| = n_i$ and $|E_i| = m_i$. Let $N_{i,u}$ represent the neighbours of node u in graph i and $d_{i,u} = |N_{i,u}|$ is its degree. Also, assume $\boldsymbol{b}$ is the doubly indexed vector of BLAST sequence-similarities of proteins, i.e., $\boldsymbol{b}[u,v]$ is the sequence similarity between proteins $u \in V_1$ and $v \in V_2$. The vector $\boldsymbol{e} = \frac{\boldsymbol{b}}{|\boldsymbol{b}|_1}$ is the normalized vector of sequence similarity scores. Also, $\boldsymbol{P}$ is a $n_1 n_2 \times n_1 n_2$ square matrix, where $\boldsymbol{P}[u_1, u_2][v_1, v_2]$ refers to the entry at

---

[1] The material of this chapter is based on [98].

row (u_1, u_2) and column (v_1, v_2).[2] The elements of $\boldsymbol{P}$ are defined as follows:

$$\boldsymbol{P}[u_1, u_2][v_1, v_2] = \begin{cases} \frac{1}{d_{1,v_1} d_{2,v_2}}, & \text{if } (u_1, v_1) \in E_1 \text{ and } (u_2, v_2) \in E_2. \\ 0, & \text{otherwise} \end{cases}$$

Problem 1 (IsoRank similarity problem [176])**.** *Find the vector* $\boldsymbol{r}$ *from*

$$\boldsymbol{r} = \alpha \boldsymbol{P} \boldsymbol{r} + (1 - \alpha)\boldsymbol{e}, \tag{6.1}$$

for $0 \le \alpha \le 1$. *If we assume* $|\boldsymbol{r}|_1 = 1$ *then the problem is equivalent to finding* $\boldsymbol{r}$ *from*

$$\boldsymbol{r} = \left(\alpha \boldsymbol{P} + (1 - \alpha)\boldsymbol{e}\mathbb{1}^T\right)\boldsymbol{r}.$$

The first step of the IsoRank algorithm is to compute $\boldsymbol{r}$, where $\boldsymbol{r}[u, v]$ corresponds to the similarity between nodes $u \in G_1$ and $v \in G_2$. The value of $\boldsymbol{r}[u, v]$ can be interpreted as a likelihood such that the node u aligns with the node v based on structural and sequence similarities. The second step is to construct an alignment based on the similarity vector $\boldsymbol{r}$. The original IsoRank [175, 176] proposes two approaches for alignment: (i) solving the maximum-weight bipartite graph matching, where edge weights are elements of $\boldsymbol{r}$; and (ii) greedily aligning the most similar nodes first and removing them, then matching the most similar among the remaining and so on. [175, 176]. The greedy method is much faster and has shown slightly better alignment quality in many cases [175].

6.2 Structural IsoRank Depends Only on Degrees

In this section, we show that structure-only IsoRank ($\alpha = 1$) depends only on node degrees and does not use any other structural information from the two graphs $G_{1,2}$. This is a surprisingly weak dependence on the network structure, in the sense that any rewiring that conserves node degrees does not affect the alignment produced by IsoRank.

We first define the tensor product (Kronecker product) of two graphs.

Definition 30 (Tensor product of two graphs [74])**.** The tensor product $G_1 \times G_2$ of two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ is the graph $G(V, E)$ defined as follows:

- $V = V_1 \times V_2$ is the Cartesian product of the two sets $V_{1,2}$.
- There is an edge between (u_1, u_2) and $(v_1, v_2) \in V$ (i.e., $((u_1, u_2), (v_1, v_2)) \in E$) if and only if $(u_1, v_1) \in E_1$ and $(u_2, v_2) \in E_2$.

Lemma 31. *The IsoRank-similarity problem is equivalent to the PageRank problem [113] over the graph* $G = G_1 \times G_2$ *with the teleportation constant* α *and the preference vector* $\boldsymbol{e}$.

---

[2] Both the rows and columns are doubly indexed.

Proof. Call $\boldsymbol{A}$ the adjacency matrix of graph G. Based on the definition of tensor product of two graphs, it is easy to show that the degree of node $(v_1, v_2) \in V$ is $d_{1,v_1} d_{2,v_2}$. Then the PageRank problem over graph G with preference vector $\boldsymbol{e}$ is to find $\boldsymbol{r}'$ such that

$$\boldsymbol{r}' = \alpha \boldsymbol{D}^{-1} \boldsymbol{A} \boldsymbol{r}' + (1-\alpha)\boldsymbol{e}, \tag{6.2}$$

where $\boldsymbol{D}$ is the diagonal matrix of weighted degrees. Again, it is straightforward to see that $\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{A}$. From these two facts, we conclude that $\boldsymbol{r} = \boldsymbol{r}'$. □

Lemma 32. *For the case* $0 < \alpha < 1$ *we have*

$$\boldsymbol{r} = (1-\alpha)(\boldsymbol{I} - \alpha\boldsymbol{P})^{-1}\boldsymbol{e}.$$

Proof. The equation is simply derived from (6.1). We need only show that $\boldsymbol{I} - \alpha\boldsymbol{P}$ is non-singular. To prove this, note that $\boldsymbol{I} - \alpha\boldsymbol{P}$ is a strictly diagonally dominant matrix. From the Levy-Desplanques theorem [79], we know that a strictly diagonally dominant matrix is non-singular. □

Next, we explain how to compute $\boldsymbol{r}$ efficiently for three different cases (i) $\alpha = 0$ (ii) $\alpha = 1$ and (iii) $0 < \alpha < 1$.

For the case $\alpha = 0$, the trivial answer is $\boldsymbol{r} = \boldsymbol{e}$. $\boldsymbol{r}$

Lemma 33. *For the IsoRank-similarity problem with* $\alpha = 1$*, we have*

$$\boldsymbol{r} = \left[\frac{d_{1,1}d_{2,1}}{m}, \cdots, \frac{d_{1,u}d_{2,v}}{m}, \cdots, \frac{d_{1,n_1}d_{2,n_2}}{m}\right] \tag{6.3}$$

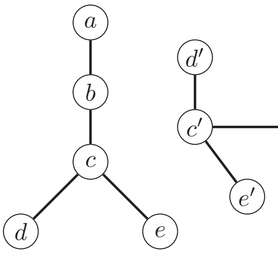
where $m = \sum_{i\in V_1} \sum_{j \in V_2} d_{1,i} d_{2,j}$.

Proof. In this case, the IsoRank-similarity problem is equivalent to the PageRank problem over the undirected graph G with $\boldsymbol{e} = 0$. It is easy to show that the vector $\boldsymbol{r}$ is the steady-state probability distribution of a random walk over G. It is a well-known result that this probability distribution is proportional to the degree of each node [125]. The lemma follows from the fact that the degree of a node $(v_1, v_2) \in V$ is $d_{1,v_1} d_{2,v_2}$ and the elements of $\boldsymbol{r}$ should sum to one. □

From Lemma 33, we conclude that when IsoRank uses only the structural properties of the two input graphs $G_{1,2}$ (i.e., $\alpha = 1$), the similarity of two nodes $u \in G_1$ and $v \in G_2$ is only a function of their degrees $d_{1,u}$ and $d_{2,v}$. This means that, in this case, IsoRank matches nodes only based on the product of their degrees. In particular, the matching generated by the greedy approach is as follows: (i) The node with highest degree in G_1 is matched to the highest degree node in G_2; then (ii) the unmatched nodes with the second highest degrees in the two graphs are matched, and so on.

Example 34. We illustrate this concept by using an example taken from [176], see Figure 6.1. The equations are for the case where $\alpha = 1$. The goal of the IsoRank-similarity problem is to find the values of r_{ij}. It is easy to see that the product of the degrees of the nodes (as stated by Lemma 33) is the non-trivial answer for this set of equations, e.g., if we have $r_{bb'} = 2 \times 2 = 4, r_{ac'} = 3, r_{ca'} = 3, r_{aa'} = 1$ and $r_{cc'} = 9$, then

$$r_{bb'} = \frac{1}{3} r_{ac'} + \frac{1}{3} r_{ca'} + r_{aa'} + \frac{1}{9} r_{cc'}.$$



$\boldsymbol{r}$

	a'	b'	c'	d'	e'
a	0.0312		0.0937		
b		0.1250		0.0625	0.0625
c	0.0937		0.2812		
d		0.0625		0.0312	0.0312
e		0.0625		0.0312	0.0312

$$r_{aa'} = \tfrac{1}{4} r_{bb'} \qquad r_{bb'} = \tfrac{1}{3} r_{ac'} + \tfrac{1}{3} r_{ca'} + r_{aa'} + \tfrac{1}{9} r_{cc'} \qquad r_{dd'} = \tfrac{1}{9} r_{cc'}$$

$$r_{cc'} = \tfrac{1}{4} r_{bb'} + \tfrac{1}{2} r_{be'} + \tfrac{1}{2} r_{bd'} + \tfrac{1}{2} r_{eb'} + \tfrac{1}{2} r_{db'} + r_{ee'} + r_{ed'} + r_{de'} + r_{dd'}$$

Figure 6.1 – A small example from [176] of the IsoRank similarity problem for the case $\alpha = 1$.

We performed experiments to confirm the invariance of IsoRank to degree-conserving rewiring in the case $\alpha = 1$. We generated two correlated graphs $G_{1,2}$ by using the random bigraph model from [99, 100], and compute the IsoRank-similarity vector with $\alpha = 1$. We randomly rewire some edges from both $G_{1,2}$ such that node degrees are preserved (using the method from [130]), and then align the two rewired graphs using the IsoRank implementation of the GraphM package [72].[3] This experiment confirmed Lemma 33. In conclusion, we observe that output of IsoRank with $\alpha = 1$ is only a function of node degrees and is otherwise independent of graph structure.

Corollary 35. *For $n = \max(|V_1|, |V_2|)$ and $\alpha \in \{0, 1\}$, we can compute the IsoRank vector in n^2 steps. Also, we can compute the similarity between any two nodes in $O(1)$.*

Note that in (6.3) the normalizing constant $m = \sum_{i \in V_1} \sum_{j \in V_2} d_{1,i} d_{2,j} = \sum_{i \in V_1} d_{1,i} \sum_{j \in V_2} d_{2,j} = 4|E_1||E_2|$ can be computed in $O(1)$, assuming the total sizes of the edge sets are available.

Corollary 36. *For $n = \max(|V_1|, |V_2|)$ and $\alpha = 1$, we can find the output of greedy IsoRank in $O\big(n \log n\big)$ steps.*

---

[3] To the best of our knowledge, this package is currently the most faithful implementation of the IsoRank algorithm, which is why we used it for this experiment.

Corollary 36 is because we can separately order the nodes in each of the graphs $G_{1,2}$ based on node degrees, then we can match the two lists.

6.3 Fast Approximate IsoRank

For the case $0 < \alpha < 1$, we can use the results of [14, 71] to approximate node-pair similarities efficiently. From the result of Lemma 31, we know that the IsoRank-similarity problem is equivalent to the PageRank problem over an undirected graph. Graham and Zhao [71] designed an approximate algorithm for solving the PageRank problem over undirected graphs with tight error bounds. Their algorithm is an improved version of the algorithm from [14].

Assume $\boldsymbol{e}[u,v]$ is the normalized sequence-similarity between two nodes $u \in G_1$ and $v \in G_2$. The `SharpApproximateIsoRank` algorithms returns vector $\tilde{\boldsymbol{r}}$ as the approximation of $\boldsymbol{r}$, where $\tilde{\boldsymbol{r}}[u,v]$ is the (approximate) total similarity between u and v. Algorithm 9 describes `SharpApproximateIsoRank`.[4]

Algorithm 9: `SharpApproximateIsoRank`$(\boldsymbol{e}, \alpha, \epsilon)$

```
$\beta \leftarrow \frac{1-\alpha}{\alpha}, \epsilon' \leftarrow 1, \tilde{e} \leftarrow \boldsymbol{e}$ and $\tilde{\boldsymbol{r}} \leftarrow 0$ ;
while $\epsilon' > \epsilon$ do
    $\epsilon' \leftarrow \epsilon'/2$;
    $\tilde{\boldsymbol{r}}', \tilde{e}' \leftarrow$ ApproximateIsoRank$(\tilde{e}, \beta, \epsilon')$ ;
    $\tilde{\boldsymbol{r}} \leftarrow \tilde{\boldsymbol{r}} + \tilde{\boldsymbol{r}}'$ and $\tilde{e} \leftarrow \tilde{e}'$ ;
```

Algorithm 10: `ApproximateIsoRank`$(\boldsymbol{e}, \beta, \epsilon)$

```
$\tilde{\boldsymbol{r}} \leftarrow 0$ and $\tilde{e} \leftarrow \boldsymbol{e}$ ;
while there exists at least a pair $(u,v) \in V$ such that $\tilde{e}[u,v] \geq \epsilon d_{1,u} d_{2,v}$ do
    pick any pair $(u,v) \in V$ such that $\tilde{e}[u,v] \geq \epsilon d_{1,u} d_{2,v}$;
    $\tilde{\boldsymbol{r}}, \tilde{e} \leftarrow$ Push$((u,v), \tilde{\boldsymbol{r}}, \tilde{e})$;
return $\tilde{\boldsymbol{r}}$ and $\tilde{e}$;
```

Algorithm 11: `Push`$\big((u,v), \tilde{\boldsymbol{r}}, \tilde{e}, \beta\big)$

```
$\tilde{\boldsymbol{r}}' \leftarrow \tilde{\boldsymbol{r}}$ and $\tilde{e}' \leftarrow \tilde{e}$ ;
$\tilde{\boldsymbol{r}}'[u,v] \leftarrow \tilde{\boldsymbol{r}}[u,v] + \frac{\beta}{2+\beta}\tilde{e}[u,v]$ ;
$\tilde{e}'[u,v] \leftarrow \frac{1}{2+\beta}\tilde{e}[u,v]$ ;
for each pair $(u',v')$ such that $(u',u) \in E_1$ and $(v',v) \in E_2$ do
    $\tilde{e}'[u',v'] \leftarrow \tilde{e}[u',v'] + \frac{\tilde{e}[u,v]}{(2+\beta)d_{1,u}d_{2,v}}$ ;
return $\tilde{\boldsymbol{r}}'$ and $\tilde{e}'$;
```

Lemma 37. *For the number of edges m in the product graph G, we have* $m \leq \min(2|E_1|D_2, 2|E_2|D_1)$, *where* $D_{1,2}$ *are the maximum degrees in the two networks.*

---

[4] The same as `SharpApproximatePR` from [14].

Proof. From $m = \sum_{i\in V_1}\sum_{j\in V_2} d_{1,i}d_{2,j}$ we conclude that $m \le \sum_{i\in V_1} d_{1,i}D_2 \le 2|E_1|D_2$ and $m \le 2|E_2|D_1$. □

Theorem 38. *The* SharpApproximateIsoRank *algorithm reruns the two vectors* $\tilde{\boldsymbol{r}}$ *and* $\tilde{\boldsymbol{e}}$ *such that*

$$\tilde{\boldsymbol{r}} = \alpha \boldsymbol{P}\tilde{\boldsymbol{r}} + (1-\alpha)(\boldsymbol{e} - \tilde{\boldsymbol{e}}),$$

where $|\frac{\tilde{e}(u,v)}{d_{1,u}d_{2,v}}| \le \epsilon$ *for all pairs* $(u,v) \in V$. *The running time of the algorithm is*

$$O\left(\frac{(1+\alpha)\min(|E_1|D_2, |E_2|D_1)\log(1/\epsilon)}{1-\alpha}\right).$$

Proof. This theorem is the direct result of Theorem 2 from [71] and Lemmas 31 and 37. □

Corollary 39. *For* $n = \max(|V_1|, |V_2|)$ *and given constants* $c > 0, 0 < \alpha < 1$, *we can approximate the IsoRank vector* $\boldsymbol{r}$ *in* $O(n^3\log(n))$ *steps with* $\epsilon = \Omega(n^{-c})$.

Note that the time complexity of the original IsoRank algorithm for computing the approximate IsoRank vector is $O(n^4)$. Corollary 39 gives the worst case performance. For many real (sparse) biological networks, time complexity is much smaller.

6.4 Simulation Result

We compared the performance of the original IsoRank algorithm[5] with our implementation of the `SharpApproximateIsoRank` algorithm on aligning PPI networks of the five major eukaryotic species. Table 6.1 provides a brief description of the PPI networks that are extracted from the IntAct database [1, 76]. The amino-acid sequences of proteins for each species are collected in the FASTA format from the UniProt database [4, 15]. The BLAST bit-score similarities [13] are calculated using these amino acid sequences. The IsoRank algorithm[6] took 13 hours and 31 minutes to perform all ten pairwise alignments between species from Table 6.1.[7] The `SharpApproximateIsoRank` algorithm performed these ten alignments in 53 minutes for $\epsilon = 10^{-12}$, 59 minutes for $\epsilon = 10^{-13}$, and one hour and 11 minutes for $\epsilon = 10^{-14}$. For larger networks, the relative advantage of `SharpApproximateIsoRank` would be even more pronounced.

---

[5] The official IsoRank implementation from http://groups.csail.mit.edu/cb/mna/isobase/

[6] Run with parameters `-K 50 -thresh 1e-5 -alpha 0.9 -maxveclen 1000000`. Note that in this version of IsoRank the parameter `-maxveclen` sets a limit on the number of non-zero entries in the IsoRank vector, e.g., 10^6 out of $\approx 2\times 10^8$ possible entries between H. sapiens and M. musculus in this example.

[7] The GraphM package [72] took several days to finish these alignments.

Table 6.1 – PPI networks of five major eukaryotic species from the IntAct molecular interaction database [1, 76].

species	#nodes	#edges	Avg. deg.
C. elegans	4950	11550	4.67
D. melanogaster	8532	26289	6.16
H. sapiens	19141	83312	8.71
M. musculus	10765	22345	4.15
S. cerevisiae	6283	76497	24.35

6.5 Summary

We have shown that the IsoRank node-similarity metric has a peculiar structure, in that the network (structural) similarity depends only on the nodes' degrees and not on the actual edge set of the two networks. It appears that this fact has not been noted before and provides some insight into its relatively poor performance for $\alpha = 1$. We have also shed light on the relationship between the IsoRank and PageRank problems. The IsoRank-similarity problem is in fact equivalent to applying PageRank over the Kronecker product of the two graphs. This equivalence enables us to apply ideas for efficient PageRank approximation algorithms to the IsoRank-similarity problem, with significant gains in runtime and memory complexity.

7 Modeling and Mitigating Epidemics

In this chapter[1], we consider an important application area of network mining in public health. We focus on human-mediated epidemics.[2] We develop and model strategies for mitigating an epidemic in a large-scale dynamic contact network. In particular, we explore new mitigation methods based on a realistic modeling of epidemics, while we consider mobility and contact network of individuals in a geographical area. We argue that taking advantage of mobile technology opens up many new opportunities for mitigating the spread of an epidemic. Importantly, mobile technology is unique in that it enables the *personalization* of countermeasures through precise measurements at the individual level, as well as individualized recommendations.

Human mobility and contacts among population are crucial factors that enable the epidemic to travel and spread geographically. At a high level, our mitigation approach is to maintain deliberate contacts and to rewire the accidental ones. The idea is to weaken the links that, in the contact network, form the path through which the epidemic spreads. By changing the network structure, we seek to decelerate the dynamics and drive the epidemic down to a sub-critical level. We show that the combination of information extracted from mobile data (e.g., call-data records) and subsequent personalization of prevention advice opens up novel ways of mitigating an epidemic. We envision a mobile service that sends recommendations that encourage individuals to adapt their behavior, for example, by delaying or canceling a trip. More generally, we formulate subtle, precise and minimally restrictive personalized behavioral rules that, if followed even partially, will have a positive global effect on an epidemic.

In Section 7.1, using a human-contacts network, we build a time-dependent model of human mobility; this enables us to accurately capture population movements across a geographical area. These mobility patterns then power the core of our epidemic model, which enables us to analyze epidemic outbreaks at the level of single individuals in Section 7.2. Beyond these models, our main contribution is to foster the idea of a mobile service that sends personal recommendations in order to help mitigate an epidemic. In Section 7.3, we present several

---

[1] The material of this chapter is based on [93].

[2] Transmitted by human contact,e.g., influenza

concrete micro-measures and discuss their potential. In Section 7.4, we empirically evaluate their effectiveness by using our epidemic model and providing some insights into further research directions. In Section 7.5, we conclude this chapter.

7.1 Mobility Model

The spread of epidemics depends greatly on the mobility of infected individuals, and on the locations where they interact with other individuals. In this section, we present a realistic, data-driven mobility model that is an essential tool for simulating realistic epidemic propagation. Our mobility model is build upon the call-data records (CDRs) provided by the *Data for Development* (D4D) challenge[3] organized by France Telecom-Orange, a global telecommunications operator [31]. The D4D dataset contains anonymized data gathered from 2.5 billion calls and SMS exchanges made by 5 million users in Ivory Coast over a period of five months, from December 2011 to April 2012. In this dataset[4], for each CDR the following information is provided:

```
time, caller id, call duration, antenna id
```

This dataset contains high-resolution trajectories of 50,000 randomly selected individuals over ten two-week periods. For each period, 50,000 of the customers are randomly selected and then assigned anonymized identifiers (i.e., `caller id`) to protect the privacy of users. The records are composed of the identifiers of the antennas (i.e., `antenna id`) from which individuals made phone calls or sent SMSs over a two-week period. Also, for each antenna, we have the corresponding, but slightly blurred, geolocation.

7.1.1 The Features of Mobility Model

A precise yet tractable model for population mobility should take into account certain microscopic aspects at the individual level, and still scales up to millions of individuals to model the propagation of epidemics. Moreover, it should capture the main differences between the mobility of different groups of individuals, where a group is constituted of individuals exhibiting similar mobility profiles. To construct a mobility model that fulfils these requirements, our intuition is as follows: The home location of individuals strongly shapes their mobility patterns because the places they visit regularly, e.g., their workplaces, schools or the shopping centers, depend on the proximity to their home. Typically, we expect the most visited location (home) and the second most visited location (school, university or work) to be geographically close to each other. In addition to this geographical aspect, mobility is strongly time-dependent: Individuals commute between home and work during the weekdays, with a substantial change in their travel behavior during the weekends.

---

[3] See: http://www.d4d.orange.com/

[4] In order to build our model from data, we use `SET2`, one of the four datasets from D4D challenge.

Building on this, we make the assumption that the individuals that share the same home-location exhibit a similar time-dependent mobility pattern. Therefore, we construct a location and time-based mobility model that depends on the variables presented in Table 7.1. The conditional distribution of the location $X(n)$ of user u depends on her home antenna $a_{\text{home}}(u)$, but also on the time of the visits $(h^k(n), w(n))$:

$$p\left(X(n)|u, t(n)\right) = p\left(X(n)|h^k(n), w(n), a_{\text{home}}(u)\right). \tag{7.1}$$

First, we choose the time resolution $k = 3$, in order to divide the day into 3 distinct periods: Morning (6 am to 1 pm), afternoon (1 pm to 8 pm) and night (8 pm to 6 am). Second, conditioning on the parameter $w(n)$ enables us to distinguish between weekdays and weekends. Finally, the home antenna $a_{\text{home}}(u)$ of user u is defined as the most visited antenna during the night period. Consequently, given the period of the day, the day type and the home antenna of user u, the distribution of the location that she might visit (7.1) is a multinomial distribution with $|\mathcal{A}|$ categories.

Table 7.1 – List of the definition and domain of the variables relative to user u, as well as those describing her n^{th} visit.

Definition	Domain	Explanation
$\mathcal{A} = \{1, \ldots, 1231\}$	-	Set of antennas
$\mathcal{SP} = \{1, \ldots, 255\}$	-	Set of sub-prefectures
k	$\mathbb{N}$	Time resolution
$sp_{\text{home}}(u)$	$\mathcal{SP}$	Home sub-prefecture for user u
$a_{\text{home}}(u)$	$\mathcal{A}$	Home antenna for user u
$X(n)$	$\mathcal{A}$	Antenna
$t(n)$	$\mathbb{N}$	Absolute time
$h^k(n)$	$\{1, \ldots, k\}$	Period of the day
$d(n) = \text{day}(t(n))$	$\{1, \ldots, 7\}$	Day of the week
$w(n) = \text{weekday}(t(n))$	$\{0, 1\}$	Day type: weekday or weekend

7.1.2 Learning and Evaluating Mobility Models

To avoid having to deal with users whose location samples are very sparse, we consider only the users who visited more than 1 antenna and made on average more than 1 call per day. In order to evaluate the realism of our mobility model, we separate the data into two parts: For each user, we put 90% of the calls in the training set and the remaining 10% in the test set. First, by using a maximum-likelihood estimator, we learn a mobility model from the training set. Then, we evaluate the accuracy of our mobility model by computing the average log-likelihood of the calls in the unseen test set. The average log-likelihood reflects how well our model generalizes to unseen data. As the test set might contain some locations not visited by a given class of users in the training set, the maximum-likelihood estimate of the distribution (7.1) assigns a zero probability to these observations. We cope with this by assuming that the distribution (7.1) is a multinomial distribution drawn from an exchangeable Dirichlet distribution, which implies that the inferred distribution (7.1) is a random variable drawn from a posterior distribution conditioned on the training data. A more detailed description of this smoothing procedure is given by Blei et al. [29].

We tested several variants of mobility models by varying their structure and parameters (time resolution, day of the week, etc). To have three representative baseline models for comparison, we choose three predictors out of the several variants we tested.

Time-based Mobility (TM) The first baseline model is a time-based mobility (TM) model defined by

$$p\left(X(n)|u,t(n)\right) = p\left(X(n)|h^k(n),w(n)\right), \tag{7.2}$$

where all mobile-phone users exhibit the same time-dependant mobility pattern.

Markov Chain (MC) The second baseline is a location-dependent first order Markov chain (MC) defined by

$$p\left(X(n)|u,t(n),X(n-1),\ldots,X(0)\right) = p\left(X(n)|X(n-1)\right), \tag{7.3}$$

where the current location of a user depends only the location she visited just before.

Sub-Prefecture Mobility (SPM) The third baseline is a time and sub-prefecture dependant mobility model (SPM) defined by

$$p\left(X(n)|u,t(n)\right) = p\left(X(n)|h^k(n),w(n),sp_{\text{home}}(u)\right), \tag{7.4}$$

where the home of a user is represented by a sub-prefecture instead of an antenna. This implies a more important aggregation of users, where two users who share the same home sub-prefecture, have the same mobility pattern.

The experimental results are shown in Table 7.2. The first order Markov chain (MC) performs the worst. This is not surprising as the time difference between two call records varies greatly, ranging from a few minutes to a few days. The location associated with a call made in the past few hours or days does not necessarily affect the current location. As the location data is sporadic, it is not surprising that any model that learns from transitions performs poorly and is outperformed by time-based models. Our model performs the best; and by comparing it to the time-based model (TM), we realise that knowing the home-locations of users enhances the predictive power of the mobility model. Moreover, the granularity of home locations is crucial: Our model significantly outperforms the sub-prefecture dependent mobility model

Table 7.2 – Log-likelihood of the unseen data from the test set. Our mobility model significantly outperforms the baseline models since its predictive power, with respect to the test set, is higher.

Mobility model	Average log-likelihood
Our model	**-1.07**
SPM	-1.67
TM	-2.9
MC	-6.49

because it has a finer granularity of the home locations.

A realistic mobility model is an essential building block of a realistic epidemic-propagation model because mobility drives population flows between regions, hence the geographical proximity between individuals. In the next section, we introduce the model we use to simulate an epidemic propagation.

7.2 Epidemic Model

Building up on the mobility, this section introduces our epidemic model. It is based on a discretized, stochastic version of the *SIR* model [106]; Tables 7.3 and 7.4 provide an overview of the parameters and quantities used throughout the section. We assume that the size of the population (N individuals) remains constant—there are no births or deaths, a reasonable assumption if the time horizon is limited to at most a few months. Under the *SIR* model, an individual can be either *susceptible* to the disease, *infective*, or *recovered* from the disease and immunized against further infections.[5] We assume that most of the population is initially susceptible, except for a small number of infective individuals that form the seed of the epidemic. Individuals successively go through the susceptible, infective and recovered states; a desirable outcome would have many individuals stay susceptible without ever becoming infective. The basic *SIR* model assumes a *random mixing* of the whole population: any given individual meets any other one with a uniform probability. In our model, we relax this strong assumption by taking into account the *mobility*. We spread the population across M regions; each region bears its own *SIR* process where the corresponding meta-population mixes at random. These regional processes are independent and isolated, and the only way the epidemic crosses regional boundaries is through human mobility [102]. In summary, regional interactions take place uniformly at random, whereas global interactions are shaped by the individuals' mobility.

---

[5] In the literature, this state is sometimes known as *removed*. The important point is that they do not participate in the epidemic anymore.

Table 7.3 – Parameters of the epidemic model.

N	total population
M	number of regions
N_i^*	initial population of region i, where $i \in \{1,\ldots,M\}$
L	number of different mobility classes
β	contact probability
g	recovery probability

Table 7.4 – Notation for various quantities related to the epidemic.

c_l	mobility class l, where $l \in \{1,\ldots,L\}$
$\mathbf{S}_i$	distribution of the number of susceptible individuals in region i across classes. $\mathbf{S}_i = (S_{i,c_1},\ldots,S_{i,c_L})$
$\mathbf{I}_i$	distribution of the number of infected individuals in region i across classes. $\mathbf{I}_i = (I_{i,c_1},\ldots,I_{i,c_L})$
$\mathbf{R}_i$	distribution of the number of recovered individuals in region i across classes. $\mathbf{R}_i = (R_{i,c_1},\ldots,R_{i,c_L})$
S_i	number of susceptible individuals in region i, equal to $\lVert\mathbf{S}_i\rVert_1$
I_i	number of infected individuals in region i, equal to $\lVert\mathbf{I}_i\rVert_1$
R_i	number of recovered individuals in region i, equal to $\lVert\mathbf{R}_i\rVert_1$
N_i	population of region i, where $i \in \{1,\ldots,M\}$
λ_i	infection probability for region i. $\lambda_i = \beta\frac{I_i}{N_i}$

7.2.1 Local Epidemic Dynamics

In order to work at the individual level, we adapt the classic deterministic *SIR* model to have a discrete-time stochastic variant. The contact probability β and recovery probability g are constant across all regions.[6] For a region $i \in \{1,\ldots,M\}$ we compute, at each time step, the force of infection $\lambda_i = \beta\frac{I_i}{N_i}$. This quantity represents the probability of making a contact that results in an infection. During a time step, every susceptible individual gets infected independently at random with probability λ_i, and every infective individual recovers independently at random with probability g. If we denote by ΔX_i the variation of $X_i, X_i \in \{S, I, R\}$ after one time step, it is easy to see that

$$\begin{aligned}\mathbb{E}(\Delta S_i) &= -\lambda_i S_i\\ \mathbb{E}(\Delta I_i) &= \lambda_i S_i - gI_i\\ \mathbb{E}(\Delta R_i) &= gI_i\end{aligned}$$

are the expected difference equations for the *SIR* model under the random mixing assumption. We note that our model has many similarities with that of Colizza et al. [45], used to model the SARS pandemic.

---

[6] These quantities are *rates* in the continuous time *SIR* model. In order to carry over the characteristics of the *SIR* model to our discretized version, we need to ensure that the sampling interval is short enough to ensure that $\beta, g < 1$.

7.2.2 Implementation

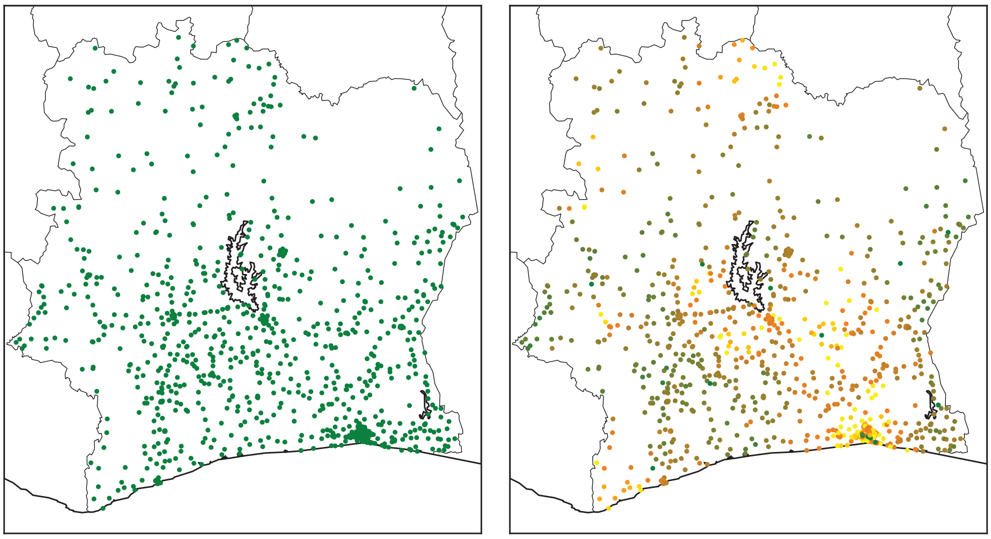


Figure 7.1 – Snapshots of a sample epidemic process where each dot represents a region (here, the surroundings of an antenna). Colors indicate the relative proportion of infective individuals. Initially, just a few individuals form a seed of infectives (left). A little more that 9 days later, the epidemic has spread over most of the country (right).

To allow for distinctive mobility patterns across the population, individuals belong to one out of L classes $\{c_1, \ldots, c_L\}$ that fully characterize their mobility patterns. In accordance with the mobility model (see Section 7.1), the individuals' class is determined by their home antenna. The implementation is best understood when decomposed into two distinct, successive phases: a *mobility* phase where individuals can move between regions, and an *epidemic* phase where individuals get infected or recover.

Mobility phase We consider every individual: Suppose the individual is in region i; the mobility model assigns a new region j according to its mobility class. If $i \neq j$ we update the vectors $\mathbf{X}_i$ and $\mathbf{X}_j$ accordingly, where $\mathbf{X} \in \{\mathbf{S}, \mathbf{I}, \mathbf{R}\}$ depends on the current state of the individual.

Epidemic phase We consider every region $i \in \{1, \ldots, M\}$: We begin this phase by updating the infection rate λ_i, given the current values of N_i and I_i. Every infected individual then recovers with probability g, whereas every susceptible individual gets infected with probability λ_i. $\mathbf{S}_i$, $\mathbf{I}_i$ and $\mathbf{R}_i$ are updated accordingly.

This process is repeated until the end of the period of interest.

7.3 Mobile Micro-measures

Traditional epidemic-mitigation methods consist of heavy, top-down approaches such as blockades, quarantines or large-scale vaccinations. For an alternative, we suggest that mobile technology could enable a much richer and sophisticated set of mitigation measures for human-mediated epidemics. In particular, we introduce the concept of *micro-measures*, individual countermeasures tailored to their recipients' specific behavior; this new approach is the opposite of the one-size-fits-all pattern that characterizes most traditional mitigation measures. The main characteristics of such micro measures are as follows:

Personalized. Recommendations are generated and communicated on an individual basis. Mobile technology enables this in two ways: First, it allows for a quantity of valuable behavioral information (such as location and activity) to be recorded, and second, it provides a readily available unicast communication channel.

Adaptive. As the epidemic progresses and each individuals' intentions are discovered, the recommendations are instantly adapted to the situation. The personalization of mobile micro-recommendations ensures their effectiveness. Such recommendations, in contrast with most large-scale mitigation efforts, would typically require much less time to be set up and would always be in phase with the current state of the epidemic.

Microscopic. In contrast with a one-size-fits-all policy that typically considers an epidemic from a macroscopic perspective, micro-measures tend to focus on subtle and local changes. These changes, when looked at independently, are mostly insignificant; but taken together, they result in important global improvements.

State-independent. An additional property of the service is that it is epidemic-state independent: the recommendation should not depend on whether the individual is infective or not. First, it does not require prior knowledge about the state of an individual as it is often hard to determine precisely when she becomes infected. Second, it aligns the incentives: Without additional knowledge, everyone can expect to benefit from following the recommendation; this might not necessarily be the case when the state is known.

This lays the foundation of our approach but does not yet suggest any concrete mitigation scheme. Still, there are fundamental questions related to the feasibility of micro-measures. Under which conditions do small, local changes (such as an individual's agreeing to commute slightly earlier) have a global impact? How many individuals need to cooperate, and how does this, significantly alter the dynamics of the epidemic? An epidemic can often be seen as being either supercritical (the epidemic grows) or sub-critical (it declines). What microscopic changes are more likely to bring about a phase transition? Although a precise characterization of these changes and, by extension, rigorous answers to these questions are beyond the scope of this chapter, we intend to show initial evidence of the relevance of such a mobile service.

7.3.1 Concrete Micro-Measures

Beyond the theoretical arguments, our contribution is the description and evaluation of three concrete strategies to generate micro-measures. These strategies represent initial baselines for further developments. Let us first note that contacts between individuals can broadly be categorized into two groups: the deliberate contacts are, for example, between family members or at work, whereas accidental contacts are formed by random encounters, for instance, while shopping or commuting. At a high level, our approach is to maintain deliberate contacts and to rewire the accidental ones. The idea is to *weaken* the links in the contact network that form the path through which the epidemic spreads. By changing its structure, we seek to decelerate the dynamics and drive the epidemic down to a sub-critical level.

Table 7.5 – We recapitulate the main characteristics of the three strategies we have implemented to mitigate the spread of epidemic.

	CUTCOMMUNITIES	DECREASEMIX	GOHOME
Knowledge to maintain	List of communities of locations	Social communities of users	State of the epidemic across regions
Recommendation	Do not cross community boundaries	Stay with your social circle	Go/stay home
Intuition	Weakening the weak geographical links	Segmenting social communities	Home is a safe place

CUTCOMMUNITIES **strategy**

It is clear that mobility is a driving factor for the spread of an epidemic. A straightforward strategy would therefore be to reduce long-range contacts; it might be at the expense of reinforcing local contacts. Uniformly reducing mobility is, however, both expensive and inflexible. To overcome this, our first strategy, CUTCOMMUNITIES, takes into account *communities of locations* in the mobility network and focuses on reducing human mobility over inter-community links—this is, in a sense, analogous to *weakening the weak links* in the network. The main difference with a simple blockade is that our strategy is able to adapt to changes in the network (note that mobility patterns vary over time, e.g., see Section 7.1). In practice, the service operator would maintain a list of location communities identified through the mobility patterns of its userbase; when an individual checks whether a trip is safe, the service would verify whether it crosses community boundaries and, if this is the case, discourage the individual from making this trip.[7] If additional per-location information is available about the current state of the epidemic, recommendations could be further corrected according to the strength of the epidemic at the individual's current and projected locations.

---

[7] As a relaxation of this counter-measure, we could consider *postponing* the trip instead. Simply by delaying certain trips, we could prevent harmful interactions between groups of individuals. This is analogous to *time-division multiplexing*; a slight change in the habits of a group of people could significantly change the contact surface.

DECREASEMIX **strategy**

Instead of acting on mobility, to segment contacts across location communities, we also consider the segmentation of *social communities*. We separate individuals *inside* the same location, e.g., by making them visit different aisles of the same supermarket at different times. Putting in place such a segmentation is more sophisticated than in the case of mobility, but this strategy is the perfect example of another extremal point in the solution space. The service operator would keep a list of social communities and would communicate a distinctive tag (e.g., a color) to individuals according to their community. Individuals would access locations differently, depending on their tag; for example, seating in a theater would be organized in such a way that contacts between communities are minimized. We are aware that this strategy could raise many concerns, because it segregates people, therefore great care would be needed if it were to be implemented. Despite this, we retain it because it reflects a different trade-off with respect to CUTCOMMUNITIES: Instead of discouraging individuals from going to certain locations where they can be in contact with everyone, we allow them go everywhere, but restrict the contact network.

GOHOME **strategy**

We consider a third case where the service recommends individuals to *go home*. The intuition behind this strategy is that we assume that when at home, the contact rate decreases. Whereas the previous strategies target the individuals' location or contact network, this one is distinctive in that it affects the rate of contact. With information on the progress of the epidemic across locations, the operator could prioritize sending advice to those individuals whose cooperation would yield the greatest effect. In Section 7.4, we will provide a detailed evaluation of the three described strategies.

7.4 Empirical Evaluation

Next, we use our previously developed mobility and epidemic models to test the strategies described in Section 7.3. Before evaluating our strategies, we first explain how the epidemic model is parameterized and how epidemic spreads are quantitatively characterized.

7.4.1 Model Parameters and Evaluation Metrics

In order to be consistent with our mobility model, the epidemic model defines regions to be the area surrounding the antennas ($M = 1231$). Hence, we will use the words *region* and *antenna* interchangeably. As an individual's mobility is tied to her home antenna, we distinguish among $L = 1231$ different classes. To initialize the population attached to each antenna, we use data from the AfriPop project [185] that provides us with Ivory Coast population figures at the hectare level; to account for the fact that not every individual is mobile, we allow only

55% of the population to move during the mobility phase[8], which roughly corresponds to the proportion of the population in the 15-to-64 age bracket [191]. Days are divided into three time steps in order to match the mobility model[9], and the typical time horizon is between 100 and 400 time steps (i.e., 1–4 months). Contact and recovery probability are usually set to $\beta = 1$ and $g = 0.5$, respectively; although these synthetic values do not directly match any well-known disease, they are still qualitatively close to realistic cases, such as influenza. All our simulations start with a seed set of 23 infectives distributed across 5 antennas[10] in the Attécoubé district of Abidjan.

In order to quantify the difference between epidemic spreads, we propose three metrics for evaluating the effectiveness of our mitigation strategies. Figure 7.2 shows how these quantities are related to the epidemic's evolution over time. For notational clarity, let $X = \sum_{i=1}^{M} X_i, X \in \{S, I, R\}$ be the total number of individuals in each state over the country as a whole. As these quantities evolve over time, they are functions of the time step n. The first metric is the *size of the largest outbreak* or, equivalently, the maximal proportion of infective individuals,

$$I^* = \max_n \frac{I(n)}{N}.$$

The reasoning behind this metric is self-evident: in most cases, the larger the proportion of infective individuals, the more difficult the control of the epidemic. It is also, broadly speaking, a good indicator of the epidemic's strength. Our second metric is closely related to the first one, but considers the complementary dimension: it measures the *time of the largest outbreak*,

$$T^* = \operatorname*{argmax}_n I(n).$$

Delaying the moment at which the epidemic reaches its peak enables individuals and governments to have enough time to adapt their behavior, respectively, to deploy measures. Finally, our last metric captures the tail behavior of the epidemic: it measures the *final proportion of recovered users*,

$$Q^* = \lim_{n \to \infty} \frac{R(n)}{N}.$$

Note that we would like to minimize this metric. After the epidemic dies out, all individuals are either recovered or susceptible, and a low proportion of recovered individuals means that a high percentage of the population did not go through the infective state at all.

---

[8] This distinction is rather crude and could certainly be further refined. However we deemed it to be sufficient for our purposes.

[9] Notice that this is not a formal requirement. We use this subdivision mainly for simplicity.

[10] In the datasets provided by France Telecom-Orange, these antennas have the following identifiers: 57, 146, 330, 836, 926.

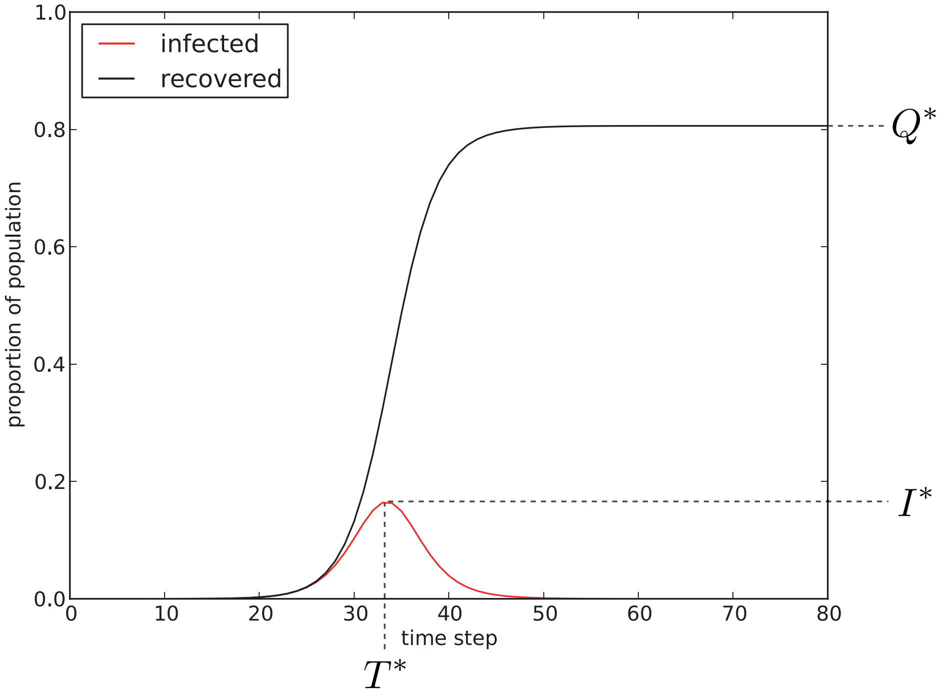


Figure 7.2 – Metrics used to evaluate the effectiveness of mitigation strategies. I^* indicates the magnitude of the epidemic's peak, T^* the time at which the peak happens, and Q^* describes the asymptotic number of individuals that got infected and recovered.

7.4.2 Results

We now take a closer look at our three proposed strategies. We will describe how we instantiate them and we provide qualitative and quantitative assessments with respect to their effectiveness.

CutCommunities **strategy**

The first strategy divides the country into location communities, according to the network of mobility. We consider the weighted, undirected graph where nodes represent antennas, and edge weight is equal to the average number of trips between the two endpoints (regardless of direction). We use the Louvain community detection algorithm [30]; Figure 7.3 shows the 30 identified communities. It is interesting—but not surprising—to note that the communities are roughly geographicaly based.[11] This confirms our hypothesis stating that there are *geographical weak links*. Micro-measures are then generated as follows: When an individual checks whether a trip is safe, the service first verifies whether the trip crosses community

---

[11] As a sidenote, we ran the Louvain method on a number of other graphs generated from the datasets provided for the D4D challenge, including one derived from `SET1` representing total antenna-to-antenna communications. The communities always displayed the same geographical clustering. Furthermore, we observed that mobility communities seem to be correlated to phone call communities.

boundaries and whether the current or projected locations are affected by the disease; if both of these conditions are met, the individual is discouraged from making the trip. The recipient then complies with probability p.

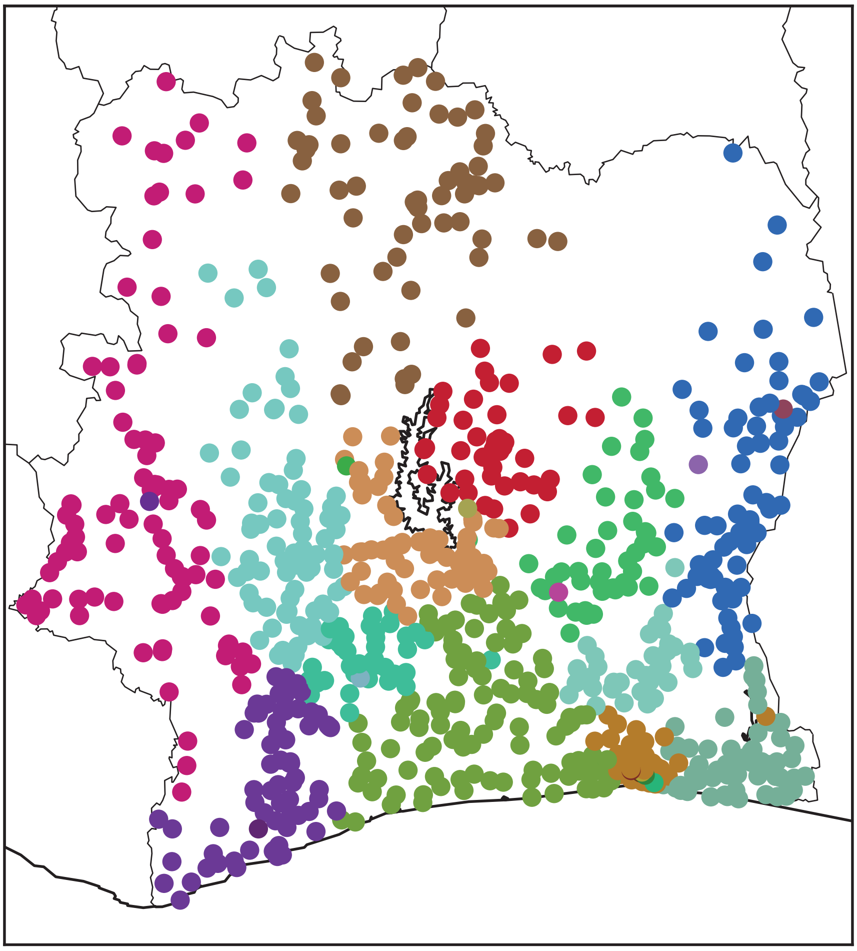


Figure 7.3 – We find 30 communities in the mobility network, when using the Louvain community detection algorithm [30]. It is not surprising that these communities reflect the geographical proximity between nodes, as trips between close antennas are more frequent than between distant ones.

Figure 7.4 shows the effect of CUTCOMMUNITIES for different values of p. Compared to the baseline ($p = 0$), the strategy affects the size I^* and the time T^* of the epidemic's peak. However, it does not change much the tail behavior: Q^* stays constant at around 0.8, except for the degenerate case where $p = 1$, which represents a blockade around the community initially infected. We also observe that there seem to be two infection phases, made progressively more apparent as $p \to 1$, and that the blockade removes the second phase; these two phases correspond to infections happening inside, respectively, outside the initially infected community. Recall that this strategy only sends micro-measures to a fraction of the individuals, those who cross community boundaries—a case that by definition does not happen too often. It is therefore interesting to consider the number of trips actually *canceled* as a result: Table 7.6 lists the average and maximal proportion for different values of p. The numbers are quite low[12], suggesting that the communities form a natural partitioning of the regions. In conclusion, this strategy does not affect the asymptotic behavior of the epidemic but significantly shifts its peak. Altogether, it justifies the relevance of mobility-based geographical communities as a data source to generate micro-measures.

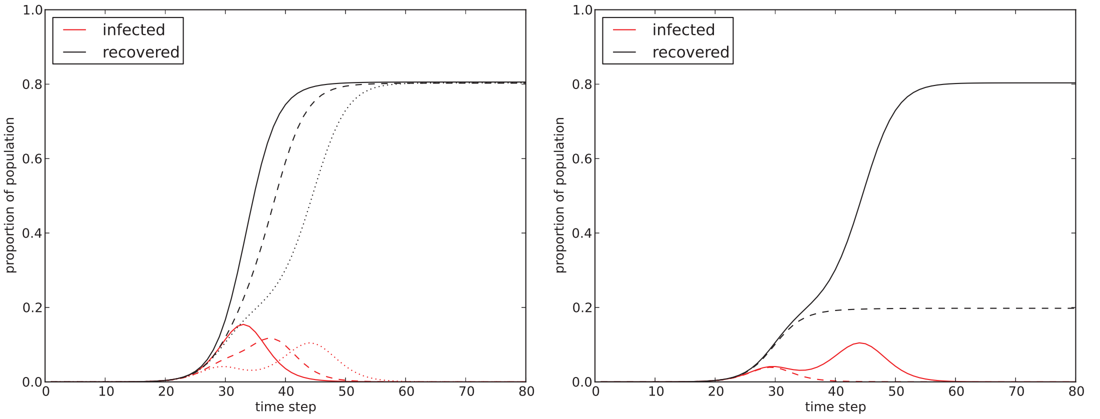


Figure 7.4 – Shape of the epidemic under the CutCommunities strategy, $\beta = 1.0$, $g = 0.5$. On the left: solid lines represent the baseline ($p = 0$), dashed lines $p = 0.9$, dotted lines $p = 0.99$. On the right, we compare $p = 0.99$ (solid) to a complete blockade ($p = 1$, dashed).

Table 7.6 – Proportion of movements affected when using the CutCommunities strategy for three different values of the compliance probability p. We indicate the overall average over the 80 time steps, as well as the maximum value.

p	Affected movements	Maximum
0.90	10.91%	21.38% ($ts = 42$)
0.99	12.57%	22.91% ($ts = 51$)
1.00	5.32%	12.20% ($ts = 33$)

DecreaseMix **strategy**

Recall that this strategy assigns tags to individuals according to the social community to which they belong, and it segregates contacts across social communities. A service operator might use the call graph (i.e., the social network generated by using the information from who calls whom) to infer social communities in the population; unfortunately, we do have access to such data. In order to quantify the effectiveness this strategy, we proceed as follows. Similarly to our mobility model, we make the assumption that the individual's community C is determined by

---

[12] That these proportions are lowest when $p = 1$ is due to the fact that the epidemic is local to the infective seeds' community

his home antenna. The DECREASEMIX strategy does not decrease the total number of contacts; instead it rewires contacts across communities to contacts inside the community. This is done by splitting the contact probability into intra-community and inter-community contact probabilities and introducing a mixing parameter q as follows:

$$\begin{aligned}
\beta_{i,C} &= \left(1 - q + q\frac{N_{i,C}}{N_i}\right)\beta \\
\beta_{i,\overline{C}} &= \beta - \beta_{i,C} \\
\lambda_{i,C} &= \beta_{i,C}\frac{I_{i,C}}{N_{i,C}} + \beta_{i,\overline{C}}\frac{I_{i,\overline{C}}}{N_{i,\overline{C}}},
\end{aligned}$$

where $N_{i,C}$ indicates the number of individuals of community C currently in region i, $N_{i,\overline{C}} = N_i - N_{i,C}$ and the other quantities follow the same convention of notation. The intuition is as follows: When $q = 1$, everyone mixes at random inside a region, just as if no countermeasure were applied at all. At the other extreme, when $q = 0$, contacts happen only with individuals from the same community. Intermediary values of q enable us to play with the strength of the segregation.

We evaluate the effectiveness of DECREASEMIX for different values of the mixing parameter q. Our simulations are parameterized with $\beta = 1.0$, $g = 0.5$ and $q \in \{1, 0.1, 0.01\}$; Figure 7.5 shows the average behavior of the epidemic over 10 runs. The main characteristic of this strategy is that it delays the epidemic outbreak. However, the slopes of the two curves at the strongest point of the epidemic are not that much different. As s result, the final proportion of recovered Q^* does not vary much. But by making it 10 or 100 times more likely to contact an individual of the same community, we delay T^* by approximately 5 and 16 days, respectively. Our intuition about this phenomenon is that it takes more time for the epidemic to reach certain communities (as they are more segregated), but once a community sees its first case of infection, the spread is just as fast as before. We argue that one of the main limiting factors at play here is the random mixing assumption: if we were able to bring finer structural changes to the contact graph, the situation would look very different.

GOHOME **strategy**

Our last strategy encourages individuals to go home or stay home. In order to focus the micro-measures on the most influential individuals, we assume that at each time step, the service operator knows the proportion of susceptible, infective and recovered individuals across locations. We suppose that before every trip, an individual sends a request to the service that compares the proportion of infectives in both source and destination; the service recommends going home if the destination has a proportion of infectives *lower* than the source location. Individuals then comply with probability p. The main intuition behind this choice is to avoid sending infective individuals to highly susceptible locations. Note that we keep the state-independent assumption here: we do not know the state of the individual when sending out a recommendation. The second important assumption is that, when an

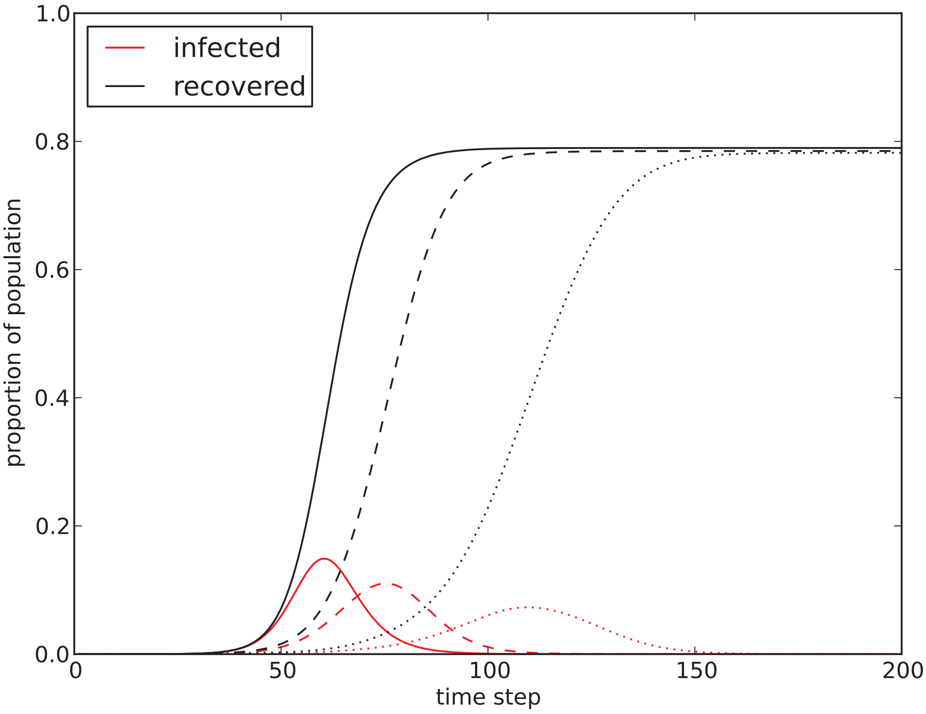


Figure 7.5 – Shape of the epidemic under the DECREASEMIX strategy averaged over 10 runs, $\beta = 1.0$, $g = 0.5$, for different values of the mixing parameter. Solid lines correspond to $q = 1.0$, dashed ones to $q = 0.1$, dotted ones to $q = 0.01$.

individual is at home, the contact probability is set to be equal to the recovery probability[13], i.e., $\beta_{home} := g$. This models the fact that there are less contacts at home, in term of accidental ones. Mixing is not exactly uniform anymore, and the infection probability is adapted as follows:

$$\lambda_{i,loc} = \beta_{home} \frac{I_i}{N_i}$$
$$\lambda_{i,vis} = \beta \frac{I_{vis}}{N_i} + \beta_{home} \frac{I_{loc}}{N_i}.$$

Quantities with *loc* and *vis* subscripts correspond to individuals whose home region is i and is not i, respectively. Note that the contact probability of visitors can significantly decrease in a region where the proportion of visitors to locals is low.

This time, the effectiveness depends on the value of the compliance probability p. We use again $\beta = 1.0$, $g = 0.5$ and let $p \in \{0.0, 0.1, 0.5, 0.7\}$; Figure 7.6 shows the behavior of the epidemic over 10 runs. As opposed to the results obtained with the DECREASEMIX strategy, we obtain

---

[13] When contact and recovery probability are equal, the single-population *SIR* epidemic (under the random mixing assumption) does not develop anymore; setting $\beta_{home} := g$ can therefore be seen as the least change needed to stabilize the epidemic.

significant improvements to Q^* as p increases.[14] This observation is not surprising because by suggesting to individuals to go home, we are directly reducing their contact probability, which is a determining factor of the epidemic's dynamics. It is also interesting to look at the actual number of trips that are affected (i.e., cancelled) because of the micro-measures; Table 7.7 shows that a relatively low number of trips have to be altered to noticeably impact the spread. In summary, this strategy has the potential to be quite effective, although the assumptions it makes deserve a closer analysis.

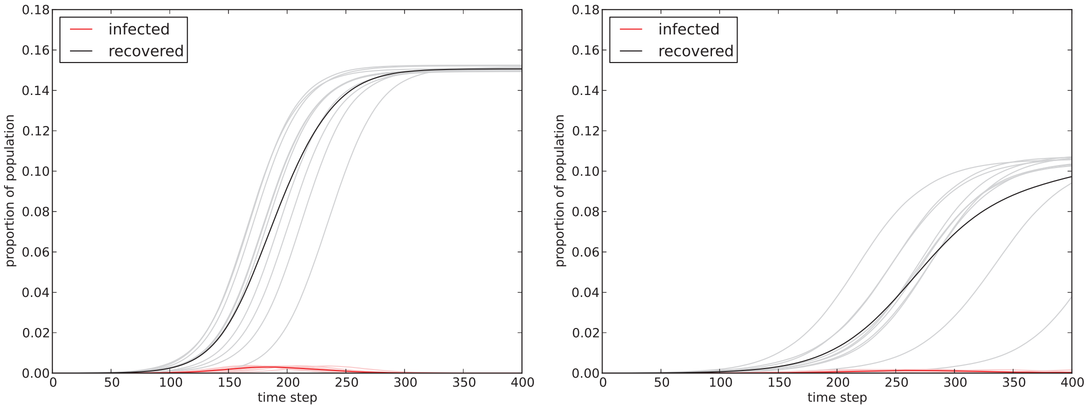


Figure 7.6 – Shape of the epidemic under the GOHOME strategy, $\beta = 1.0$, $g = 0.5$. Light curves indicate individual runs, dark curves indicate average. On the left: $p = 0.1$, on the right: $p = 0.5$.

Table 7.7 – Proportion of movements affected when using the GOHOME strategy for two different values of the compliance probability p. We indicate the overall average over the 400 time steps, as well as the maximum value.

p	Affected movements	Maximum
0.1	2.81%	5.21% ($ts = 190$)
0.5	15.80%	26.12% ($ts = 316$)

7.5 Summary

In this chapter, we have explored the novel idea of using mobile technology in order to mitigate the spread of human-mediated infectious diseases. We explicate the concept of *mobile micro-measures* that consist of personalized behavioral recommendations given to individuals based on a human-contacts network. By affecting, even partially, individual behaviors, we are able to significantly hinder the epidemic propagation. These mobile micro-measures have several original properties: they are adaptive, target individuals at the microscopic level and provide a

---

[14] Unfortunately, our simulation was limited to 400 time steps, which is not enough to clearly show the asymptotical behavior. Our idea for the significant improvements to Q^*, however, is justified by looking at the *worst* runs whose slope quickly tends to zero.

rich set of mitigation methods. Using the data provided for the Orange D4D challenge [31], we first develop a realistic mobility-model for the population of Ivory Coast. Then, we incorporate this mobility model into an epidemic model based on *SIR* in order to simulate the epidemic propagation. Taking advantage of this framework, we propose and evaluate three concrete strategies used to generate micro-measures. Our strategies weaken the epidemic's intensity, successfully delay its peak and, in one case, significantly lower the total number of infected individuals.

These preliminary results enable us to identify several research avenues. First, random mixing is the most limiting assumption. Being able to change, at a finer level, the structure of a human-contacts network is a key component of more advanced micro-measures. The mobile-call graph is an example of a source of information about a social-contacts network, one that is readily available to mobile-phone operators. Second, beyond our preliminary strategies, it is highly important to deepen our understanding of the key ingredients that make mobile micro-measures effective yet minimally restrictive. In parallel to mobile micro-measures, the availability of large-scale mobility data opens up new research directions in epidemiology; a more precise characterization of the relation between epidemic spread and human mobility patterns is an interesting topic to investigate. To conclude this chapter, we firmly believe that data-driven and personalized measures, which take advantage of mobile technology, are an important step towards effective epidemic mitigation.

8 Conclusion and Perspectives

In this thesis, we have succeeded in solving several network-mining problems from three different aspects: the modeling and theory perspective, the computational perspective, and the application perspective. To make better inferences about a network and its properties, we tackled the problem of merging information from different sources; the main focus was on network alignment. More specifically, in the first part of the thesis, we have established fundamental bounds about the feasibility of network alignment between two networks. In the second part of this thesis, we proposed a new network-alignment algorithm, distinguishing computational limitations from theoretical limitations. In the third part of this thesis, by using our results in the first two parts, we have taken an application approach to several problems.

In Chapter 2, we have investigated the network-alignment problem from the modeling and theory perspective. We propose a stochastic model for generating two correlated graphs with partial node-overlap. We find sufficient conditions for the identifiability of the true partial-alignment between the vertex sets of the two graphs. More specifically, we formulate conditions for network density and prove that within these conditions a perfect alignment is feasible. We show that the condition is indeed a mild condition on the scaling of the average degrees of the two networks. Our theoretical results imply that, given unbounded computational resources, network alignment is feasible in the presence of some minimal structural similarity between two networks.

In Chapter 3, we have studied the network-alignment problem from the computational perspective. We propose a new percolation-based network alignment algorithm that, by using only the network structure and a handful of initially pre-matched node couples called seed set, can match large networks. We achieve a dramatic reduction in the size of the seed set. We prove that under a wide range of network parameters, with high probability, our algorithm will percolate, generating a large number of incorrect candidate couples along the way, but will align only correct couples. By using ideas from bootstrap percolation theory, a phase transition in the seed-set size of the percolation graph-matching (PGM) algorithm is formally established. We also show the excellent performance of our algorithm in matching several large real social-networks.

From the application perspective, we have considered two important application areas of network mining in biology and public health. The first application area is protein-protein interaction (PPI) network alignment in biology. The alignment of PPI networks enables us to uncover the relationships between different species, which leads us to a deeper understanding of biological systems. Network alignment can be used to transfer biological knowledge between species. Although different PPI-alignment algorithms were introduced during the last decade, developing an accurate and scalable algorithm that can find alignments with high biological and structural similarities among PPI networks is still challenging.

In Chapter 4, we have introduced a new global pairwise-network alignment algorithm for PPI networks; we call the algorithm PROPER. Compared to other global network-alignment methods, our algorithm shows higher accuracy and speed over real PPI datasets and synthetic networks. We show that the PROPER algorithm can detect large portions of conserved biological pathways between species. We highlight that PROPER has high potential in further applications, such as detecting biological pathways, finding protein complexes and PPI prediction.

In Chapter 5, we have extended PROPER to the global multiple-network alignment problem. We have introduced MPROPER, a new scalable and accurate algorithm for aligning multiple networks. We show that MPROPER outperforms the other state-of-the-art algorithms. To generate k correlated networks, we present a graph-sampling model, as a generalization of the model introduced in Chapter 2. By using this model, we guarantee the performance of the MPROPER algorithm.

In Chapter 6, we have explored IsoRank, one of the first and most referenced global pairwise-network alignment algorithms. We show that, when IsoRank similarity depends only on the network structure, the final alignment is only a function of node degrees. Also, we develop an approximation algorithm that outperforms IsoRank in time and memory complexity, by several orders of magnitude, despite only a negligible loss in precision.

Our second application area is the control of epidemic processes. In Chapter 7, we have developed and modeled strategies for mitigating an epidemic in a large-scale dynamic contact network. We model the spread of epidemics on a network by using many pieces of information about the mobility and behavior of a population, such as mobile call-data records. We first develop a realistic mobility model for the population. Then, we incorporate this mobility model into an epidemic model in order to simulate the epidemic propagation. Taking advantage of this framework, we propose three concrete strategies used to mitigate the effect of epidemics on that network. The goal of each strategy is a large reduction in infections, with a small effect on the normal course of daily life. Finally, we evaluate these strategies over the Orange D4D dataset and show the benefit of them, even if only a fraction of the population participates.

8.1 Open Questions

There are still several open questions that await further exploration.

- In Chapter 3, we have done a comprehensive evaluation of PGM algorithms. Our experimental results confirm that with only a few number of seeds it is possible to align many real networks and random graphs. Despite the excellent performance of PGM algorithms, their success depends on the structural similarity of two networks. It would be beneficial to investigate the necessary and sufficient conditions under which PGM algorithms are able to align two networks successfully.
- We believe it is possible to push further the class of PGM algorithms and design seed-less graph-matching algorithms. For this reason, we can rely on two important ideas: (i) Structural information of real networks (e.g., node degrees in networks with heavy-tailed degree distribution) could be used for finding initial (noisy) seed couples; and (ii) it is possible to make PGM algorithms even more robust to the noise. For example, we have hard thresholding for matching. Indeed, in our PGM algorithms, when a node couple receives enough marks, it is permanently matched, and an incorrectly matched couple from the early steps cannot be corrected later. By relaxing this hard thresholding, a PGM algorithm can tolerate a higher level of noise.
- We have demonstrated that aligning k different network, when initially enough number of seed-tuples is provided, is possible. We believe that, in the multiple-network alignment problem, there is more information than in the case of pairwise alignment. For example, to better align two networks, we can use the similarities of these networks with a third network. An interesting research objective is further study of multiple-network alignment from the modeling and theory perspective, and the computational perspective.
- Although there are significant public benefits to release network data, these networks often contain sensitive information about the node identities and interactions among them. From the privacy point of view, revealing the inclusion or removal of a node or an edge in a network is important and has many implications. A fundamental research question is how to perturb the structure of a network in oder to make it resilient against PGM de-anonymization attacks, while preserving global (or local) properties of the network.
- A promising research direction could be improving the performance of our PPI network-alignment algorithms and exploring their applications:
  - Designing a variant of PROPER that takes into account gene duplication, network motifs, clustering within networks and modularity of biological networks.
  - Improving our multiple-network alignment algorithm by incorporating information about the level of similarities of organisms, e.g., by using data from phylogenetic tree.

  - Designing algorithms for detecting protein complexes and biological pathways by using network alignment.

8.2 Future Research Directions

Traditionally, reasoning about a network (or network inference) is based on the assumption that the whole network is observable and that there is no ambiguity in the states and labels of nodes. Unfortunately, it is not often possible to have a complete and unambiguous view of a network, and in many scenarios there is insufficient information. For example, when the underlying network is hidden, we might have access only to a set of active and passive measurements of a network, such as a partial observation of a diffusion process over a network or a temporal state of nodes in a dynamic network. Sometimes real networks are observed through their subgraphs (patches), where node labels are ambiguous. In this regard, network alignment, when relying solely on the structure of networks, is the most important function for making inferences, when we are given two large (global) patches with no further (or a very restricted) side information.

Designing efficient algorithms, to overcome the limitations caused by an incomplete view of a network, will be a major benefit to the community of network science researchers and will have applications in many domains, such as public health, biology and technology. Hence, one research goal, for making effective inferences about global (or local) structure and properties of networks, could be to provide a unified framework for combining information from different types of network patches. We believe that the following directions provide a promising avenue for future research.

- Developing a fundamental understanding of the properties of networks under incomplete, noisy and partial observations.
- Developing a unified graph-sampling framework for characterizing a wide range of partial observations of networks.
- Designing algorithms for network reconstruction and for inferring global (or local) properties of networks from network samples (e.g., patches) and noisy measurements.
- Using the developed models and algorithms to make inferences about real networks, mainly for biological networks (e.g., to find biological pathways) and for public health studies (e.g, to study hidden population-networks).

Bibliography

[1] IntAct: an open source molecular interaction database. http://www.ebi.ac.uk/intact/. Accessed 04 April 2016.

[2] IsoBase: A Database of Functionally Related Orthologs. http://groups.csail.mit.edu/cb/mna/isobase/. Accessed 04 April 2016.

[3] KEGG pathway database - Kyoto University Bioinformatics Centre. http://www.genome.jp/kegg/pathway.html. Accessed 04 April 2016.

[4] UniProt: the universal protein knowledgebase. http://www.uniprot.org/. Accessed 04 April 2016.

[5] The GOA database. http://www.ebi.ac.uk/GOA/downloads/. Accessed 04 April 2016.

[6] PINALOG web server for protein interaction network alignment. http://www.sbg.bio.ic.ac.uk/~pinalog/. Accessed 04 April 2016.

[7] F. Abel, N. Henze, E. Herder, and D. Krause. Interweaving Public User Profiles on the Web. In *UMAP, HI, USA*, pages 16–27, September 2010.

[8] E. Adar and L. A. Adamic. Tracking information epidemics in blogspace. In *Proceedings of the 2005 IEEE/WIC/ACM international conference on web intelligence*, pages 207–214. IEEE Computer Society, 2005.

[9] R. Aebersold and M. Mann. Mass spectrometry-based proteomics. *Nature*, 422(6928): 198–207, 2003.

[10] A. E. Aladag and C. Erten. SPINAL: scalable protein interaction network alignment. *Bioinformatics*, 29(7):917–924, 2013.

[11] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[12] F. Alkan and C. Erten. Beams: backbone extraction and merge strategy for the global many-to-many alignment of multiple ppi networks. *Bioinformatics*, 30(4):531–539, 2014.

[13] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[14] R. Andersen, F. Chung, and K. Lang. Local Graph Partitioning using PageRank Vectors. In *Proc. of 47th Annual IEEE Foundations of Computer Science (FOCS) 2006*, Berkeley, CA, USA, October 2006.

[15] R. Apweiler, A. Bairoch, C. H. Wu, B. Barker, Winona Cand Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, et al. UniProt: the universal protein knowledgebase. *Nucleic Acids Research*, 32(suppl 1):D115–D119, 2004.

[16] J. Arino and P. Van den Driessche. A multi-city epidemic model. *Mathematical Population Studies*, 10(3):175–193, 2003.

[17] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, et al. Gene Ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

[18] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x?: Anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 181–190, New York, NY, USA, 2007. ACM.

[19] A.-L. Barabási. The origin of bursts and heavy tails in human dynamics. *Nature*, 435 (7039):207–211, 2005.

[20] A.-L. Barabási. *Network science*. Cambridge University Press, 2016.

[21] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *science*, 286 (5439):509–512, 1999.

[22] D. Barrell, E. Dimmer, R. P. Huntley, D. Binns, C. O'Donovan, and R. Apweiler. The goa database in 2009—an integrated gene ontology annotation resource. *Nucleic acids research*, 37(suppl 1):D396–D403, 2009.

[23] J. Batson, D. A. Spielman, and N. Srivastava. Twice-ramanujan sparsifiers. *SIAM Journal on Computing*, 41(6):1704–1721, 2012.

[24] M. Bayati, D. F. Gleich, A. Saberi, and Y. Wang. Message-passing algorithms for sparse network alignment. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 7(1): 3, 2013.

[25] M. A. Bayir, M. Demirbas, and N. Eagle. Discovering SpatioTemporal Mobility Profiles of Cellphone Users. In *WoWMoM 2009*, pages 1–9. IEEE, 2009.

[26] R. Becker, R. Cáceres, K. Hanson, S. Isaacman, J. M. Loh, M. Martonosi, J. Rowland, S. Urbanek, A. Varshavsky, and C. Volinsky. Human Mobility Characterization from Cellular Network Data. *Communications of the ACM*, 56(1):74–82, 2013.

[27] V. Belik, T. Geisel, and D. Brockmann. The impact of human mobility on spatial disease dynamics. In *CSE 2009*, volume 4, pages 932–935. IEEE, 2009.

[28] J. Berg, M. Lässig, and A. Wagner. Structure and evolution of protein interaction networks: a statistical model for link dynamics and gene duplications. *BMC evolutionary biology*, 4(1):51, 2004.

[29] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[30] V. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10): P10008, 2008.

[31] V. D. Blondel, M. Esch, C. Chan, F. Clerot, P. Deville, E. Huens, F. Morlot, Z. Smoreda, and C. Ziemlicki. Data for Development: the D4D Challenge on Mobile Phone Data. *arXiv preprint arXiv:1210.0137*, 2012.

[32] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang. Complex networks: Structure and dynamics. *Physics reports*, 424(4):175–308, 2006.

[33] B. Bollobás. *Random Graphs*. Cambridge University Press, 2001.

[34] B. Bollobás and O. Riordan. The Diameter of a Scale-Free Random Graph. *Combinatorica*, 24(1):5–34, 2004.

[35] M. Carcassoni and E. R. Hancock. Alignment using spectral clusters. In *BMVC*, pages 1–10, 2002.

[36] A. Chatr-aryamontri, B.-J. Breitkreutz, S. Heinicke, L. Boucher, A. Winter, C. Stark, J. Nixon, L. Ramage, N. Kolas, L. O'Donnell, et al. The BioGRID interaction database: 2013 update. *Nucleic acids research*, 41(D1):D816–D823, 2013.

[37] L. Chen, F. W. Crawford, and A. Karbasi. Seeing the unseen network: Inferring hidden social ties from respondent-driven sampling. In *Proc. of 30th AAAI Conference on Artificial Intelligence (AAAI 2016)*, Phoenix, AZ, USA, February 2016.

[38] S. Chen, J. Fan, G. Li, J. Feng, K.-l. Tan, and J. Tang. Online topic-aware influence maximization. *Proceedings of the VLDB Endowment*, 8(6):666–677, 2015.

[39] J. Cheng, L. Adamic, P. A. Dow, J. M. Kleinberg, and J. Leskovec. Can cascades be predicted? In *Proceedings of the 23rd international conference on World wide web*, pages 925–936. ACM, 2014.

[40] C. F. Chiasserini, M. Garetto, and E. Leonardi. De-anonymizing scale-free social networks by percolation graph matching. In *Proc. of IEEE INFOCOM 2015*, Hong Kong, April 2015.

[41] C. F. Chiasserini, M. Garetto, and E. Leonardi. Impact of Clustering on the Performance of Network De-anonymization. In *Proc. of ACM COSN 2015*, Palo Alto, CA, USA, November 2015.

[42] F. Chung and L. Lu. Connected components in random graphs with given expected degree sequences. *Annals of combinatorics*, 6(2):125–145, 2002.

[43] C. Clark and J. Kalita. A comparison of algorithms for the pairwise alignment of biological networks. *Bioinformatics*, 30(16):2351–2359, 2014.

[44] V. Colizza, A. Barrat, M. Barthélemy, and A. Vespignani. The role of the airline transportation network in the prediction and predictability of global epidemics. *PNAS*, 103 (7):2015–2020, 2006.

[45] V. Colizza, A. Barrat, M. Barthélemy, and A. Vespignani. Predictability and epidemic pathways in global outbreaks of infectious diseases: the SARS case study. *BMC Medicine*, 5(34), 2007.

[46] S. R. Collins, P. Kemmeren, X.-C. Zhao, J. F. Greenblatt, F. Spencer, F. C. Holstege, J. S. Weissman, and N. J. Krogan. Toward a comprehensive atlas of the physical interactome of saccharomyces cerevisiae. *Molecular & Cellular Proteomics*, 6(3):439–450, 2007.

[47] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence*, 18 (03):265–298, 2004.

[48] F. W. Crawford. Hidden network reconstruction from information diffusion. In *Proc. of 18th International Conference on Information Fusion (Fusion 2015)*, Washington, DC, USA, July 2015.

[49] D. Cullina and N. Kiyavash. Improved Achievability and Converse Bounds for Erdős–Rényi Graph Matching. In *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*, New York, NY, USA, 2016. ACM.

[50] D. Cullina, K. Singhal, N. Kiyavash, and P. Mittal. On the simultaneous preservation of privacy and community structure in anonymized networks. *arXiv preprint arXiv:1603.08028*, 2016.

[51] D. Davis, Ö. N. Yaveroğlu, N. Malod-Dognin, A. Stojmirovic, and N. Pržulj. Topology-function conservation in protein–protein interaction networks. *Bioinformatics*, page btv026, 2015.

[52] D. Devos and A. Valencia. Practical limits of function prediction. *Proteins: Structure, Function, and Bioinformatics*, 41(1):98–107, 2000.

[53] J. Diamond. *Guns, germs, and steel: The fates of human societies*. WW Norton & Company, 1999.

[54] D. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, New York, NY, USA, 1st edition, 2009. ISBN 0521884276, 9780521884273.

[55] D. Easley and J. Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010.

[56] A. Egozi, Y. Keller, and H. Guterman. A Probabilistic Approach to Spectral Graph Matching. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1):18–27, 2013.

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

[58] P. Erdős and A. Rényi. On Random Graphs I. *Publ. Math. Debrecen*, 6:290–297, 1959.

[59] F. E. Faisal, L. Meng, J. Crawford, and T. Milenković. The post-genomic era of biological network alignment. *EURASIP Journal on Bioinformatics and Systems Biology*, 2015(1): 1–19, 2015.

[60] N. M. Ferguson, D. A. Cummings, C. Fraser, J. C. Cajka, P. C. Cooley, and D. S. Burke. Strategies for mitigating an influenza pandemic. *Nature*, 442(7101):448–452, 2006.

[61] M.-L. Fernández and G. Valiente. A graph distance metric combining maximum common subgraph and minimum common supergraph. *Pattern Recognition Letters*, 22(6): 753–758, 2001.

[62] J. Flannick, A. Novak, B. S. Srinivasan, H. H. McAdams, and S. Batzoglou. Graemlin: general and robust alignment of multiple large interaction networks. *Genome research*, 16(9):1169–1181, 2006.

[63] P. Foggia, G. Percannella, and M. Vento. Graph matching and learning in pattern recognition in the last 10 years. *International Journal of Pattern Recognition and Artificial Intelligence*, 28(01), 2014.

[64] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979. ISBN 0716710447.

[65] T. C. Germann, K. Kadau, I. M. Longini Jr, and C. A. Macken. Mitigation strategies for pandemic influenza in the United States. *PNAS*, 103(15):5935–5940, 2006.

[66] V. Gligorijević, N. Malod-Dognin, and N. Pržulj. Fuse: multiple network alignment via data fusion. *Bioinformatics*, page btv731, 2015.

[67] K.-I. Goh, M. E. Cusick, D. Valle, B. Childs, M. Vidal, and A.-L. Barabási. The human disease network. *Proceedings of the National Academy of Sciences*, 104(21):8685–8690, 2007.

[68] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin. Powergraph: Distributed graph-parallel computation on natural graphs. In *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, pages 17–30, 2012.

[69] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabási. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, 2008.

[70] R. Gottfried. *Black Death*. Simon and Schuster, 1985.

[71] F. C. Graham and W. Zhao. A Sharp PageRank Algorithm with Applications to Edge Ranking and Graph Sparsification. In *Proc. of 7th International Workshop on Algorithms and Models for the Web-Graph (WAW) 2010*, Stanford, CA, USA, December 2010.

[72] GraphM. Graph Matching package. http://cbio.ensmp.fr/graphm/.

[73] J. Gratten, N. R. Wray, M. C. Keller, and P. M. Visscher. Large-scale genomics unveils the genetic architecture of psychiatric disorders. *Nature neuroscience*, 17(6):782–790, 2014.

[74] R. Hammack, W. Imrich, and S. Klavžar. *Handbook of product graphs*. CRC press, 2011.

[75] K. Henderson, B. Gallagher, L. Li, L. Akoglu, T. Eliassi-Rad, H. Tong, and C. Faloutsos. It's Who You Know: Graph Mining Using Recursive Structural Features. In *SIGKDD, San Diego, CA, USA*, pages 663–671, August 2011.

[76] H. Hermjakob, L. Montecchi-Palazzi, C. Lewington, S. Mudali, S. Kerrien, S. Orchard, M. Vingron, B. Roechert, P. Roepstorff, A. Valencia, et al. IntAct: an open source molecular interaction database. *Nucleic Acids Research*, 32(suppl 1):D452–D455, 2004.

[77] H. W. Hethcote. The Mathematics of Infectious Diseases. *SIAM review*, 42(4):599–653, 2000.

[78] P. W. Holland, K. B. Laskey, and S. Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

[79] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge university press, 2012.

[80] J. Hu, B. Kehr, and K. Reinert. Netcoffee: a fast and accurate global alignment approach to identify functionally conserved proteins in multiple networks. *Bioinformatics*, page btt715, 2013.

[81] L. Hufnagel, D. Brockmann, and T. Geisel. Forecast and control of epidemics in a globalized world. *Proceedings of the National Academy of Sciences of the United States of America*, 101(42):15124–15129, 2004.

[82] R. Ibragimov, M. Malek, J. Baumbach, and J. Guo. Multiple graph edit distance: simultaneous topological alignment of multiple protein-protein interaction networks with an evolutionary algorithm. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 277–284. ACM, 2014.

[83] T. Inglesby, J. Nuzzo, T. O'Toole, and D. Henderson. Disease Mitigation Measures in the Control of Pandemic Influenza. *Biosecurity and Bioterrorism: Biodefense Strategy, Practice, and Science*, 4(4):366–375, 2006.

[84] S. Isaacman, R. Becker, R. Cáceres, S. Kobourov, M. Martonosi, J. Rowland, and A. Varshavsky. Ranges of Human Mobility in Los Angeles and New York. In *PERCOM 2011*, pages 88–93. IEEE, 2011.

[85] T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, and Y. Sakaki. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proceedings of the National Academy of Sciences*, 98(8):4569–4574, 2001.

[86] S. Janson, T. Luczak, T. Turova, and T. Vallier. Bootstrap Percolation On The Random Graph $G_{n,p}$. *The Annals of Applied Probability*, 22(5):1989–2047, 2012.

[87] H. Jeong and B.-J. Yoon. Accurate multiple network alignment through context-sensitive random walk. *BMC systems biology*, 9(Suppl 1):S7, 2015.

[88] S. Ji, W. Li, M. Srivatsa, and R. Beyah. Structural data de-anonymization: Quantification, practice, and implications. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 1040–1053. ACM, 2014.

[89] S. Ji, W. Li, N. Z. Gong, P. Mittal, and R. Beyah. On your social network de-anonymizablity: Quantification and large scale evaluation with seed knowledge. In *Proc. of the Network and Distributed System Security (NDSS) Symposium*, San Diego, CA, USA, February 2015.

[90] S. Ji, W. Li, P. Mittal, X. Hu, and R. Beyah. Secgraph: A uniform and open-source evaluation system for graph data anonymization and de-anonymization. In *Proc. of USENIX Security Symposium 2015*, Washington, D.C., USA, August 2015.

[91] S. Ji, W. Li, N. Z. Gong, P. Mittal, and R. Beyah. Seed-based de-anonymizability quantification of social networks. *IEEE Transactions on Information Forensics and Security*, 11 (7):1398–1411, 2016.

[92] T. Joshi and D. Xu. Quantitative assessment of relationship between sequence similarity and function similarity. *BMC genomics*, 8(1):222, 2007.

[93] M. Kafsi, E. Kazemi, L. Maystre, L. Yartseva, M. Grossglauser, and P. Thiran. Mitigating Epidemics through Mobile Micro-measures. *CoRR*, abs/1307.2084, 2013.

[94] M. Kalaev, M. Smoot, T. Ideker, and R. Sharan. Networkblast: comparative analysis of protein networks. *Bioinformatics*, 24(4):594–596, 2008.

[95] M. Kanehisa and S. Goto. KEGG: kyoto encyclopedia of genes and genomes. *Nucleic acids research*, 28(1):27–30, 2000.

[96] A. Karbasi, A. H. Salavati, and M. Vetterli. Learning network structures from firing patterns. In *International Conference on Acoustics, Speech and Signal Processing*, 2015.

[97] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 565–574. IEEE, 2000.

[98] E. Kazemi and M. Grossglauser. On the structure and efficient computation of isorank node similarities. *arXiv preprint arXiv:1602.00668*, 2016.

[99] E. Kazemi, S. H. Hassani, and M. Grossglauser. Growing a graph matching from a handful of seeds. *Proc. of the VLDB Endowment*, 8(10):1010–1021, 2015.

[100] E. Kazemi, L. Yartseva, and M. Grossglauser. When Can Two Unlabeled Networks Be Aligned Under Partial Overlap? In *Proc. of IEEE Communication, Control, and Computing (53st Annual Allerton Conference) 2015*, Monticello, IL, USA, October 2015.

[101] E. Kazemi, S. H. Hassani, M. Grossglauser, and H. Pezeshgi Modarres. PROPER: Global Protein Interaction Network Alignment through Percolation Matching. *BMC bioinformatics*, 2016.

[102] M. J. Keeling, L. Danon, M. C. Vernon, and T. A. House. Individual identity and movement networks for disease metapopulations. *PNAS*, 107(19):8866–8870, 2010.

[103] B. P. Kelley, R. Sharan, R. M. Karp, T. Sittler, D. E. Root, B. R. Stockwell, and T. Ideker. Conserved pathways within bacteria and yeast as revealed by global protein network alignment. *Proc. of the National Academy of Sciences*, 100(20):11394–11399, 2003.

[104] B. P. Kelley, B. Yuan, F. Lewitter, R. Sharan, B. R. Stockwell, and T. Ideker. PathBLAST: a tool for alignment of protein interaction networks. *Nucleic acids research*, 32(suppl 2): W83–W88, 2004.

[105] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.

[106] W. Kermack and A. McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society A*, 115:700–721, 1927.

[107] G. Klau. A new graph-based method for pairwise global network alignment. *BMC Bioinformatics*, 10(Suppl 1):S59, 2009.

[108] D. Knossow, A. Sharma, D. Mateus, and R. Horaud. Inexact matching of large and sparse graphs using laplacian eigenvectors. In *Graph-Based Representations in Pattern Recognition*, pages 144–153. Springer, 2009.

[109] N. Korula and S. Lattanzi. An efficient reconciliation algorithm for social networks. *Proc. of the VLDB Endowment*, 7(5):377–388, 2014.

[110] M. Koyutürk, Y. Kim, U. Topkara, S. Subramaniam, W. Szpankowski, and A. Grama. Pairwise alignment of protein interaction networks. *Journal of Computational Biology*, 13(2):182–199, 2006.

[111] O. Kuchaiev and N. Pržulj. Integrative network alignment reveals large regions of global network similarity in yeast and human. *Bioinformatics*, 27(10):1390–1396, 2011.

[112] O. Kuchaiev, T. Milenković, V. Memišević, W. Hayes, and N. Pržulj. Topological network alignment uncovers biological function and phylogeny. *Journal of the Royal Society Interface*, pages 1341–1354, 2010.

[113] A. N. Langville and C. D. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, 2011.

[114] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1482–1489. IEEE, 2005.

[115] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636. ACM, 2006.

[116] J. Leskovec and A. Krevl. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data/, June 2014.

[117] J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, 1(1):5, 2007.

[118] Z. Liang, M. Xu, M. Teng, and L. Niu. Netalign: a web-based tool for comparison of protein interaction networks. *Bioinformatics*, 22(17):2175–2177, 2006.

[119] C.-S. Liao, K. Lu, M. Baym, R. Singh, and B. Berger. IsoRankN: spectral methods for global alignment of multiple protein networks. *Bioinformatics*, 25(12):i253–i258, 2009.

[120] L. Licata, L. Briganti, D. Peluso, L. Perfetto, M. Iannuccelli, E. Galeota, F. Sacco, A. Palma, A. P. Nardozza, E. Santonico, et al. MINT, the molecular interaction database: 2012 update. *Nucleic acids research*, 40(D1):D857–D861, 2012.

[121] G. N. Lin, R. Corominas, I. Lemmens, X. Yang, J. Tavernier, D. E. Hill, M. Vidal, J. Sebat, and L. M. Iakoucheva. Spatiotemporal 16p11. 2 protein network implicates cortical late mid-fetal brain development and kctd13-cul3-rhoa pathway in psychiatric diseases. *Neuron*, 85(4):742–754, 2015.

[122] S. W. Linderman and R. P. Adams. Discovering latent network structure in point process data. *arXiv preprint arXiv:1402.0914*, 2014.

[123] E. M. Loiola, N. M. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido. A survey for the quadratic assignment problem. *European journal of operational research*, 176(2):657–690, 2007.

[124] P. W. Lord, R. D. Stevens, A. Brass, and C. A. Goble. Investigating semantic similarity measures across the gene ontology: the relationship between sequence and annotation. *Bioinformatics*, 19(10):1275–1283, 2003.

[125] L. Lovász. Random walks on graphs: A survey. *Combinatorics, Paul Erdős is eighty*, 2(1): 1–46, 1993.

[126] M. Madan Babu, S. A. Teichmann, and L. Aravind. Evolutionary dynamics of prokaryotic transcriptional regulatory networks. *Journal of molecular biology*, 358(2):614–633, 2006.

[127] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 135–146. ACM, 2010.

[128] A. Malhotra, L. C. Totti, W. M. Jr., P. Kumaraguru, and V. Almeida. Studying User Footprints in Different Online Social Networks. In *ASONAM, Istanbul, Turkey*, pages 1065–1070, August 2012.

[129] N. Malod-Dognin and N. Pržulj. L-GRAAL: Lagrangian graphlet-based network aligner. *Bioinformatics*, 31(13):2182–2189, 2015.

[130] S. Maslov and K. Sneppen. Specificity and stability in topology of protein networks. *Science*, 296(5569):910–913, 2002.

[131] L. R. Matthews, P. Vaglio, J. Reboul, H. Ge, B. P. Davis, J. Garrels, S. Vincent, and M. Vidal. Identification of potential interaction networks using sequence-based searches for conserved protein-protein interactions or “interologs”. *Genome research*, 11(12):2120–2126, 2001.

[132] F. McSherry, M. Isard, and D. G. Murray. Scalability! but at what cost? In *15th Workshop on Hot Topics in Operating Systems (HotOS XV)*, 2015.

[133] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. In *ICDE, San Jose, CA, USA*, pages 117–128, February 2002.

[134] V. Memišević and N. Pržulj. C-graal: C ommon-neighbors-based global gra ph al ignment of biological networks. *Integrative Biology*, 4(7):734–743, 2012.

[135] L. Meng, A. Striegel, and T. Milenković. Local versus global biological network alignment. *Bioinformatics*, 2016.

[136] X. Meng and L. Chen. The dynamics of a new SIR epidemic model concerning pulse vaccination strategy. *Applied Mathematics and Computation*, 197(2):582–597, 2008.

[137] T. Milenković, W. L. Ng, W. Hayes, and N. Pržulj. Optimal network alignment with graphlet degree vectors. *Cancer informatics*, 9:121, 2010.

[138] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel. You are who you know: inferring user profiles in online social networks. In *Proc. of ACM WSDM 2010*, New York City, USA, February 2010.

[139] M. Mistry and P. Pavlidis. Gene ontology term overlap as a measure of gene functional similarity. *BMC bioinformatics*, 9(1):327, 2008.

[140] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *Proc. of IEEE Symposium on Security and Privacy 2009*, Oakland, CA, USA, May 2009.

[141] S. Navlakha and C. Kingsford. Network archaeology: uncovering ancient networks from present-day interactions. *PLoS Comput Biol*, 7(4):e1001119, 2011.

[142] M. Newman. *Networks: an introduction*. Oxford university press, 2010.

[143] M. E. Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409, 2001.

[144] C. Notredame, D. G. Higgins, and J. Heringa. T-coffee: A novel method for fast and accurate multiple sequence alignment. *Journal of molecular biology*, 302(1):205–217, 2000.

[145] A. Nunes, P. Calado, and B. Martins. Resolving user identities over social networks through supervised learning and rich similarity features. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing 2012*, Trento, Italy, March 2012.

[146] S. Orchard, S. Kerrien, S. Abbani, B. Aranda, J. Bhate, S. Bidwell, A. Bridge, L. Briganti, F. S. Brinkman, G. Cesareni, et al. Protein interaction data curation: the International Molecular Exchange (IMEx) consortium. *Nature methods*, 9(4):345–350, 2012.

[147] N. N. Parikshak, M. J. Gandal, and D. H. Geschwind. Systems biology and gene networks in neurodevelopmental and neurodegenerative disorders. *Nature Reviews Genetics*, 16 (8):441–458, 2015.

[148] D. Park, R. S. 0001, M. Baym, C.-S. Liao, and B. Berger. IsoBase: a database of functionally related proteins across PPI networks. *Nucleic Acids Research*, 39(Database-Issue):295–300, 2011.

[149] R. Patro and C. Kingsford. Global network alignment using multiscale spectral signatures. *Bioinformatics*, 28(23):3105–3114, 2012.

[150] P. Pedarsani and M. Grossglauser. On the privacy of anonymized networks. In *Proc. of ACM SIGKDD 2011*, San Diego, CA, USA, August 2011.

[151] P. Pedarsani, D. R. Figueiredo, and M. Grossglauser. A Bayesian Method for Matching Two Similar Graphs without Seeds. In *Conference on Communication, Control, and Computing, Allerton Park, Monticello, IL, USA*, pages 1598–1607, October 2013.

[152] S. Peri, J. D. Navarro, T. Z. Kristiansen, R. Amanchy, V. Surendranath, B. Muthusamy, T. Gandhi, K. Chandrika, N. Deshpande, S. Suresh, et al. Human protein reference database as a discovery resource for proteomics. *Nucleic acids research*, 32(suppl 1): D497–D501, 2004.

[153] C. Pesquita, D. Faria, H. Bastos, A. E. Ferreira, A. O. Falcão, and F. M. Couto. Metrics for GO based protein semantic similarity: a systematic evaluation. *BMC bioinformatics*, 9 (Suppl 5):S4, 2008.

[154] H. T. T. Phan and M. J. E. Sternberg. PINALOG: a novel approach to align protein interactionnetworks—implications for complex detection and function prediction. *Bioinformatics*, 28(9):1239–1245, 2012.

[155] D. M. Powers. Evaluation: from Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.

[156] A. Radu and M. Charleston. Node handprinting: a scalable and accurate algorithm for aligning multiple biological networks. *Journal of Computational Biology*, 22(7):687–697, 2015.

[157] M. Remm, C. E. Storm, and E. L. Sonnhammer. Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. *Journal of molecular biology*, 314(5): 1041–1052, 2001.

[158] P. Resnik et al. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *J. Artif. Intell. Res.(JAIR)*, 11:95–130, 1999.

[159] A. Rinaldo, E. Bertuzzo, L. Mari, L. Righetto, M. Blokesch, M. Gatto, R. Casagrandi, M. Murray, S. M. Vesenbeckh, and I. Rodriguez-Iturbe. Reassessment of the 2010–2011 Haiti cholera outbreak and rainfall-driven multiseason projections. *PNAS*, 109(17): 6602–6607, 2012.

[160] D. M. Romero, B. Meeder, and J. Kleinberg. Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 695–704. ACM, 2011.

[161] C. Roth, S. Rastogi, L. Arvestad, K. Dittmar, S. Light, D. Ekman, and D. A. Liberles. Evolution after gene duplication: models, mechanisms, sequences, systems, and organisms. *Journal of Experimental Zoology Part B: Molecular and Developmental Evolution*, 308 (1):58–73, 2007.

[162] S. M. E. Sahraeian and B.-J. Yoon. A network synthesis model for generating protein interaction network families. *PloS one*, 7(8):e41474, 2012.

[163] S. M. E. Sahraeian and B.-J. Yoon. SMETANA: accurate and scalable algorithm for probabilistic alignment of large-scale biological networks. *PLoS One*, 8(7):e67995, 2013.

[164] M. Salathé and J. H. Jones. Dynamics and control of diseases in networks with community structure. *PLoS Computational Biology*, 6(4):e1000736, 2010.

[165] V. Saraph and T. Milenković. Magna: Maximizing accuracy in global network alignment. *Bioinformatics*, 30(20):2931–2940, 2014.

[166] L. Sattenspiel and K. Dietz. A Structured Epidemic Model Incorporating Geographic Mobility Among Regions. *Mathematical Biosciences*, 128(1):71–91, 1995.

[167] L. Sattenspiel and D. A. Herring. Simulating the Effect of Quarantine on the Spread of the 1918–19 Flu in Central Canada. *Bulletin of Mathematical Biology*, 65(1):1–26, 2003.

[168] A. Schlicker and M. Albrecht. Funsimmat: a comprehensive functional similarity database. *Nucleic acids research*, 36(suppl 1):D434–D439, 2008.

[169] B. Seah, S. S. Bhowmick, and C. F. D. Jr. DualAligner: a dual alignment-based strategy to align protein interaction networks. *Bioinformatics*, 30(18):2619–2626, 2014.

[170] D. Shah. *Gossip algorithms.* Now Publishers Inc, 2009.

[171] R. Sharan and T. Ideker. Modeling cellular machinery through biological network comparison. *Nature biotechnology*, 24(4):427–433, 2006.

[172] R. Sharan, S. Suthram, R. M. Kelley, T. Kuhn, S. McCuine, P. Uetz, T. Sittler, R. M. Karp, and T. Ideker. Conserved patterns of protein interaction in multiple species. *Proceedings of the National Academy of Sciences*, 102(6):1974–1979, 2005.

[173] R. Sharan, S. Suthram, R. M. Kelley, T. Kuhn, S. McCuine, P. Uetz, T. Sittler, R. M. Karp, and T. Ideker. Conserved patterns of protein interaction in multiple species. *Proceedings of the National Academy of Sciences of the United States of America*, 102(6):1974–1979, 2005.

[174] B. Shulgin, L. Stone, and Z. Agur. Pulse Vaccination Strategy in the SIR Epidemic Model. *Bulletin of Mathematical Biology*, 60(6):1123–1148, 1998.

[175] R. Singh, J. Xu, and B. Berger. Pairwise Global Alignment of Protein Interaction Networks by Matching Neighborhood Topology. In *Proc. of Research in Computational Molecular Biology 2007*, San Francisco, CA, USA, April 2007.

[176] R. Singh, J. Xu, and B. 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.

[177] K. Sjölander. Phylogenomic inference of protein molecular function: advances and challenges. *Bioinformatics*, 20(2):170–179, 2004.

[178] D. A. Spielman. Faster isomorphism testing of strongly regular graphs. In *Proc. of ACM STOC 1996*, Philadephia, Pen., USA, May 1996.

[179] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

[180] D. A. Spielman and S.-H. Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011.

[181] M. Srivatsa and M. Hicks. Deanonymizing mobility traces: Using social network as a side-channel. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 628–637. ACM, 2012.

[182] L. Stone, B. Shulgin, and Z. Agur. Theoretical Examination of the Pulse Vaccination Policy in the SIR Epidemic Model. *Mathematical and Computer Modelling*, 31(4):207–215, 2000.

[183] S. Suthram, T. Sittler, and T. Ideker. The Plasmodium protein network diverges from those of other eukaryotes. *Nature*, 438(7064):108–112, 2005.

[184] Y. Tanahashi, J. R. Rowland, S. North, and K.-L. Ma. Inferring Human Mobility Patterns from Anonymized Mobile Communication Usage. In *MoMM 2012*, pages 151–160. ACM, 2012.

[185] A. J. Tatem. Côte d'Ivoire AfriPop Data 2010 (alpha version). Emerging Pathogens Institute, University of Florida, 2010. URL http://www.clas.ufl.edu/users/atatem/index_files/CIV.htm.

[186] J. Taubenberger and D. Morens. 1918 Influenza: The mother of all pandemics. *Rev Biomed*, 17:69–79, 2006.

[187] S. A. Teichmann and M. M. Babu. Gene regulatory network growth by duplication. *Nature genetics*, 36(5):492–496, 2004.

[188] A. H. Y. Tong, M. Evangelista, A. B. Parsons, H. Xu, G. D. Bader, N. Pagé, M. Robinson, S. Raghibizadeh, C. W. Hogue, H. Bussey, et al. Systematic genetic analysis with ordered arrays of yeast deletion mutants. *Science*, 294(5550):2364–2368, 2001.

[189] L. Torresani, V. Kolmogorov, and C. Rother. Feature correspondence via graph matching: Models and global optimization. In *Computer Vision–ECCV 2008*, pages 596–609. Springer, 2008.

[190] J. Truscott and N. M. Ferguson. Evaluating the Adequacy of Gravity Models as a Description of Human Mobility for Epidemic Modelling. *PLOS Computational Biology*, 8(10): e1002699, 2012.

[191] United Nations, Department of Economic and Social Affairs. World Population Prospects, the 2010 Revision, 2010. URL http://esa.un.org/unpd/wpp/index.htm.

[192] R. Van Der Hofstad. *Random graphs and complex networks*. Available: https://www.win.tue.nl/~rhofstad/NotesRGCN.pdf, 2016.

[193] M. Vidal, M. E. Cusick, and A.-L. Barabási. Interactome networks and human disease. *Cell*, 144(6):986–998, 2011.

[194] V. Vijayan, V. Saraph, and T. Milenković. MAGNA++: Maximizing Accuracy in Global Network Alignment via both node and edge conservation. *Bioinformatics*, 31(14):2409–2411, 2015.

[195] E. M. Volz, J. C. Miller, A. Galvani, and L. A. Meyers. Effects of heterogeneous and clustered contact patterns on infectious disease dynamics. *PLoS Comput Biol*, 7(6): e1002042, 2011.

[196] B. Von Bahr and A. Martin-Löf. Threshold Limit Theorems for Some Epidemic Processes. *Advances in Applied Probability*, 12(2):319–349, 1980.

[197] T. Wang, Y. Chen, Z. Zhang, T. Xu, L. Jin, P. Hui, B. Deng, and X. Li. Understanding graph sampling algorithms for social network analysis. In *2011 31st International Conference on Distributed Computing Systems Workshops*, pages 123–128. IEEE, 2011.

[198] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393 (6684):440–442, 1998.

[199] G. Wondracek, T. Holz, E. Kirda, and C. Kruegel. A practical attack to de-anonymize social network users. In *Proc. of IEEE Symposium on Security and Privacy 2010*, Oakland, CA, USA, May 2010.

[200] World Health Organization. Summary of probable SARS cases with onset of illness from 1 November 2002 to 31 July 2003, 2004. URL http://www.who.int/csr/sars/country/table2004_04_21/en/index.html.

[201] R. S. Xin, J. E. Gonzalez, M. J. Franklin, and I. Stoica. Graphx: A resilient distributed graph system on spark. In *First International Workshop on Graph Data Management Experiences and Systems*, page 2. ACM, 2013.

[202] L. Yartseva and M. Grossglauser. On the performance of percolation graph matching. In *Proc. of ACM COSN 2013*, Boston, MA, USA, October 2013.

[203] H. Yu, N. M. Luscombe, H. X. Lu, X. Zhu, Y. Xia, N. Han, Jing-Dong Jand Bertin, S. Chung, M. Vidal, and M. Gerstein. Annotation transfer between genomes: protein–protein interologs and protein–DNA regulogs. *Genome research*, 14(6):1107–1118, 2004.

[204] G. Zaman, Y. Han Kang, and I. H. Jung. Stability analysis and optimal vaccination of an SIR epidemic model. *BioSystems*, 93(3):240–249, 2008.

[205] G. Zaman, Y. H. Kang, and I. H. Jung. Optimal treatment of an SIR epidemic model with time delay. *BioSystems*, 98(1):43–50, 2009.

[206] T. Zaman, E. B. Fox, E. T. Bradlow, et al. A bayesian approach for predicting the popularity of tweets. *The Annals of Applied Statistics*, 8(3):1583–1611, 2014.

[207] M. Zaslavskiy, F. Bach, and J.-P. Vert. Global alignment of protein-protein interaction networks by graph matching methods. *Bioinformatics*, 25(12):i259–1267, 2009.

[208] X. Zhang and B. M. Moret. Refining regulatory networks through phylogenetic transfer of information. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 9(4):1032–1045, 2012.

Ehsan Kazemi - Résumé

Address	BC 203, LCA4, IINFCOM, IC, EPFL, Station 14, CH-1015 Lausanne	**Office Phone** **Mobile Phone**	+41 21 693 12 61 +41 78 952 70 55
Date of Birth	16th May 1987	**Email**	ehsan.kazemi@epfl.ch
Nationality	Iranian	**Website**	http://people.epfl.ch/ehsan.kazemi

Education

2011-2016 Ph.D., Computer, Communication and Information Sciences,
Thesis title: "Models and Algorithms for Mining Networks",
Under the supervision of Prof. Matthias Grossglauser,
Laboratory for computer Communications and Applications (LCA4),
EPFL, Lausanne, Switzerland.

2010-2011 Ph.D. candidate, Computer, Communication and Information Sciences,
Unaffiliated Ph.D. student, EPFL, Lausanne, Switzerland
PhD. research project, "Hiding in the Mobile Crowd: Location Privacy through Collaboration."
LCA1, Under the supervision of Prof. Jean-Pierre Hubaux.

2008-2010 MS.c., Electrical Engineering and Communication Systems,
Under the supervisions of Prof. Mohammad Reza Aref and Prof. Taraneh Eghlidos,
Sharif University of Technology, Tehran, Iran.

2004-2008 BS.c., Electrical Engineering and Communication Systems,
Sharif University of Technology, Tehran, Iran.

Honors and Awards

- Won the Mobile Data Challenge (MDC) launched by Nokia Research Center, Lausanne, 2012.
- Chosen among the top students in the B.Sc. program and exempted from the M.Sc. entrance exam of Sharif University of Technology, 2008.
- Ranked first among over 50 students of communication engineering at Sharif University in M.Sc. program.
- Ranked 24th in the nationwide university entrance exam in Iran, 2004.

Publication

- E. Kazemi, and M. Grossglauser. "On the Structure and Efficient Computation of IsoRank Node Similarity." Technical report, 2016.
- E. Kazemi, S. H. Hassani, and M. Grossglauser, and H. Pezeshgi Modarres. "PROPER: Global Protein-Protein Interaction Network Alignment with Percolation Graph-Matching." Accepted to BMC Bioinformatics, 2016. (The PROPER algorithm web interface: `proper.epfl.ch`)
- E. Kazemi, S. H. Hassani, and M. Grossglauser. "Growing a Graph Matching from a Handful of Seeds." Proceedings of the Vldb Endowment International Conference on Very Large Data Bases,vol. 8, num. 10, 2015.
- E. Kazemi, L. Yartseva, and M. Grossglauser. "When Can Two Unlabeled Networks Be Aligned Under Partial Overlap?". In Conference on Communication, Control, and Computing, Allerton Park, Monticello, IL, USA, 2015.

- R. Shokri, G. Theodorakopoulos, P. Papadimitratos, E. Kazemi, and J-P. Hubaux. "Hiding in the Mobile Crowd: LocationPrivacy through Collaboration." IEEE Transactions on Dependable and Secure Computing, 11, no. 3 (2014): 266-279.
- M. Kafsi, E. Kazemi, L. Maystre, L. Yartseva, M. Grossglauser, and P. Thiran. "Mitigating epidemics through mobile micro-measures." arXiv preprint arXiv:1307.2084 (2013). (finalist of the Orange D4D challenge)
- V. Etter, M. Kafsi, E. Kazemi, M. Grossglauser, and P. Thiran. "Where to go from here? Mobility prediction from instantaneous information." Pervasive and Mobile Computing 9, no. 6 (2013): 784-797.
- V. Etter, M. Kafsi, and E. Kazemi. "Been there, done that: What your mobility traces reveal about your behavior." Mobile Data Challenge by Nokia Workshop, in conjunction with Int. Conf. on Pervasive Computing. 2012. (won the NOKIA Mobile Data Challenge on the Next Place Prediction)

Supervised Student Projects

- Weizhe Liu, "Author Inference of Doubly Blind Papers." 2015.
- Silas Kieser, "Complex identification in multispecies PPI Networks." 2015.
- Frédéric Moret, "Web Interface for PPI Network Alignment." 2015.
- Maxime Coriou, "Finding Biological Pathways by Global Alignment of PPI networks." 2015.
- Jéremy Weber, "MapReduce Implementation of Percolation Graph Matching Algorithms." 2014.
- Abdul Wasay, "Network De-anonymization Using Community Detection." 2013.
- Rodrigo Paim, "Network Inference with Structural Ambiguities." 2013. This project won the best internship of the year (prix du meilleur projet de recherche) among more than 200 competitors from ENSTA ParisTech.

Skills

- Programming languages: Python, C/C++ and JAVA
- Softwares: Hadoop (MapReduce), Matlab, Maple, Linux, $\LaTeX$ and Git

Languages

- Farsi: Native
- English: Fluent
- French: Basic

Teaching Assistant

- Internet Analytics, Mobile Networks, Principles of Digital Communications, Analysis I/II, Satellite Communication, Advanced Cryptography, Introduction to Cryptography, Circuits and Systems and Electrical Measurement.

Interests

- Walking long distances, running (my 10km record is 48m:15s), cycling (the longest distance I have biked in one day is 180km) and hiking
- Amatour photography
- Novel and poetry enthusiastic

Referees

Name	Matthias Grossglauser
Affiliation	LCA4, IC, EPFL
Position	Associate Professor
Contact	matthias.grossglauser@epfl.ch

Name	Patrick Thiran
Affiliation	LCA3, IC, EPFL
Position	Full Professor
Contact	patrick.thiran@epfl.ch

Name	Amin Karbasi
Affiliation	I.I.D., EE, Yale
Position	Assistant Professor
Contact	amin.karbasi@yale.edu

Name	Forrest W. Crawford
Affiliation	Biostatistics, Yale
Position	Assistant Professor
Contact	forrest.crawford@yale.edu

Name	Hamed Hassani
Affiliation	LASG, ETHZ
Position	Post-doctoral Scholar
Contact	hamed@inf.ethz.ch