Alignment and Assembly: Inferring Networks from Noisy Observations

THÈSE N^{O} 7562 (2017)

PRÉSENTÉE LE 10 AVRIL 2017

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

Lyudmila YARTSEVA

acceptée sur proposition du jury:

Prof. B. Rimoldi, président du jury
Prof. M. Grossglauser, Prof. M. A. Shokrollahi, directeurs de thèse
Prof. A. Chaintreau, rapporteur
Prof. M. Garetto, rapporteur
Prof. P. Frossard, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2017

The limits of the possible can only be defined
by going beyond them into the impossible.
— Arthur C. Clarke

When faced with competing hypotheses, select the one
that makes the fewest assumptions.
Do not multiply entities without necessity.
— Occam's razor

Acknowledgements

I will begin by acknowledging my main supervisor: Matthias Grossglauser, I am happy that I had a chance to work and learn from such a great scientist and talented supervisor. Thank you for your brilliant intuition and your immense patience and understanding. There was not a single time when I asked for help and did not get it. I came to the meetings with new inspiring results or with devastating errors in the papers and I always found support and inspiration. Thank you, Matthias, the best supervisor I could dream of (or I could imagine)!

At the end of first year at EPFL was a difficult moment of setting up my PhD and at this time I was lucky to meet Professor Amin Shokrollahi, a very kind person who became my co-advisor and to whom I am deeply grateful for helping me in a tough moment. I enjoyed working in such a well organized environment and this allowed me to start my research smoothly. I would like to also thank Professors Rüdiger Urbanke, the head of the EDIC at the point in time, and Bernard Moret, my mentor, who in the same period provided me with a high level of involvement and support.

I also would like to thank my jury members for accepting to come at such an inconvenient time and to read my thesis as their Christmas reading: Augustin Chaintreau, Pascal Frossard and Michele Garetto. Also I thank Bixio Rimoldi for being the president.

I was happy to work at EPFL, with my brilliant colleagues: Elisa Celis,Vincent Etter, Sébastien Henri, Julien Herzen, Mohamed Kafsi, Emtiyaz Khan, Victor Kristof, Lucas Maystre, Pedram Pedarsani, Farnood Salehi, William Trouleau and Christina Vlachou. I worked together and learned a lot from my colleagues Ehsan Kazemi and Jefferson Ebert, thank you for productive discussions and for finding mistakes in my proofs.

My dear friends, thank you for support, chats, coffee, lunches, hikes, travels and everything else: Brunella Spinelli, I am glad that we were put to be officemates and you became my good friend; Marc Desgroseilliers, who introduced me to rock climbing ;-); Nastya Tychinskaya for being unfailing and supportive friend; Young-Jun Ko for your support, empathy and sense of humor and to Karol Kruzelecki, Stefano Rosati, Andrej Spielman, Adrian Tarniceriu and to all who I forgot to mention.

I thank our lab secretaries Angela, Danielle, Holly and Patricia for organizing our life here and for always being so nice. Thank you Marc-André, Stéphane and Yves for the IT support. I also thank Damir for your IT help, quick and efficient. Of course, I give special thanks to Holly for editing our work, teaching, sharing wisdom and always being patient and making it such a fun activity.

Finally, I want to thank my mom Elena Yartseva, who taught me everything, to whom I owe

big part of my math education and who always set the bar high for me.
My husband, who has supported me at every step, covering my abrupt decisions and unfinished business; without him it would be just impossible to come here and make it through.
My daughter who expanded my world to a new dimension. There are no words to describe what she means to me and what she brings to my life.

Lausanne, 16 January 2017 L. Y.

Abstract

Over recent years, many large network datasets have become available, giving rise to novel and valuable applications of data mining and machine learning techniques. These datasets include social networks, the structure of the internet, and protein-interaction networks in computational biology, to name just a few. Graph mining exploits information hidden in these data to shed light on such problems as identifying a source of an epidemic in a human contact network, finding relevant pages on the web, or identifying communities of strongly connected individuals. Clearly, to address such problems, we first need the accurate and reliable network graph. This thesis is about obtaining such a full graph from raw data with imperfections.

In many real-world scenarios, the full graph is not available for free. For example, data-collection processes may be noisy and unreliable, names of users in social networks may not be unique, or node identifiers may be hidden for privacy protection. Therefore, we cannot rely on the global uniqueness of node labels to infer the full graph. In addition, data is often provided in a form of small local observations: For example, given a set of papers with corresponding authors, we need to reconstruct the whole co-authorship network. In the hardest case of completely ambiguous labels, we can only rely on structural information to obtain the full graph. In this thesis, we address fundamental and practical questions of inferring a true full network from multiple observations, each one of which is subject to noise and to label ambiguity.

We formulate two variations of this problem: *network alignment* and *network assembly*. In each variant, we address two types of questions: first, we characterize how graph features impact the fundamental feasibility of reconstruction, regardless of computational cost; second, we seek efficient algorithms that can scale to very large networks, and provide performance guarantees under some classes of networks. We use random graph models for both feasibility and performance analysis. We also evaluate our algorithms over real network data.

In the first part of this thesis, we consider *network alignment*. We assume two large, noisy observations of the true network; in addition, we assume that the node labels in one network are meaningless in the other, and vice versa. Network alignment refers to the problem of aligning the vertex sets of the two networks using only structural cues. A motivating example is the deanonimization of a social network graph by aligning it with a side information graph. The network alignment problem can be viewed as a generalization of the classic graph–isomorphism problem. We make the following contributions. First, we introduce a random bigraph model that generates two correlated random graphs G_1 and G_2, by removing

some nodes and edges from the true graph G with probabilities t and s, respectively. We analyze the feasibility of aligning G_1 with G_2 generated from Erdős-Rényi random graph, where every edge exists with identical probability p. We characterize conditions on p, t and s for the feasibility of graph alignment. Second, we create an algorithm named percolation graph-matching (PGM) that takes a small set of pre-matched nodes S, called seeds, and then incrementally maps each pair of nodes (i, j) with at least r neighboring mapped pairs. We prove conditions on the model parameters p, t, s and r for which percolation graph-matching succeeds, and we establish a phase transition in $|S|$.
In the second part of this thesis, we consider *network assembly*. We assume many small, noisy observations of the true network, called patches. The node labels are either absent or not unique, making the reconstruction problem nontrivial. The network assembly problem consists in reconstructing the true graph from these patches. For example, a professional social network might be reconstructed from social networks of small organizations, or the scientific co-authorship network might be reconstructed from individual papers. We make the following contributions. First, we introduce and analyze a novel random-graph model called $G(n, p; q)$; this model starts with an Erdős-Rényi graph $G(n, p)$; where each triangle is closed with probability q. The interest of this model is to generate high clustering (or transitivity), a salient property of real networks. We characterize feasibility conditions on p and q such that reconstruction of G is possible, even from small patches with structural noise. Second, using this result, we build a practical algorithm that relies on canonical labeling to reconstruct the true graph from noiseless patches. We also propose a heuristic assembly algorithm that tries to reconstruct the true graph, without a priori assumptions about the sizes of subgraphs and label ambiguity.

Key words: network analysis, graph mining, complex networks, network reconstruction, graph alignment, graph assembly, network privacy, random graphs, graph isomorphism, graph matching, social networks

Résumé

Ces dernières années, de nombreuses données portant sur des réseaux à grande échelle ont été rendues publiques, engendrant de nouvelles applications précieuses pour des techniques d'exploration de données ou d'apprentissage automatique. Ces nouvelles données portent notamment sur certains réseaux sociaux, sur la structure d'Internet ou, en biologie computationnelle, sur des réseaux d'interaction entre protéines. Les techniques d'exploration des graphes utilisent l'information cachée dans ces données pour résoudre des problèmes tels que l'identification de la source d'une épidémie dans le réseau des interactions sociales humaines, la recherche des pages les plus pertinentes sur Internet, ou l'identification de communautés d'individus fortement liés les uns aux autres. Il est clair que pour pouvoir résoudre ces problèmes, il est nécessaire d'avoir accès à un graphe fiable et intégral. Cette thèse s'intéresse aux moyens d'obtenir de tels graphes intégraux à partir de données partielles et imparfaites.

Dans de nombreux scénarios de la vie réelle, le graphe entier n'est pas disponible facilement. Par exemple, les processus de collection des données peuvent être bruités et non-fiables, les noms dans des réseaux sociaux peuvent ne pas être uniques, ou les identifiants peuvent être cachés pour protéger ca vie privée. En conséquence, il n'est pas possible de compter sur l'unicité de l'étiquette des nœuds pour inférer le graphe intégral. De plus, les données sont souvent fournies sous la forme d'observations locales à petite échelle : par exemple, à partir d'un ensemble d'articles avec le nom des auteurs, il faut reconstruire l'ensemble du réseau des collaborations entre auteurs. Dans le cas le plus difficile, avec des étiquettes complètement ambiguës, on ne peut compter que sur les informations structurelles pour obtenir le graphe intégral. Dans cette thèse, nous nous intéressons à la question pratique et fondamentale de l'inférence du réseau intégral réel à partir d'observations multiples, chacune de ces observations étant sujette au bruit et à l'ambiguïté de son étiquette. Nous formulons deux variantes de ce problème : l'alignement de réseaux et l'assemblage de réseaux. Pour chaque variante, nous nous intéressons à deux types de questions : d'abord, nous caractérisons la manière dont les caractéristiques des graphes influent sur la faisabilité fondamentale de la reconstruction, quel qu'en soit la complexité de calcul ; ensuite, nous cherchons des algorithmes efficaces pouvant passer à l'échelle sur de très grands réseaux, et nous donnons des garanties de performance pour certaines classes de réseaux. Nous utilisons des modèles de graphes aléatoires à la fois pour la faisabilité et pour l'analyse de performance. Nous évaluons également nos algorithmes sur des réseaux réels.

Dans la première partie de cette thèse, nous considérons l'alignement de réseaux. Nous sup-

posons avoir deux versions différentes, bruitées, d'un réseau réel à grande échelle. En outre, nous supposons que les étiquettes d'un des deux réseaux bruités n'ont aucune signification dans l'autre réseau, et inversement. L'alignement de réseaux correspond au problème de l'alignement des nœuds des deux réseaux bruités en utilisant uniquement la structure de ces réseaux. Un bon exemple d'application est la dé-anonymisation du graphe d'un réseau social par son alignement avec un graphe d'informations issues d'une autre source. Le problème d'alignement de réseaux peut être vu comme la généralisation du problème classique d'isomorphisme des graphes. Nous apportons les contributions suivantes. Premièrement, nous présentons un modèle de graphe biparti aléatoire qui permet de générer deux graphes aléatoires corrélés G_1 et G_2 en enlevant des nœuds et des liens du graphe réel G, avec probabilité respectivement t et s. Nous analysons la faisabilité de l'alignement de G_1 avec G_2 quand ils sont générés par des graphes aléatoires de Erdös-Renyi, où chaque lien existe avec la même probabilité p. Nous caractérisons les conditions sur p, t et s pour la faisabilité de l'alignement de réseaux. Deuxièmement, nous présentons un algorithme, appelé couplage de graphe par percolation (PGM, Percolation graph-matching), qui utilise un petit nombre de nœuds, appelés graines, dont on connaît la correspondance, et qui associe de manière incrémentale chaque paire de nœuds (i, j) avec au moins r paires voisines déjà associées. Nous démontrons que sous certaines conditions sur les paramètres du modèle p, t, s et r, l'algorithme PGM renvoie le bon résultat, et nous établissons une transition de phase liée à $|S|$.

Dans la deuxième partie de cette thèse, nous considérons l'assemblage de réseaux. Nous supposons avoir accès à plusieurs petites observations bruitées, appelées pièces, du vrai réseau. Les étiquettes des nœuds sont soit absentes soit non-uniques, ce qui rend le problème de reconstruction non-trivial. Le problème d'assemblage de réseaux consiste en la reconstruction du graphe réel à partir de ces pièces. Par exemple, un réseau social professionnel peut être reconstruit à partir des réseaux sociaux de petites organisations, ou bien le réseau des collaborations entre auteurs reconstruit à partir des articles uniquement. Nous apportons les contributions suivantes. Premièrement, nous présentons et analysons un nouveau modèle de graphes aléatoires, appelé $G(n, p; q)$; ce modèle part d'un graphe de Erdös-Renyi $G(n, p)$, et chaque triangle est ensuite fermé avec probabilité q. L'intérêt de ce modèle est de générer un fort cloisonnement, propriété remarquable des réseaux réels. Nous caractérisons les conditions sur p et q pour lesquelles la reconstruction de G est possible, même à partir de petites pièces avec un bruit structurel. Deuxièmement, nous construisons à partir de ces résultats un algorithme pratique qui utilise l'étiquetage canonique pour reconstruire le graphe réel à partir de pièces non-bruitées. Nous proposons également un algorithme d'assemblage heuristique qui tente de reconstruire le graphe réel sans faire d'hypothèse sur la taille des sous-graphes ou sur l'ambiguïté des étiquettes.

Contents

1 Introduction

Graphs are natural means for modeling various datasets that contain entities and their interactions. The application areas vary from the analysis of graphs of social networks [81, 2, 25, 57] to the modeling of protein-protein-interaction(PPI) networks in biology [106, 99]. The following are examples of problems that are solved by network analysis: the efficient routing of packets through computer networks [72], community detection in social networks [22, 78], identifying the source of an epidemic in a human interaction network [65, 8, 91, 81] and finding a relevant page in the web [7]. Clearly these applications rely on the assumption that the network-graph is accurate. Unsuitable approaches to obtaining the network graph can lead to fundamentally incorrect results. Consider the following example: the degree-distribution of the Internet graph is one of the important characteristics for building Internet applications. The famous work by Faloutsos et al. [43] claimed a power-law degree distribution in the graph of routers in the Internet, by sampling traceroute packets. It was shown later that this graph reconstruction method leads to a biased degree-distribution of the resulting graph [73]. The cause of this phenomenon is that, when sampling from few sources to multiple destinations, edges close to the source are sampled more often than edges further away. These erroneous observations could lead to inefficient search and routing strategies and to unbalanced load distribution over different links.

In some contexts it is straightforward to obtain the correct network: For example, the IP-network of one domain is reconstructible from IP-addresses; and the social network inside an organization can be obtained from e-mail exchanges. However, with advances in data analysis, we have to work more often with human-generated data, data needed to be merged from different sources or data that are itself an output of some analysis tools [44, 16]. Moreover, in dealing with graphs over more complex objects the concepts of a “node” and an “edge” are not defined. For example, suppose we are given a corpus of text, each piece describing social interactions and transactions between social contacts, i.e., characters in a play or a novel. Clearly, in the inferred social network, there are multiple types of connections possible (co-occurrence, explicit relation, etc.). This situation arises in digital humanities [77, 94, 111], for example, in a project of processing a historical archive of Venice in [58] (which takes 80 km

of shelving of historical documents over more than 1000 years). One of the challenges is to extract a social network from the contracts, taxation documents, etc. Naturally, there are multiple references for an individual, which gives rise to label ambiguity. For another example of graphs over complex objects, consider the entity resolution problem [1, 46], that initially arose in databases analysis and that addresses the problem of disambiguating references to real world entities. Consider a Facebook entity graph where we deal with multiple types of entities: people, companies or geographical-locations, etc. Some objects have multiple types, for example, McDonald's is a company and a location. Edges have multiple types as well, which results in multilayered graphs [19]. The last example refers to networks changing over time: one approach to identifying the source of an epidemic is to analyze human contacts during the epidemic. Understandably, human behavior (hence interactions graph) changes during epidemics [57, 82]; for some problems it is sufficient to consider a static network, but it is important to carefully select and reconstruct this network. However, it is not straightforward to define a static graph [110, 82].

We roughly classify the difficulties of reconstruction of the network into two types: structural noise and label ambiguity. The first type, *structural noise*, involves missing nodes and edges, as well as nodes and edges erroneously included in the network: for example, when crawling a social network from the web, very often we have some parts of the networks that remain hidden. One reason for this is deliberate privacy protection: users hide their friend circle by adjusting their privacy settings [52]. Another reason data may be missing is because we need to sample networks due to the scalability issues [47, 30, 76]. Different sampling methods have different drawbacks and introduce errors in measuring the main network characteristics: for example, the random sampling of nodes or local neighborhoods leads to overestimating the path-length of the considered graphs. Another, breadth-first search sampling provides incorrect estimators of characteristics, such as degree distribution or clustering coefficient (density of a node's neighborhood), because the samples are biased towards observing high degree nodes [71]. Another domain, where we find examples of structural noise, is in biology in the analysis of protein and gene-interaction networks. Protein–protein interactions (PPIs) refer to physical contacts between two protein molecules as a result of biochemical events in a cell or in a living organism, in a specific biomolecular context [37]. The analysis of PPI networks enables us to find proteins with common functions in different species and to shed light on questions such as connection between network motifs and cancer development in the organism [12]. Currently, the methods for registering protein-protein interactions are certain for only a relatively small fraction of interactions, thus introducing a large fraction of false positives and false negatives [17, 112]. This creates false and missing edges and makes the global network analysis difficult [11, 97].

We call the second source of uncertainty *label ambiguity*; these are cases where an object is referred to by different labels or several objects are referred to by the same label. For example, a typical individual or unit belongs to several networks, but can possess different identifiers in different networks [26], making it very difficult to cross-identify users among these networks, whereas many unrelated individuals can use the same name or nickname in the same net-

work. In the text corpus example from digital humanities, see [58], each character is referred to by ambiguous identifiers for the protagonists, e.g., first name, nickname, or some descriptive reference. Another example refers to creating a co-authorship graph from a dataset of publications (for example computer science publications from DBL, see [20]) where names are not unique and also formating style differs from conference to conference.

Consequently, in general, to construct a network we cannot rely completely on either labels or structure of the raw data. In this thesis, we address a natural question: Given only noisy observations, how can we find/approximate the correct network?

1.1 Network Reconstruction Problem

The preceding discussion explains the importance and challenges of having a correct network or at least its approximation. We now turn to the central part of this thesis that addresses a question about the reconstruction of a network from imperfect data, i.e., data having structural noise and/or label ambiguities. To define the network reconstruction problem, we assume the existence of a correct solution that is some underlying true network we try to learn. In the case of social networks, it can be all the people and relationships among them [98]. In the case of computer networks, it can be the actual machines and routers and physical connections. In the most general form we define a *master graph* $G(V,E)$, which represent this true network. Then *network reconstruction* refers to restoring the master graph G from multiple different noisy observations. Below, we elaborate on different types of observations and specify the reconstruction problem, respectively.

This work addresses the questions about network reconstruction and consists of two major parts: (i) One part assumes we have two large correlated observations of the master graph, and we are interested in finding the correct bijection between the two vertex sets under some conditions (or at least part of the bijection), hence inferring knowledge of the master graph. This problem is referred to as *network alignment*; (ii) the second part assumes that we have multiple small observations of the network and we are interested in reconstructing the whole master graph, this is called *network assembly*. We now define the problems more formally.

Network Alignment. The first part of this thesis addresses the question about the alignment of two graphs $G_1(V_1,E_1)$ and $G_2(V_2,E_2)$. These graphs can be viewed as observations or realizations of a master graph G. For example, G can be a real social network, G_1 can be personal connections observed via Facebook graph, and G_2 can be work connections observed via e-mail exchanges in the organization. See Fig 1.1, for an example of input of the problem.

We are looking for a *matching* between the two graphs, defined as follows:

Definition 1.1 (Matching)**.** *A partial matching π between two graphs is a bijective partial function $\pi : V_1 \to V_2$. Let Π be the set of all partial matchings π from the vertex set V_1 to V_2.*

Note that in earlier works [61], the matching is defined as restriction of having only one coun-

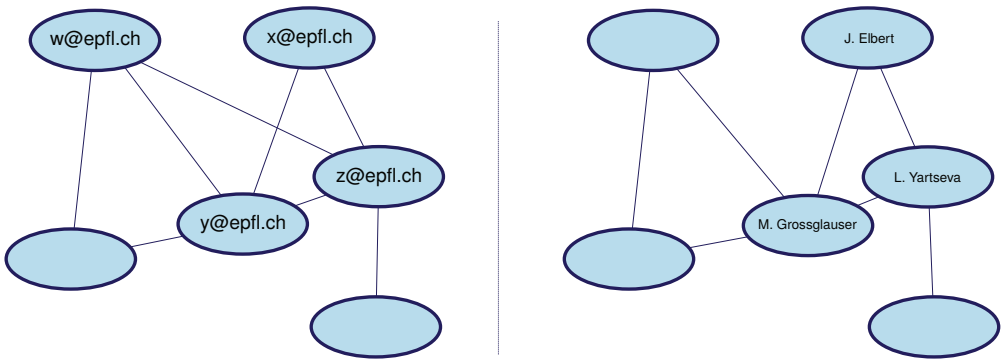

Figure 1.1 – The graphs of an anonymized e-mail exchange and a LinkedIn crawl within the same organization.

terpart for each node. These two definitions are equivalent, and the bijective conditions make the definition symmetric for V_1 and V_2.

There are multiple names for the problem: it is known as "network reconciliation", "network alignment" or "graph matching"[1]. We use the term "network alignment" and refer to the term "matching " to address the bijection itself.

In the above example, some users are present in both networks. Without loss of generality, we denote the nodes of G observed in both G_1 and G_2 by V_0. Further, we slightly abuse notation and say $V_0 = V_1 \cap V_2$. Then we can define a correct matching $\pi_0 : V_1 \to V_2$ s.t. for any $u \in V_0$ $\pi_0(u) = u$ or, equivalently, π_0 is $\{(u,u); u \in V_0\}$. The network-alignment problem is defined formally as follows.

Definition 1.2 (Network-Alignment Problem)**.** *Given the two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$, the goal is to find the correct matching π_0.*

The application areas vary: on the one hand, network alignment enables us to enrich users information, see [68, 60], as aligning graphs G_1 and G_2 gives us better approximation of the true network G than each graph separately. On the other hand, if one of the graphs is anonymized, aligning these graphs means de-anonymizing it [60, 85, 89, 32, 31], see details and examples in Section 1.2. Theoretically, the feasibility of an alignment is a generalized form of graph isomorphism: in the noiseless case of perfect observations, finding the alignment means finding an isomorphism between the two graphs.

We are interested in the fundamental question about the feasibility of aligning the vertex sets through structural information, in the extreme case when node labels have very little or no meaning. We answer the question about whether the structure of the two graphs reveals the correspondence of some or all of the vertices. See details of our contribution in Section 1.3.1.

[1] In the first works on this problem (see [117], [88]) it was addressed as "graph matching", however, this term is ambiguous because it also refers to another problem of finding an independent edge set of the graph.

Network Assembly. The problem of network assembly is addressed in the second part of the thesis and refers to the problem of reconstructing the master graph G from many, noisy, ambiguous observations. These observations we call *patches*; they are extracted from the master graph. The problem consists of putting these pieces together in an assembled graph $\hat{G}$. We define the input of the problem in its most general form as follows.

Definition 1.3 (Patch Collection)**.** *A* patch collection *is an indexed family of graphs* $\mathscr{P} = \{G_i = (V_i, E_i)\}_{i \in I}$*, for some set of indices* I*.*

Definition 1.4 (Patch Generation)**.** *We say that a patch collection* $\mathscr{P} = \{G_i(V_i, E_i)\}_{i \in [n]}$ [2] *is generated from a graph* $G(V = [n], E)$ *if it is endowed with a set of functions* $\{f_i\}_{i \in [n]}$ *, called* patching functions*. For each patch* f_i *is a bijection from some* $V_i^G \subseteq V$ *to* V_i*.*

For example, if a full social network cannot be released, due to the concern that this network could be deanonymized, one protection mechanism that has been used in the literature is the release of all the 1-hop egonets of this network, with all node identities withheld [21]. Reassembling the network would enable us to study its features [102], however this would endanger the anonymity. Another example where a full network needs to be reconstructed is DBLP co-authorship graph mining (DBLP collects bibliographical data about publications at conferences and journals from different sources); the graph is used for analyzing scientific communities and collaborations, and for identifying influential researches [115, 20] or revealing security threats [27]. However, multiple issues occur due to the different formats of naming for different conferences, translations and multiple sources of data extraction. See Figure 1.2 for example. In both of these cases, it is useful to have a complete graph for analysis, however it is not clear how and whether it is feasible to reconstruct this graph. Therefore, we address graph assembly from multiple ambiguous patches with no or few labels.

The network-assembly problem is defined formally as follows.

Definition 1.5 (Network-Assembly Problem)**.** *Given a patch collection* $\mathscr{P} = \{G_i = (V_i, E_i)\}_{i \in I}$*, the goal is to find an* assembly*, that is a pair* $(\hat{G}, \{a_i\}_{i \in I})$*, where* $\hat{G}(\hat{V}, \hat{E})$ *is a graph (called* assembled graph*) and each* $a_i : [V_i] \to [V]$ *is an injective function.*

We do not specify here any criteria about how well an estimator $\hat{G}$ approximates a master graph. Note, that for different variations of the problem, additional conditions might be imposed, such as for labeled graphs we might require label consistency. In Section 1.3.2 we define several variations of the problem.

[2] By notation $[n]$ we mean a set of integers from 1 to n.

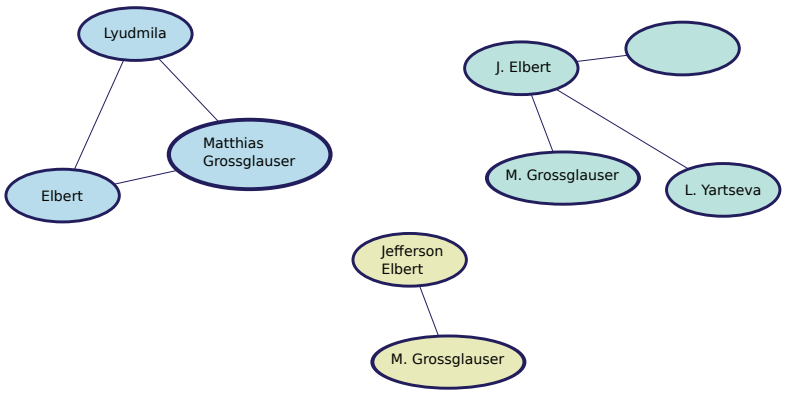

Figure 1.2 – The co-authorship patches extracted from the publications dataset.

1.2 Related Work

1.2.1 Network Alignment

We group the work related to network alignment in three categories. One category consists of the works driven by particular applications, such as biological network alignment, de-anonymizing networks with implications in privacy, and enriching user information in social networks. The second category includes works on heuristic algorithms of network alignment. The last group is related to the theoretical feasibility of alignment and the graph isomorphism problem.

Among the works from the first category, aligning vertex sets of two graphs has strong implications for privacy. A naive way of protecting user's privacy, while releasing data, is replacing their identities with random unique IDs, a process known as a naive anonymization. This method and similar mechanisms to preserve users' privacy are proposed in [118, 49]. However, recent works [114, 15, 85, 88] demonstrate that naive anonymization of the network is not enough to protect users when releasing network graphs. Network alignment is considered as a privacy violation in [48, 69]; in [67] authors analyze the different types of attacks to online social networks. In their seminal work [85], Narayanan and Shmatikov succeed in matching a large-scale anonymized social network to a second social network that serves as side information. Although the node labels in the first network contain no information, the privacy of the network is compromised through the knowledge of a correlated secondary network that has node identities. Their algorithm starts with a small set of prealigned nodes, named *seed set*, and adds new pairs to the matching, based on the number of common neighbors and other statistics. However, their algorithm does not provide any guarantees or an analysis of the performance for different input parameters. Although, the authors observe a threshold behavior in the size of the seeds set. A similar approach of expanding a seed set is used in [70]. Overall, this body of work leaves two questions open: does there exist an algorithm of alignment of two networks from seeds with provable guarantees? And is it feasible to align two networks with no side information?

On the positive side, aligning two graphs of networks from different domains enables us to enrich information about the structure of the network and to correct structural errors. In [117], we propose a *percolation graph matching algorithm* (PGM) by analogy with bootstrap percolation theory; the alignment algorithm "percolates" from initially prealigned nodes called seeds. We show that a sub-linear seed-set size can suffice for a matching with zero-error in some circumstances. In [60] Kazemi et al. drastically decrease the necessary number of seeds but with a small payoff of negligible fraction of errors. Korula and Lattanzi [68] provide an analysis for preferential-attachment generator graphs and random Erdős-Rényi graphs. They consider a regime of very dense seeds, where the mapping for a constant fraction of nodes is known a-priori. In this regime, for Erdős-Rényi network, they show that most of the network can be matched, with high probability, in a single propagation step. Their argument relies on every non-seed node having several seeds as neighbors, which means that non-seed nodes can be matched in a single step directly from these neighbors. They also extend the result to the preferential attachment graphs. In [31] Chiasserini et al. consider a more realistic network model, Chung-Lu [3, 53] random graphs. The graphs sampled from this model have scale-free node degree distribution. The authors consider two regimes: (1) in the first regime the algorithm has randomly selected seeds, and the authors showed that $n^{\frac{1}{2}+\varepsilon}$ seeds are enough to assure almost complete alignment; whereas (2) in the other regime the algorithm can select particular seeds and, in this case, n^{ε} seeds are enough. In [32] the authors analyze the number of seeds and number of errors for random graphs with high clustering (random geometric graphs [90]).

One question to consider about this class of algorithms is an efficient method for searching for seeds. Several ideas are proposed, such as injecting small subgraphs [15], a manual inspection of distinguishable (for example, by degree) nodes [85]. In [87] the authors proposed Bayesian framework for seedless graph matching. They proposed an algorithm that uses nodes features such as degree and distances to other nodes as fingerprints. The algorithm merges likely pairs in rounds and after each round it generates additional features of the unmatched nodes. The theoretical side of the question (selecting seeds with guarantees), however, remains open.

There is another significant branch of application-driven research on network alignment in biology. The alignment of gene and protein networks helps us to infer and predict motifs and to find proteins with common functions among species [105, 103, 106]. Aligning protein-protein-interaction (PPI) networks helps us to increase confidence in the interactions that occur in multiple species, thus to reduce noise in the network. However, most of the work is focused on finding a local alignment, for example, identifying conserved small subgraphs (motifs or pathways) across different species. In [63, 62], the authors search for a small-to-large graph alignment. The known results on global alignment rely on node-labels and some very restricted structural information hence, do not provide either performance guarantees or feasibility analysis [106, 79].

A second group of works contains heuristic algorithms: a few algorithms are proposed, based

on the formulation of network alignment as an optimization problem, and they use linear programming relaxations [66, 40] and belief propagation [18] to efficiently compute an approximate solution. They do not provide, however, performance/optimality guarantees and, usually, do not scale well. Network alignment also arises in other fields, such as ontology alignment. Several automated tools were created to match sets of labels describing data [38, 104, 96]. The specifics of the problems assume small-scale graphs [38], and the algorithms rely heavily on the properties and attributes of the nodes. It is shown in [50] that structural features are much more powerful for graph mining tasks such as network classification and de-anonymization.

The last group of works related to network alignment are theoretical contributions. Interestingly, network alignment can be considered as a generalization of the classic subgraph isomorphism problem or a maximal common subgraph problem. Both are known to be NP-complete [35]. For specific classes of graphs, more is known: for example, for the Erdős-Rényi random graph $G(n,p)$ [42], the threshold function for asymmetry is known to be $p = \log(n)/n$ (see [14, 24]), and to have symmetries clearly implies the impossibility of unique alignment. The class of graphs that appear the most challenging is thought to be the strongly regular graphs [109]. In addition, in the scenarios considered in this thesis, the two graphs are subject to noise and uncertainties, hence we want to check whether two graphs are "inexactly" isomorphic [34] and the problem that we address is similar to an approximate or inexact isomorphism [10].

In another seminal work, a random graph model served to provide insight into the fundamental feasibility of graph alignment for an adversary with unlimited computational power [88]. Pedarsani and Grossglauser demonstrate that under rather benign conditions, two graphs can be aligned perfectly. They introduce a metric thats quantifies the quality of the matching and is minimized by the correct matching (The metric is the number of mismatched edges under fixed matching of two graphs). The drawback of their work is an unrealistic assumption of complete node overlap; we relax this assumption in this thesis. Later in [36] this achievability bound was improved by a factor s, also a converse bound was stated with only a constant-factor gap with the achievability bound. However, the authors assume the full node overlap of the two graphs.

1.2.2 Network Assembly

Network assembly is a fundamental problem related to graph reconstruction, network disambiguation, pattern search and subgraph isomorphism, and it was addressed in areas such as data mining, machine learning and in theoretical computer science. In the same manner as for network alignment, we classify related work into three categories: theoretical works on the feasibility of assembly and subgraph isomorphism, application driven research and, finally, heuristic algorithms.

The first group of works address the feasibility of graph assembly. Under partial or full node-

ambiguity, reassembling the true graph from small subgraphs (called *patches*) is an interesting statistical and computational problem. It is related to the *reconstruction conjecture*, that is formulated by Kelly[64], it addresses the question of a graph G being uniquely identifiable by all its subgraphs obtained by deleting one vertex from G (this collection of subgraphs is called a deck). In [23] Bollobás shows that almost all graphs are reconstructible but, in general, the conjecture stays unproven. A closely related problem was considered recently by Mossel et al. [83, 84], who are also interested in the graph assembly problem; they address a network assembly for several graph models with a low clustering coefficient. Among other models, they consider the assembly of Erdős-Rényi and random regular graphs. They find thresholds for the feasibility of the assembly expressed as a function of the graph density and of the radius of patches. They find that the patches still have to be quite large (though smaller than in the conjecture) for assembly to be feasible. For the Erdős-Rényi random graph for a sparse regime with np a constant, the patch radius r has to be $\Omega(\log n)$. For the denser regime with $np \gg \log^2 n$, assembly with $r = 3$ is feasible. The idea is that, in this case, each node has a unique neighbor-degree sequence, thus this node is identifiable in other egonets. We conjcture that $r = 3$ is required because of the lack of transitivity (short cycles) in such graphs and we prove that the assembly is feasible even for $r = 1$ if the graph is more clustered.

Problems of the feasibility of an assembly, similar to ours (in flavor) are considered in the area of genome assembly. In [74, 28], the authors state theoretical limits for assembling a genome sequence from multiple noisy reads and answer the fundamental question about whether there exists a unique reconstruction. In another work [108], the authors consider a problem of reconstructing a neural network (a graph formed by a set of neurons and connections between them) and propose a method of observing small samples of the network. The goal of the method is to infer the whole network out of these observations. Neurons have several types that can serve as labels of the nodes.

In the second group of works, a large bulk of research is in the field of databases, in particular, entity resolution: several interesting solutions for resolving ambiguities in data are known [1, 59]. However they rely mostly on the similarity between the labels of the entities and entity features, hence, text mining, and the later phases involve some structural information through features. The main drawback of these traditional approaches is that they do not use the graph structure overall, instead they search for a statistical explanation of the data. These approaches are more suitable for relational databases and do not scale well in general [33].

Another important application of network assembly is (KG) Knowledge Graph construction. A KG was proposed by Google to improve the quality of search data [107]. The idea is to shift from search with keywords over documents to search with entities over organized data. The challenges include frequent updates and rapid growth of available data, multiple sources of noise and a high diversity of data [93, 92]. There are several approaches to constructing the KG, including the entity resolution approach described above. These methods rely on the text analysis of entities data; they incorporate structural information as a first step to cluster

entities [59] and merge them later, based on some probabilistic criteria. However we think that a more profound use of the network structure will push the quality of the data on the new level.

The last examples of the application-driven research are several works that address the network-assembly question under the authorship-name ambiguity problem: this problem arises because different authors publish under the same name or the same author publish under various names due to abbreviations, nicknames, etc. In [80], the authors introduce a ranking-based name-matching algorithm for solving the name-ambiguity problem. In [95], graph-based algorithms solve the name-ambiguity problem as follows, they first construct a graph by creating a node for each ambiguous name and then employ some clustering algorithms to find and merge duplicate entities.

The last group of works contains several heuristic algorithms: In the first example [102, 101], Sharad et al. used a machine-learning approach to reassemble the call-network graph from small ego-graphs; where an ego-graph is a graph induced by phone calls of one individual. This data was released during The Data for Development (D4D) Challenge [21] for the advancement of quality of life in Ivory Coast. The authors show that the original phone-call graph is reconstructible from a set of ego-graphs. The result shows that these anonymization techniques do not ensure data privacy. The second example of the group is related to patterns discovery. In [6], the authors address a question about constructing a graph with ambiguous labels. They look at the problem of approximate labeling of the nodes and they propose and minimize some distance function rather than focus on using structural information. The last example is in [41], where the authors study what information about the node can be learned from its neighbors. They learn the attributes of the nodes by reconstructing a bipartite feature graph, whereas we are interested in reconstructing connections between the entities.

1.3 Contributions

As we discussed above this thesis addresses network reconstruction in general and consists of the two major chapters that address the network-alignment and the network-assembly problems, respectively. For both we group contributions into three parts: tractable models for problem analysis, analysis of the theoretical feasibility of network reconstruction and practical algorithms for network reconstruction.

We highlight the key results in the Table 1.1 where we describe information-theoretic and algorithmic contributions and consider two regimes with or without side information. The four major results are: feasibility of network alignment, feasibility of network assembly for random graph models and algorithms for network alignment and network assembly where side information is present. We describe each contribution in the detail below.

	Information-Theoretic	Algorithmic
no side information	• feasibility of network alignment [61] • feasibility of network assembly [116]	
with side information		• percolation graph matching from a seed-set [117] • labeled graph assembly

Table 1.1 – Summary of contributions to the network reconstruction problem

1.3.1 Network Alignment

$BiG(G;t,s)$ **Sampling Model**

For the first contribution, we introduce a model that generates two observations of the master graph. The model has tunable parameters to control the amount of noise. This model is an extension of the graph sampling model from [88]. The novelty is that our model takes into account partial node overlap, reflecting scenarios when two network node sets are not exactly the same, but correlated. Without loss of generality we can assume that all nodes are identified by a number in $[1 \dots n]$ where $n = |V|$ is the size of the node-set of the master graph.

Definition 1.6 ($BiG(G;t,s)$ sampling model)**.** *Let $G(V,E)$ be a master graph and let $G_1(V_1,E_1)$ and $G_2(V_2,E_2)$ be two samples of G obtained as follows: Each node $i \in V$ is sampled with probability t independently to both V_1 and V_2. After that an edge e is sampled in E_1 and E_2 with probability s if both its endpoints are sampled in V_1 and V_2, respectively.*

We show two graphs generated from the $BiG(G;t,s)$ sampling model at Figure 1.3.

For the special case $t = 1$ we call it the $BiG(G;s)$ sampling model, while it is equivalent to the model introduced in [88]. For the analysis of feasibility of alignment, we use an Erdős-Rényi random graph $G(n,p)$ as a master graph, where every edge exists with identical probability $p(n)$, independently of all the other edges. The $G(n,p)$ model has been widely used in the study of complex and social networks [42, 54, 86], and it is a plausible candidate for the study of the network reconstruction problem. This parsimonious model is a poor approximation of most real networks, that have salient properties not shared with random graphs (skewed degree distribution, 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 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. If the master graph G is an Erdős-Rényi graph, we call the sampling models $BiG(n,p;t,s)$ and $BiG(n,p;s)$, respectively.

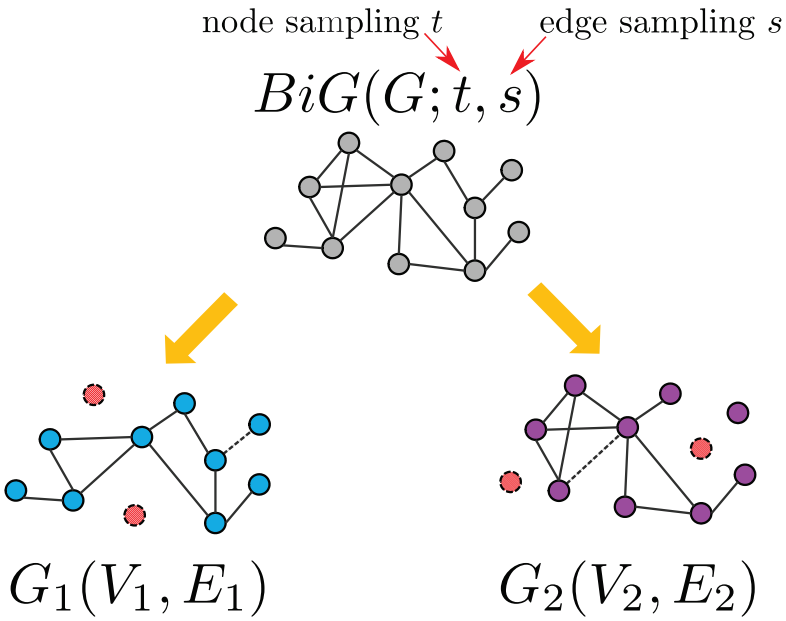

Figure 1.3 – The $BiG(G;t,s)$ random bigraph sampling model. The two graphs $G_1(V_1,E_1)$ and $G_2(V_2,E_2)$ are sampled from the generator graph $G(V,E)$ through node sampling (with probability t)and edge sampling (with probability s) processes.

Feasibility of Alignment Under Partial Node-Overlap

For the next contribution, in Section 2.1, we show the feasibility of alignment for large networks with different node sets. We explore the fundamental limits for de-anonymization, regardless of the specific algorithm employed (or given an adversary with infinite computational power who is able to check each alignment solution) and establish the relationship between network parameters and the feasibility of network alignment with no side information. We introduce a metric measuring the quality of a matching π, named Δ_π, basically a weighted count of "mismatched" edges with a weighting parameter α (see formal definition in Section 2.1), and we show that this metric is minimized by a correct matching π_0. We state the following theorem:

Theorem 2.3. *In the $BiG(n,p;t,s)$ bigraph model with $\frac{\log n}{ns^3t^2} \ll p \ll 1$[3], there exists a value of α such that with high probability*

$$\pi_0 = \operatorname*{argmin}_{\pi} \Delta_\pi.$$

where α is a weighting parameter.

[3]We use $f \ll g$ and $f \gg g$ meaning $f = o(g)$ and $f = \omega(g)$, respectively.

Percolation Graph Matching Algorithm

For a third contribution in the network alignment chapter, in Section 2.2, we propose a network alignment algorithm that iteratively builds a matching from a side information, that is a set of prealigned nodes $S = \{(i, j) : i \in V_1, j \in V_2\}$ called *seed-set*. We call it *Percolation Graph Matching* (PGM) as it relies on a threshold rule reminiscent of bootstrap percolation models [4]. Simply put, it starts from a seed-set S as an initial set of aligned pairs, and a pair (i, j) is added to the matching if there are at least r aligned pairs that are neighbors of (i, j)[4]. It repeats this process until there are no more pairs to add. We describe a basic version of the PGM and a deferred version for improved performance in Section 2.2.

Percolation Graph Matching Algorithm; Guarantees Under the $BiG(n; p, s)$ Model

For the fourth contribution, we analyze the performance of PGM under the $BiG(n; p, s)$ model. Consider two graphs G_1 and G_2, generated from the $BiG(n; p, s)$ sampling model and let $r \geq 4$. Define

$$a_c = \Big(1 - \frac{1}{r}\Big)\Big(\frac{(r-1)!}{n(ps^2)^r}\Big)^{\frac{1}{r-1}} \tag{1.1}$$

and note that ps^2 is the probability of an edge being sampled in both G_1 and G_2 or, equivalently, the probability of an edge being contained in the intersection of the edge sets $E_1 \cap E_2$. Here we show that a_c is the critical value of the initial size of the seed set. This means that for an initial number of seeds a_0 lower than a_c, the PGM algorithm stops with the final matching size a^* at most $2a_0$; and for a_0 larger than a_c, the algorithm propagates to most of the graph.

Theorem 2.15. *[Subcritical regime] Fix $\varepsilon > 0$. For $n^{-1} \ll ps^2 \ll s^2 n^{-\frac{3}{r}-\varepsilon}/\log n$, if $a_0/a_c \to \alpha < 1$, the PGM algorithm stops with $a^* \leq \frac{r}{r-1} a_c$ w.h.p.*[5] *In particular $a^* = (\phi(\alpha) + o(1))\frac{r}{r-1} a_c \leq \frac{r}{r-1} a_0$, where $\phi(\alpha)$ is the unique root in $[0, 1]$ of $r\phi(\alpha)) - \phi(\alpha)^r = (r-1)\alpha$.*

This means that in the subcritical regime, the final map is only slightly larger than the seed set, because the mapping process does not percolate.

Theorem 2.16. *[Supercritical regime] Fix $\varepsilon > 0$. For $n^{-1} \ll ps^2 \ll s^2 n^{-\frac{3}{r}-\varepsilon}/\log n$, if $a_0/a_c \geq \alpha > 1$ the algorithm propagates, and the size of the final mapping is $a^* = n - o(n)$ w.h.p.*

In summary, there is a sharp phase transition at $a_0 = a_c$ which separates almost-certain failure from almost-certain success of the percolation graph matching process.

[4] More precisely, two pairs (i, j) and (i', j') are neighbors iff $(i, i') \in E_1$ and $(j, j') \in E_2$.
[5] With high probability, i.e., with probability that tends to 1 as $n \to \infty$.

Evaluation and Performance Optimization

For the last contribution of the chapter, we evaluate the algorithm over both random graphs and real social network data, and we confirm the presence of the phase transition in the seed set.

We also describe the optimizations of some steps of the PGM. We introduce two optimizations: The first uses efficient data structures that enables the time complexity to be optimized and the second one modifies the main steps of PGM to make it suitable for a parallel map-reduce implementation. With these optimizations, we are able to push the sizes of the considered graphs to millions of nodes.

1.3.2 Network Assembly

$G(n,p;q)$ Graph Generator Model

For the first contribution of the network assembly chapter, we introduce a new random-graph model of independent interest, called $G(n,p;q)$: it accounts for such real-network property as high clustering and possesses randomness of network structure. In many real networks, neighborhoods of nodes are highly connected (i.e., have high clustering coefficient[6]). For example, in social networks, friends of any given person are likely to know each other. This behavior is called triadic closure [100]. We address the question of how a graph's clustering coefficient improves the feasibility of assembly.

There are several random graphs models that generate networks with high clustering: Watts and Strogatz [113] proposed a graph-generation algorithm that connects neighbors of a cycle. While the generated graphs are highly clustered the nodes-neighborhoods are similar and the degree distribution is homogeneous; it is unlike real world graphs and make these unfeasible candidate to network assembly problem.

The $G(n,p;q)$ model is defined via an intermediate Erdős-Rényi random graph $G_p(V_p,E_p) \sim G(n,p)$. The graph $G(V,E) \sim G(n,p;q)$ contains a random subset of all the possible closures of connected triples in G_p. More precisely, for each $u,v,w \in V$, if $(u,v) \in E_p$ and $(v,w) \in E_p$ we add (u,w) to E with probability q. Our goal is to obtain a model that is mathematically tractable (akin to the Erdős-Rényi model [42]), but possesses a higher clustering coefficient.

Network Assembly From Egonets

As the second contribution, in Section 3.1, we formulate a specific variation of the network assembly problem, where each patch is created by extracting the egonet around each vertex in the master graph. The *egonet*, or 1-egonet of a vertex i in a graph G, denoted H_i, is the

[6] The clustering coefficient of a node u is the density of the subgraph induced by its neighbors; assumed to be 0 if u is a singleton.

induced subgraph [7] generated by i and its neighbors in G — we say that i is the *center* of this egonet. We will further assume that, for each egonet in the patch collection, the identity of i is either kept intact or somehow inferable, but all other identities are removed.

We modify a definition 1.4 for this specific form of patches:

Definition 1.7 (Egonet Collection Extraction)**.** *Let G be a graph with $V(G) = [n]$ for some $n \in \mathbb{N}$, and edge set $E(G)$.*

- *An* unlabeled egonet collection *of G is a set of graphs $\mathscr{P} = \{G_i = f_i(H_i)\}_{i\in[n]}$, where a patching function $f_i : V(H_i) \to [|V(G_i)|]$ is a bijection such that $f_i(i) = 1$.*
- *An* unlabeled noisy egonet collection *of G is a set of graphs $\mathscr{P} = \{G_i = f_i(H_i^*)\}_{i\in[n]}$, where H_i^* are obtained from H_i by removing each edge with probability s independently and a patching function $f_i : V(H_i^*) \to [|V(G_i^*)|]$ is as defined above.*
- *Patching functions $\{f_i\}$ are called* anonymization functions *and the relabeled version of H_i denoted by G_i is called an* anonymized *egonet.*

Note that f_i relabels every vertex in H_i arbitrarily, except for i that is forcefully assigned the label 1. This means that, as long as the indices of each graph in the collection are known, the identities of the respective centers are also known.

Definition 1.8 (Egonet collection assembly)**.** *Let $\mathscr{P} = \{G_i\}_{i\in[n]}$ be a collection of graphs, such that $V(G_i) = [n_i]$ for some $n_i \in \mathbb{N}$. An* assembly *of $\mathscr{P}$ is a pair $(\hat{G}, \{a_i\}_{i\in[n]})$, where $\hat{G}$ is a graph (called* assembled graph*) with $V(\hat{G}) = [n]$, and each $a_i : [n_i] \to [n]$ is an injective function such that $a_i(1) = i$.*

An assembly determines not only which graph $\hat{G}$ is ultimately obtained, but also how each vertex in each egonet of our collection is mapped to $\hat{G}$. This is enough for us to formally state the *egonet assembly* problem:

- *Input:* an unlabeled egonet collection $\mathscr{P} = \{G_i\}_{i\in[n]}$;
- *Output:* an assembly $(\hat{G}, \{a_i\}_{i\in[n]})$ of $\mathscr{P}$.

Feasibility of Network Assembly in a Noiseless Case

For a third contribution, we prove the feasibility of a network assembly result under the aforementioned $G(n, p; q)$ model. We find that, even from relatively small patches (1-hop egonets) it is still possible to reconstruct a whole network.

[7] An induced subgraph is a subset of the vertices together with any edges whose endpoints are both in this subset.

The key observation is that, for a $G(n,p;q)$ random graph, any two edges have non-isomorphic subgraphs of common neighbors, where a subgraph of common neighbors of an edge (u,v) is a subgraph induced by nodes adjacent to u and v simultaneously . Therefore this feature acts as a fingerprint for all edges in a graph and enables us to identify these edges across different egonets.

Theorem 3.3. *Let G be a $G(n,p;q)$ random graph, with $(np)^5 p \to 0$, fixed q and $npq^2 = 12\log n + \omega(1)$, and let $\mathscr{P} = \{G_i\}_i \in [n]$ be an unlabeled egonet collection extracted from G. There exists an assembly algorithm that builds $\hat{G}$ from the input $\mathscr{P}$ and $V(\hat{G}) = V(G)$ and $E(\hat{G}) = E(G)$.*

To show feasibility, we propose a practical algorithm of network assembly. The algorithm assumes that the patches are in the form of perfect (noiseless) egonets, and it uses unique edge fingerprints to reconstruct the original graph.

Feasibility of Network Assembly in Noisy Case

As a fourth contribution of the chapter, we consider a more realistic scenario where we deal with imperfect patches. For instance, the observations of a user's circle in social networks can be noisy. In contrast with the noiseless case, perfect (no edge mismatch) assembly can no longer be expected. Rather, we expect that in low-noise scenarios, the correct assembly has a small number of edge mismatches, due to the correlation induced in the patch collection by the true graph. Therefore, we intuitively expect the correct assembly to have minimum edge inconsistency, i.e., edge mismatch.

In order to find the conditions where the hypothesis is true, we prove a result analogous to Theorem 3.3.

Theorem 3.9. *Let G be a $G(n,p;q)$ random graph, with $(np)^5 p \to 0$, fixed q and $npq^2 = \frac{32\log n + 16\log(npq^2) + \omega(1)}{s^3}$, and let $\mathscr{P} = \{G_i\}_{i\in[n]}$ be an unlabeled, noisy egonet collection extracted from G. There exists an assembly algorithm that builds $\hat{G}$ from the input $\mathscr{P}$ and $V(\hat{G}) = V(G)$ and $E(\hat{G}) = E(G)$.*

To show this, we modify the algorithm from a noiseless case to look for an edge with "closest" fingerprint rather than an edge with identical fingerprint.

General Algorithm of Network Assembly

As the last contribution in the chapter, in the Section 3.2, we consider the most realistic scenario where patches are general subgraphs with highly ambiguous labels. We assume that the master graph is labeled with a small label set and that this labeling is preserved through a patch-generation process or, in other words, the images of nodes in the patches have the

same labels as in the master graph. For example, consider a neural network assembly problem, where labels are types of neurons. The existing recording techniques enable us to observe only a small fraction of large networks simultaneously, hence it is very difficult to estimate a network, given the noise specific to the problem [108]. Another example refers to the assembly of a social network, from its multiple observations where labels are the first names. The labels are highly ambiguous because many persons can have the same name. In general, the observations are not required to be ego-centered (for example it can be some groups or communities) and some observations can be more incomplete than another.

We propose a graph assembly algorithm that merges patches pairwise until there is only one left and that is an estimator of the master graph. The algorithm selects a pair of patches based on the frequencies of common, small, labeled subgraphs that we call “seed-subgraphs” by analogy with a seed-set in the input of the PGM algorithm. These subgraphs serve as markers that enable us to stitch patches together by aligning them with the PGM algorithm.

2 Network Alignment

We recall an example of two on-line social networks where some individuals have accounts in both. Aligning users of the two networks provides information that is used for targeted marketing and for information-diffusion research; this helps to enrich user information and to reconstruct a more realistic network of a person's contacts. However, names are often ambiguous or absent, hence we have to rely on some structural information to align these networks.

As we mentioned, Narayanan and Shmatikov in [85] succeed in matching a large-scale anonymized social network to a second social network that serves as side information. This proves that network alignment is possible for a large scale network. In another work [88], a random graph model provides insight on the fundamental feasibility of graph alignment under full node-overlap for an adversary with unlimited computational power. Two major questions remain unanswered: The first is about the theoretical feasibility of an alignment of two networks under partial node-overlap with no additional information. The second question addresses the existence and analysis of practical algorithms.

In the Section 2.1, we show the feasibility of network alignment for two networks sampled from the $BiG(G;t,s)$ sampling model. In the Section 2.2, we propose and analyze a percolation-based graph-matching algorithm.

2.1 Alignment of Networks Under Partial Node Overlap

For input of the network alignment problem, we have $G_1(V_1,E_1)$ and $G_2(V_2,E_2)$ that can be viewed as observations or realizations of a master graph G. For example, G can be a real social network, G_1 be personal connections observed via a Facebook graph, and G_2 can be work connections observed via e-mail exchanges in the organization.

Recall that we are interested in finding a matching that is a partial bijection $\pi : V_1 \to V_2$ between (a subset of) the vertex sets of the two graphs. We note that, due to the bijection restriction, the definition is symmetric and V_1 and V_2 are interchangeable. We denote by $V_1(\pi)$ a do-

main of π and by $V_2(\pi)$ a range of π. For a pair of nodes $e = (i, j)$, we define $\pi(e) = \big(\pi(i), \pi(j)\big)$. We say $e = (i, j) \in E_1(\pi)$ if $i, j \in V_1(\pi)$ and $e \in E_1$, same for $E_2(\pi)$. We assume that, without loss of generality, $V_{1,2} \subset [n] = \{1, \ldots, n\}$ and denote $n_1 = |V_1|$, $n_2 = |V_2|$ and $n_0 = |V_0|$ (recall $V_0 = V_1 \cap V_2$).

If the two graphs are sampled from the $BiG(G; t, s)$ model then the matching π_0 can be written as a set of pairs of vertices sampled from the same nodes of G. Recall that our goal is to find π_0 given G_1 and G_2. This means finding corresponding domain and range $V_0 \subseteq V_1$ and $V_0 \subseteq V_2$ and find a correct matching between found nodes.

To measure the quality of the matching without knowledge of the ground truth, we now define a cost function that quantifies the structural mismatch between the two graphs under a given partial matching π.

Definition 2.1 (Cost Function)**.** *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 tradeoff 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 tradeoff: 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, the matching found by minimizing Δ_π is the correct partial matching π_0.

Example of Matching

We give an example of the matching and define few more variables:

Definition 2.2 (Matching Characteristics)**.** *For a matching π we define (i) $|\pi|$ as the size of matching π, (ii) l as the number of correctly matched nodes of the form $\pi(i) = i$ and, (iii) $k =$*

$|\pi| - l$ *as the number of wrongly matched nodes. Let* Π_k^l *represent a class of matchings of size* $|\pi| = l + k \leq \min\{n_1, n_2\}$ *with* l *correctly matched nodes. Note that the sets* Π_k^l *partition the set* Π *of all partial matchings.*

For example, Figure 2.1 shows the identity matching $\pi_0 \in \Pi_0^7$ and the matching $\pi \in \Pi_6^2$ from V_1 to V_2.

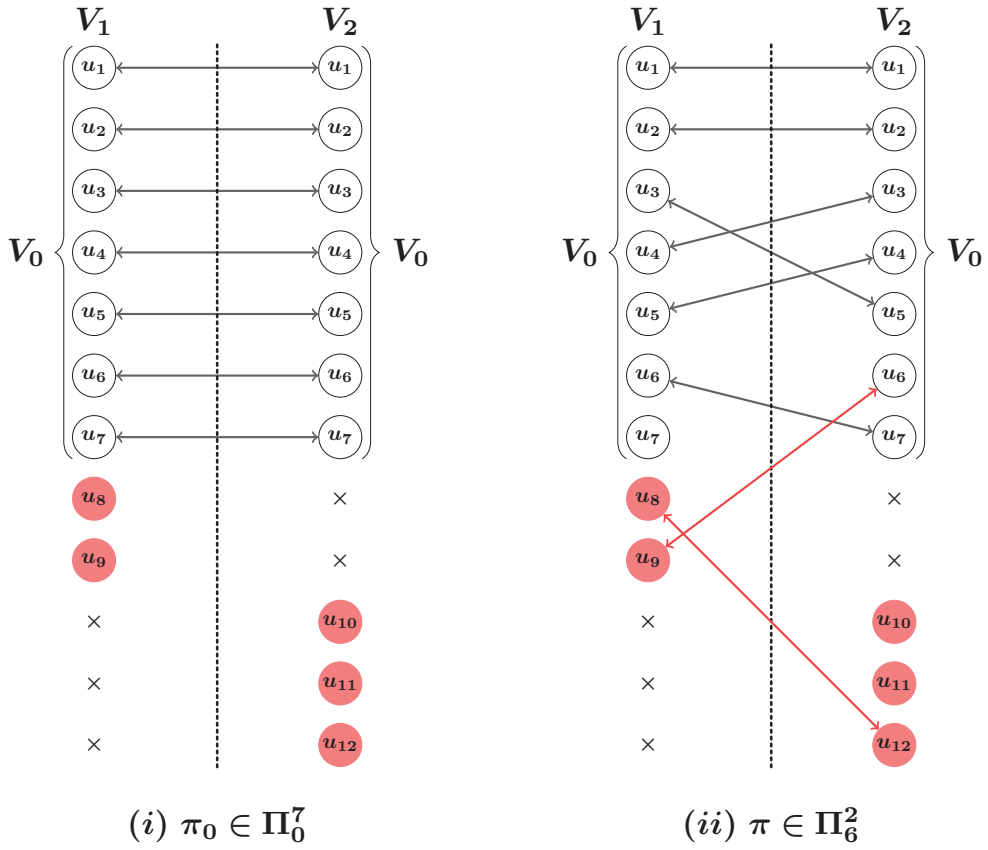

Figure 2.1 – Examples of two matchings: (i) The true matching $\pi_0 \in \Pi_0^7 = \{[u_1, u_1], \ldots, [u_7, u_7]\}$, and (ii) the matching $\pi \in \Pi_6^2$. White nodes are sampled in both graphs, while red nodes are sampled in only one but not the other.

We introduce few more contracted definitions for further simplicity: For a node u, we say $\pi(u)$ is *null* (denoted by $\pi(u) = \emptyset$) if either u is not sampled ($u \notin V_1$) 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 $\pi^{-1}(v) = \emptyset$. For example, $\pi(u_7) = \emptyset$ and $\pi^{-1}(u_{10}) = \emptyset$ at the Figure 2.1 (ii).

We now state the main result of the section.

Theorem 2.3. *In the* $BiG(n, p; t, s)$ *model with* $\frac{6144 \log n + \omega(1)}{n s^3 t^2} = p \ll 1$*, there exists a value of* α

such that with high probability

$$\pi_0 = \operatorname*{argmin}_{k,l,\pi\in\Pi_k^l} \Delta_\pi. \tag{2.2}$$

Before proving Theorem 2.3, we provide some context for the result.

Expressed in terms of the expected degree $npst$ of the two observable graphs $G_{1,2}$, the threshold is $\log(n)/s^2t$ for perfect matchability. The dependence on n is tight. To see this, consider the intersection graph $G_0 = G(V_0, E_1 \cap E_2)$. Its expected degree is nps^2t^2.[1] If this is asymptotically less than $\log nt^2$, then G_0 has symmetries w.h.p. (which in fact stem from isolated vertices [24]). 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 to estimate π_0 over the common nodes, because of 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 as well, while there is a gap of a factor of s between the achievability result in Theorem 2.3 and the trivial lower bound based on G_0.

With $t = 1$, we can recover the achievability result of Pedarsani and Grossglauser [88] up to a constant. Note that this is not trivial, as their problem formulation minimizes a cost function[2] over the set $\{\Pi_k^l : k + l = n\}$, while here we minimize over the larger set $\{\Pi_k^l : k + l \le n\}$.

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 tradeoff between penalizing mismatched mapped edges and unmapped edges needs to be controlled more finely through an appropriate choice of α that depends on p and s.

2.1.1 Proof of the Theorem 2.3

We provide a brief sketch followed by the detailed proof. Let S be the number of matchings $\pi \in \Pi$ such that $\Delta_\pi - \Delta_{\pi_0} \le 0$. Following the Markov inequality, as S is a non-negative integer-valued random variable, we have $\mathbb{P}[S \ge 1] \le \mathbb{E}[S]$. We will prove that, under the conditions of Theorem 2.3,

$$\mathbb{P}[S \ge 1] \le \mathbb{E}[S] = \sum_{\pi\in\Pi} \mathbb{P}(\Delta_\pi - \Delta_{\pi_0} \le 0) \to 0. \tag{2.3}$$

[1] To be precise, $(n-1)ps^2t^2$; we sometimes omit lower-order terms for readability.
[2] Identical to ours with $\alpha = 0$.

The main complication of the proof stems from the fact that the random variables Δ_π and Δ_{π_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 Δ_π and Δ_{π_0} can be written as sums of Bernoulli random variables. The main challenge in the proof is to decompose the difference $\Delta_\pi - \Delta_{\pi_0}$ 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. The elements of some parts of the edge partition contribute equally to Δ_π and Δ_{π_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 i.i.d. Bernoulli's random variables to apply standard concentration arguments to Δ_π and Δ_{π_0} individually, and then to stochastically bound their difference.

Proof. [Theorem 2.3] We consider the contribution of edges (or potential edges) to the terms Δ_π and Δ_{π_0} as a random variable in the $BiG(n,p;t,s)$ 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 $e=(u,v)$ that, based on the realization of the $BiG(n,p;t,s)$ random model, might 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 Φ_π counts the number of edges in both graphs that are matched to a nonexistent edge in the other graph. More precisely, the contribution of a pair $e \in V_1^2(\pi)$ and its image $\pi(e) \in V_2^2(\pi)$ to Φ_π is $\phi(e) = |\mathbb{1}_{\{e\in E_1(\pi)\}} - \mathbb{1}_{\{\pi(e)\in E_2(\pi)\}}|$. Note that the pairs e and $\pi(e)$ contribute to Φ_π 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 $\psi_1(e) = \mathbb{1}_{\{e\in E_1\setminus E_1(\pi)\}}$; it represents the contribution of the pair e to Ψ_π^1. This indicator term is equal to 1 if the edge between the 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 Δ_π as

$$\Delta_\pi = \sum_{e\in V_1^2(\pi)} \phi(e) + \alpha \left[\sum_{e\in V_1^2\setminus V_1^2(\pi)} \psi_1(e) + \sum_{e\in V_2^2\setminus V_2^2(\pi)} \psi_2(e) \right]. \tag{2.4}$$

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

2.1.2 Node Partition

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

(i) $\checkmark(\pi)$ is the set of nodes that are matched correctly by π, i.e.,

$$\checkmark(\pi) = \{u \in V_1 \cup V_2 | \pi(u) = u\}.$$

(ii) $\rightarrow(\pi)$ is the set of nodes that are matched in the graph G_1, but π^{-1} is null for them, i.e.,

$$\rightarrow(\pi) = \{u \in V_1 \cup V_2 | \pi(u) \neq \emptyset, \pi^{-1}(u) = \emptyset\}.$$

(iii) $\leftarrow(\pi)$ is the set of nodes that are matched in the graph G_2, and π is null for them, i.e.,

$$\leftarrow(\pi) = \{u \in V_1 \cup V_2 | \pi(u) = \emptyset, \pi^{-1}(u) \neq \emptyset\}.$$

(iv) $\leftrightarrow(\pi)$ is the set of nodes that are matched in both graphs $G_{1,2}$, but wrongly, i.e.,

$$\leftrightarrow(\pi) = \{u \in V_1 \cup V_2 | \pi(u) \neq \{u, \emptyset\}, \pi^{-1}(u) \neq \emptyset\}.$$

(v) $\times(\pi)$ 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 defined in Table 2.1. For an example of these pairwise intersections, see Table 2.2.

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

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.

2.1.3 Edge Partition

We now partition the set of pairs based on the classes of nodes which are defined in Table 2.1. A pair e contributes equally to Δ_π and Δ_{π_0} if it is matched in the same way by π and π_0 (i.e., $\pi_0(e) = \pi(e)$), or if it is null in both. The following sets are those pairs that contribute equally to Δ_π and Δ_{π_0}, and consequently, their contributions will cancel out in the difference $\Delta_\pi - \Delta_{\pi_0}$:

π_0 \ π	✓	↔	→	←	×
✓	u_1, u_2	u_3, u_4, u_5, u_6	$\emptyset$	u_7	$\emptyset$
×	$\emptyset$	$\emptyset$	u_8, u_9	u_{12}	u_{10}, u_{11}

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

1. Pairs between the nodes in the set $\mathscr{C}$. These pairs are present in both graphs and their endpoints are matched correctly by both π and π_0. For example, in Fig. 2.1, the pair (u_1, u_2) is matched to the same pair by matchings π_0 and π.

2. Pairs in G_1 between $\mathscr{U} \cap V_1$ (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 $(\mathscr{U} \cap V_2) \times V_2$ in the graph G_2. Note that these pairs are present in only one of the graphs. As an example, in Fig. 2.1, the pairs (u_{10}, u_{11}), (u_{10}, u_{12}) and (u_{10}, u_2) in the graph G_2 are matched neither under π nor under π_0.

3. Pairs e between $\mathscr{Q}$ and $\mathscr{S} \cup \mathscr{R}$ 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 $\mathscr{X}$ and $\mathscr{S} \cup \mathscr{L}$ in the graph G_2 contribute a term $\psi_2(e)$ under both matchings π and π_0. Note that these pairs are present only in one of the graphs. In Fig. 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 these pairs to Δ_π and Δ_{π_0}, respectively. By definition $Z_\pi = Z_{\pi_0}$. Call $\mathscr{E}$ the set of all the remaining pairs that are matched differently under π and π_0. Note that $\mathscr{E}$ depends on both matchings π and π_0. As for each instance of the $BiG(n, p; t, s)$ model the matching π_0 is fixed, for simplicity of notation we drop the dependence on π_0 and 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 $\mathscr{E}$ under the matchings π and π_0, respectively.

To wrap up, 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 the contributions of pairs from the set $\mathscr{E}$ to different indicator terms in X_π and Y_π, we partition this set into the following subclasses:

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 is 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:

- $\mathscr{E}_{\emptyset,M*} = \{(i,j) \in (\mathscr{Q} \times V_1(\pi))\}$ is the set of pairs that contribute $\psi_1(e)$ to $\Psi^1_{\pi_0}$ and $\phi(e)$ to Φ_π.
- $\mathscr{E}_{\emptyset,*M} = \{(i,j) \in (\mathscr{X} \times V_2(\pi))\}$ is the set of pairs that contribute $\psi_2(e)$ to $\Psi^2_{\pi_0}$ and $\phi\left(\pi^{-1}(e)\right)$ to Φ_π.

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

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

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

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

3. $\mathscr{E}_{M,MM} = \{(i,j) \in \mathscr{W} \times (\mathscr{C} \cup \mathscr{W})\}$ is the set of pairs that are present and matched, but wrongly, by π in both graphs $G_{1,2}$. These pairs are matched differently by π and π_0. The pairs in the set $\mathscr{E}_{M,MM}$ contribute to a $\phi(e)$ in Φ_{π_0}, and contribute to the terms $\phi(e)$ and $\phi\left(\pi^{-1}(e)\right)$ in Φ_π. For example, in Fig. 2.1, the pairs (u_1, u_3) and (u_4, u_5) which are matched differently by π_0 and π belong to the set $\mathscr{E}_{M,MM}$.

   One observation is that there are small subset of pairs that are present and matched but do not conribute to $\Delta_\pi - \Delta_{\pi_0}$. Indeed, transpositions[3] 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 wrongly matched couples is k. To be precise, we do not consider these pairs in the set $\mathscr{E}_{M,MM}$.

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

Indicator Terms and Expected Values

In Lemma 2.4, the two terms X_π and Y_π are expressed as sums of indicator terms (Bernoulli random variables) over the pairs in $\mathscr{E}$.

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

Lemma 2.4. *For X_π we have:*

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

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:

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

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

Proof. First, note that $\mathscr{E}_{\emptyset,M*}\cup\mathscr{E}_{M,M\emptyset}\cup\mathscr{E}_{M,MM} = \mathscr{E}\cap V_1^2(\pi)$ is the set of all matched pairs from G_1 which are in the set $\mathscr{E}$. Remember that by (2.5) the term X_π is the sum of indicators in Δ_π over pairs in the set $\mathscr{E}$. Thus, we get the first term in the right hand side of (2.6). Each pair e (same is true for $\pi(e)$) exists in each of the graphs $G_{1,2}$ with probability ps; therefore $\phi(e) = Be\big(2ps(1-ps)\big)$. Second, we compute the number of terms $\psi_{1,2}(e)$ that contribute to X_π. These are (i) the pairs of type $\mathscr{E}_{M,M\emptyset}\cup\mathscr{E}_{M,\emptyset M}$ that contribute to either Ψ_π^1 or Ψ_π^2, and (ii) the pairs of type $\mathscr{E}_{M,\emptyset\emptyset}$ 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 $\mathscr{E}$ to Δ_{π_0}. For each pair e matched by π_0 and π, $e\in\mathscr{E}_{M,M\emptyset}\cup\mathscr{E}_{M,\emptyset M}\cup\mathscr{E}_{M,\emptyset\emptyset}\cup\mathscr{E}_{M,MM}$ 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 the second part (2.7). □

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

Corollary 2.5. *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 the 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 the $\{\phi(e)\}$ terms is equal to the number of pairs from graph G_2 which contribute

to these terms, i.e., $|\mathscr{E}_{\phi,M*} \cup \mathscr{E}_{M,M\phi} \cup \mathscr{E}_{M,MM}| = |\mathscr{E}_{\phi,*M} \cup \mathscr{E}_{M,\phi M} \cup \mathscr{E}_{M,MM}|$. Remember that $|\mathscr{E}_{\phi,M*} \cup \mathscr{E}_{\phi,*M}| = m_1$ and $|\mathscr{E}_{M,M\phi} \cup \mathscr{E}_{M,\phi M}| = m_2$. To sum up, number of $\{\phi(e)\}$ terms which contribute to X_π (defined precisely in Lemma 2.4) is $m_3 + \frac{m_1+m_{2,1}}{2}$. The rest comes directly from the definitions of m_1, m_2 and m_3. □

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

Lemma 2.6. *If* $1 - ps > \alpha > 1 - s$, *then* $\mathbb{E}[X_\pi] > \mathbb{E}[Y_\pi]$.

Proof. From Corollary 2.5, 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 holds. □

2.1.4 Correlation Structure

Lemma 2.6 guarantees that for any $\pi \neq \pi_0$, $\mathbb{E}[\Delta_\pi] > \mathbb{E}[\Delta_{\pi_0}]$. In the following, we demonstrate that X_π and Y_π, as 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 the pairs $e \in G_1$ 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 $\phi\big(\pi^2(e)\big)$ are correlated, and so on. Thus, terms Φ_π and Ψ_π are the sums of correlated Bernoulli random variables.

To address these correlations, we first define *chains* and *cycles* of correlated pairs under the matching π. We call a sequence of different 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 $1 \leq i < q$, i.e., each pair in a chain is the image of the previous pair in that chain under the matching π. In Fig. 2.2b, 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 different pairs $(e_1, \cdots, e_i, \cdots, e_q)$ a *cycle* if (i) $\pi(e_i) = e_{i+1}$ for $1 \leq i < q$; 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 Fig. 2.3a.

Following the discussion above, we state Lemma 2.7: In Part 1 of the lemma we (i) partition all the pairs of $\mathscr{E}$ into chains and cycles; and (ii) demonstrate contributions of these pairs to different indicator terms. In Part 2 we characterize correlations between the terms in the induced sequence of indicators.

Lemma 2.7. ***Part 1**:*

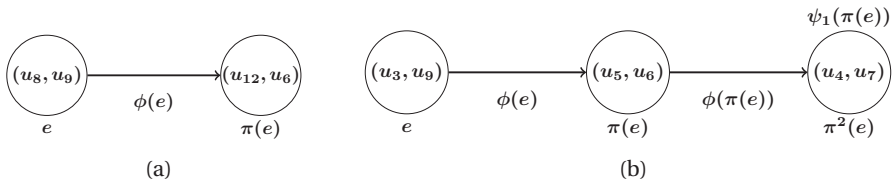

Figure 2.2 – (a) Example of a chain with length one from the matching π from Fig. 2.1. (b) Example of a chain with length three from the matching π from Fig. 2.1. The term $\psi_1(\pi(e))$ corresponds to the contribution of the pair (u_2, u_6) in the graph G_1. In this chain, the term $\phi(\pi(e))$ is correlated with the two terms $\phi(e)$ and $\psi_1(\pi(e))$.

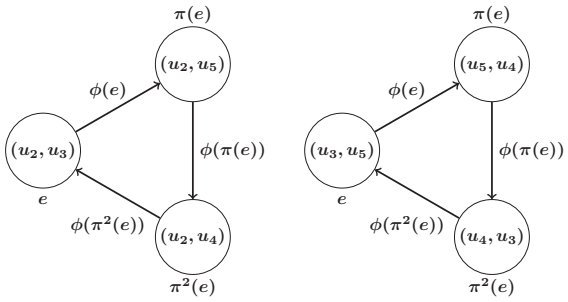

Figure 2.3 – Examples of two cycles from the matching π from Fig. 2.1. The pairs generate a cycle of dependent terms. In these cycles, the terms $\phi(e), \phi(\pi(e))$ and $\phi\left(\pi^2(e)\right)$ are correlated pairwise.

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 $(e_1, \cdots, e_i, \cdots e_q), 1 \leq i < q$, its 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 $(e_1, \cdots e_i, \cdots e_q), 1 \leq i < q$, its 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. *$e_1 \in \mathscr{E}_{M,\emptyset\emptyset}$ is a specific case where we have a chain of length one. The pair e_1 contributes to the induced sequence of indicator terms $\big(\psi_2(e_1), \psi_1(e_1)\big)$.*

***Part 2**:*

For sequences of induced indicator terms from partitions in Part 1, we have

- *All the induced indicators ϕ/ψ associated with different chains and cycles are mutually independent.*
- *For a chain, each indicator ϕ/ψ is correlated with at most the preceding and subsequent indicators in the induced sequence.*
- *For a cycle, each indicator ϕ/ψ is correlated with at most the preceding and subsequent indicators in the induced sequence, and $\phi(e_1)$ is correlated with $\phi(e_q)$.*

Proof. We prove that the set chains and cycles correctly partition the pairs in set $\mathscr{E}$, and we characterize the dependence structure of the indicators within this partition.

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

- if $\pi^c(e) \in \mathscr{E}_{\emptyset,*M}$ then $\pi^c(e)$ is matched and exists only in G_2. Therefore, this chain of pairs induces the sequence $\big(\phi(e), \cdots, \phi\big(\pi^{c-1}(e)\big)\big)$ of indicator terms. Fig. 2.2a is an example of such a chain under the matching π from Fig. 2.1.
- if $\pi^c(e) \in \mathscr{E}_{M,\emptyset M}$ then $\pi^c(e)$ exists in both graphs but is matched only in G_2. Therefore, this chain induces the sequence $\big(\phi(e), \cdots, \psi_1(\pi^c(e))\big)$ of indicator terms. Fig. 2.2b is an example of such a chain under the matching π from Fig. 2.1.

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

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

Now we formulate a cycle/chain partition process as follows:

- *Chain partition:* First, for each pair, we build a chain as described above; second, for each pair $e \in \mathscr{E}_{M,M\emptyset}$ we build another chain; third, for each pair of type $e \in \mathscr{E}_{M,\emptyset\emptyset}$ 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 $e \in \mathscr{E}_{\emptyset,*M}$ or $e \in \mathscr{E}_{M,\emptyset M}$, we can build backwards the same chain of pairs; starting from e and applying π^{-1} instead of π. Based on this observation, we compute that there are $m_1 + m_2$ pairs that start or end a chain.

- *Cycle partition:* The fourth step is to partition the remaining, unvisited pairs that all have type $\mathscr{E}_{M,MM}$ (note that they are sampled and matched by π in both graphs). For each unvisited pair e, the unvisited pair $\pi(e)$ also has type $\mathscr{E}_{M,MM}$ (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 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 $\pi^2(e) = e$.

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. Fig. 2.3 provides examples of cycles of pairs under the matching π from Fig. 2.1.

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 $\phi(e_i) \perp\!\!\!\perp \psi_1(e_j), \psi_2(e_j)$. Further, 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 simply 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.

□

2.1.5 Marking indicators

In Lemma 2.7, we defined induced sequences of indicators terms and characterized their correlation. We showed that each term $\phi(e)$ (or $\psi_{1,2}(e)$) is correlated with at most two of its neighbors (e.g., see Figs. 2.2 and 2.3). Now, we associate a mark 0 or 1 with all the induced $\phi(e)$ and $\psi_{1,2}(e)$ terms by alternating marks in such a way that almost all the indicators with the same mark are independent. This is not generally true for the terms at start and end of cycles with odd number of indicators: although they have the same mark, but they are not independent (we deal with these separately, see below). Another requirement is that for each

type of indicator, i.e., (i) indicators $\phi(e)$ and (ii) start/end indicators $\psi_{1,2}(e)$ at least a constant fraction of indicators should be marked with 0 and a constant fraction of them with 1.

Based on this marking 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.

For each sequence of indicators $\left(\phi(e_1),\cdots\phi(e_i),\cdots\phi(e_q)\right)$ which are induced by a cycle, we start with a pair $\phi(e_1)$ and mark it with $m\left(\phi(e_1)\right)=0$. Next, we mark $\phi(e_2)$ with 1, $\phi(e_3)$ with 0 and so on. We continue the next sequence with a new mark (if we ended with 1 then we start with 0 and vice versa) until there are no more cycles with unmarked indicators.

For sequences of indicators which are induced by chains, the marking is slightly more complicated. First, note that we can iteratively mark a sequence from the beginning or from the end. Second, we remind the reader that all the indicators induced by $e=e_1/e_q$ beginning/end of the chain are either $\phi(e)$ for $e\in\mathscr{E}_{\emptyset,M*}\cup\mathscr{E}_{\emptyset,*M}$ or $\psi(e)$ for $e\in\mathscr{E}_{M,M\emptyset}\cup\mathscr{E}_{M,\emptyset M}\cup\mathscr{E}_{M,\emptyset\emptyset}$. Now, let us mark all the sequences of indicators which are induced by chains while doing the following four steps iteratively:

1. Take a sequence that starts/ends with an indicator $\phi(e)$ and mark $\phi(e)$ with $m\left(\phi(e)\right)=0$ next we mark $\phi(\pi(e))$ (or $\phi\left(\pi^{-1}(e)\right)$) with 1 , $\phi\left(\pi^2(e)\right)$ with 0 and so on.
2. Take a sequence that starts/ends with an indicator $\psi(e)$ and mark $\psi(e)$ with $m\left(\psi(e)\right)=$ 0 next we mark $\phi(\pi(e))$ (or $\phi\left(\pi^{-1}(e)\right)$) with 1, $\phi\left(\pi^2(e)\right)$ with 0 and so on.
3. Take a sequence that starts/ends with an indicator $\phi(e)$ and mark $\phi(e)$ with $m\left(\phi(e)\right)=1$ next we mark $\phi(\pi(e))$ (or $\phi\left(\pi^{-1}(e)\right)$) with 0 , $\phi\left(\pi^2(e)\right)$ with 1 and so on.
4. Take a sequence that starts/ends with an indicator $\psi(e)$ and mark $\psi(e)$ with $m\left(\psi(e)\right)=$ 1 next we mark $\phi(\pi(e))$ (or $\phi\left(\pi^{-1}(e)\right)$) with 0, $\phi\left(\pi^2(e)\right)$ with 1 and so on.

If we do not have more sequences that starts/ends with an indicator of one of the types, we continue marking the remaining sequences alternating a start mark with 0 or 1.

Lemma 2.8. *The proposed strategy of marking the indicators* $\{\phi(e)\cup\psi_{1,2}(e)\}$ *with* 0/1 *marks guarantees that*

1. *at least* $\frac{1}{3}$ *indicators of pairs* $\{\mathscr{E}_{\emptyset,M*}\cup\mathscr{E}_{M,M\emptyset}\cup\mathscr{E}_{M,MM}\}$ *gets mark* 0 *and at least* $\frac{1}{3}$ *gets mark* 1.
2. *at least* $\frac{1}{6}$ *indicators* $\{\psi_1(e)\}$, $\{\psi_2(e)\}$ *of sets of pairs* $\{\mathscr{E}_{M,\emptyset M}\cup\mathscr{E}_{M,\emptyset\emptyset}\}$, $\{\mathscr{E}_{M,M\emptyset}\cup\mathscr{E}_{M,\emptyset\emptyset}\}$ *respectively gets mark* 0 *and at least* $\frac{1}{6}$ *gets mark* 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 the indicators* $\phi(e_1)$ *and* $\phi(e_2)$ *are independent. The same is true for* $\psi_{1,2}$ *indicators.*

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

Now we prove the first clause. Consider the indicators $\{\phi(e)\}$ of pairs $\{\mathscr{E}_{\emptyset,M*} \cup \mathscr{E}_{M,M\emptyset} \cup \mathscr{E}_{M,MM}\}$. For the indicators induced by cycles, we start marking with 0, and alternating 0 and 1. Thus approximately (depending if we stopped at 0 or 1) half of the pairs is marked with 0 and the rest is marked with 1. For the chains, at least $\frac{1}{6}$ start/end indicators of type ϕ is marked with 1 and the same for mark 0 (The argument here is the same as for the indicators of the pairs of type ψ.). For internal indicators, as we alternate the start counter at each iteration, at least $\frac{1}{3}$ of the indicators is marked with 0 and at least $\frac{1}{3}$ of the indicators is marked with 1.

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

For simplicity of notation, we write $m(e) = 0/1$ meaning $m\big(\phi(e)\big) = 0/1$ or $m\big(\psi(e)\big) = 0/1$.

Using this marking algorithm, we split X_π into two sums: $X_\pi = S_1 + S_2$, such that

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

and

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

Lemma 2.9. *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 2.8 and the linearity of expectation. □

2.1.6 Concentration

We define $\mu_1 = \mathbb{E}[X_\pi]$ and $\mu_2 = \mathbb{E}[Y_\pi]$ and we formulate concentration results for X_π and Y_π. (2.5).

Lemma 2.10.

$$\mathbb{P}[X_\pi < \frac{\mu_1+\mu_2}{2}] \leq 2\exp(-\frac{(\mu_1-\mu_2)^2}{96\mu_1})$$

$$\mathbb{P}[Y_\pi > \frac{\mu_1+\mu_2}{2}] \leq \exp(-\frac{(\mu_1-\mu_2)^2}{12\mu_1})$$

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

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

We prove that $\mathbb{P}[S_1 < (1-\varepsilon)\mathbb{E}[S_1])$ (the proof for S_2 is similar).

From the result of Lemma 2.8, 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 (i.e., a cycle where the beginning and the end indicators have the same mark). 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 the 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. Hence,

$$\begin{aligned}\mathbb{P}\left[S_1 < (1-\varepsilon)\mathbb{E}[S_1]\right] &= \mathbb{P}\left[\sum W_i < (1-\varepsilon)\mathbb{E}[W]\right] \\ &\leq \exp\left(-\frac{\mathbb{E}[W]\varepsilon^2}{2}\right) \text{ (by a Chernoff-Hoeffding bound A.1)} \\ &\leq \exp\left(-\frac{\mathbb{E}[S_1]\varepsilon^2}{4}\right) \leq \exp\left(-\frac{\mathbb{E}[X_\pi]\varepsilon^2}{24}\right) \text{ (by Lemma 2.9)}\end{aligned}$$

To sum up, we put $\varepsilon = \frac{\mu_1-\mu_2}{2\mu_1}$, note that $\frac{\mu_1+\mu_2}{2} = \mu_1 - \frac{\mu_1-\mu_2}{2} = \mu_1(1-\frac{\mu_1-\mu_2}{2\mu_1})$. For μ_2 we can write similarly $\frac{\mu_1+\mu_2}{2} = \mu_2(1+\frac{\mu_1-\mu_2}{2\mu_1})$, and obtain the bound for X_π.

The result for Y_π follows directly from a Chernoff bound because all the terms are independent. □

Next, we estimate $\mathbb{P}[X_\pi - Y_\pi \leq 0]$ based on the concentration results of X_π and Y_π.

$$\mathbb{P}[X_\pi - Y_\pi \leq 0] \leq \mathbb{P}\left[X_\pi < \frac{\mu_1+\mu_2}{2}\right] + \mathbb{P}\left[Y_\pi > \frac{\mu_1+\mu_2}{2}\right]. \tag{2.8}$$

We lower-bound $\frac{(\mu_1-\mu_2)^2}{\mu_1}$ to estimate (2.8) as follows. Define

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

From Corollary 2.5 we have $\mu_1 - \mu_2 \geq \alpha' ps(m_1+m_2+m_3) \geq \alpha' mps$.

Also, note that $\mu_1 \leq 2mps$ and $\mu_2 \leq 2mps$. Hence,

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

To sum up, we have

$$\mathbb{P}[X_\pi - Y_\pi \leq 0] \leq 3\exp\left(-\alpha'^2 \frac{mps}{192}\right). \tag{2.9}$$

Thus the expected number of matchings $\pi \neq \pi_0$ 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 (as the number of node pairs in the set $\mathscr{E}$) and an upper bound for $|\Pi_k^l|$.

Lemma 2.11. *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 matchings in the set Π_k^l. 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 matchings between these k chosen nodes. Therefore,

$$|\Pi_k^l| \leq \binom{n_0}{l}\binom{n_1-l}{k}\binom{n_2-l}{k}k! \leq n_0^{n_0-l}n_1^k n_2^k. \tag{2.10}$$

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, $m = |\mathscr{E}|$ is the number of pairs which 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 $\mathscr{E}$ includes all the pairs between any sampled node $v_1 \in V_0$ and any node $v_2 \in V_1(\pi) \cup V_2(\pi)$ which 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

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

□

Now, we find an upper bound for $\mathbb{E}[S]$ based on the above cases.

(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

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

(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

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

The geometric sum goes to 0 if the first term goes to 0. Thus given that $ps = \frac{1536}{\alpha'^2}\frac{\log n+\omega(1)}{n_0}$, we obtain $\mathbb{E}[S] \to 0$. We obtain $n_0 = nt^2(1+o(1))$ from a Chernoff bound and get $ps = \frac{1536}{\alpha'^2}\frac{\log n+\omega(1)}{nt^2}$.[4]

To conclude the proof of Theorem 2.3, we choose $\alpha = \frac{(1-ps)+(1-s)}{2} = 1 - \frac{s(1+p)}{2}$; then $\alpha' = \frac{s(1+p)}{2}$ and we derive the final bound $\frac{6144\log n+\omega(1)}{ns^3t^2} = p$.

□

Summary

We formulated conditions on the master-graph density p, and proved that within these conditions 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 $\omega\left(\frac{\log(n)}{s^2t}\right)$.

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. In the next section we discuss the question of efficient algorithm. The interesting result about the alignment was given at [36], it addresses a network alignment under full node-overlap, closing the gap between feasibility and a converse and improving the bound by a factor s. The question of extending this result to a partial node-overlap remains open and is a natural following step.

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

2.2 On the Performance of Percolation Graph Matching

In this section we propose an algorithm for a network alignment with a side information. Consider two networks G_1 and G_2 sampled from $BiG(n,p;s)$ model. The algorithm matches node pairs in the two graphs $G_{1,2}$ starting with a preselected seed-set and iteratively expands this set, by identifying additional pairs of nodes $i \in G_1$, $j \in G_2$ that can plausibly be matched. Whether (i,j) is a plausible match is computed from the positions of i and j, with respect to the known matched nodes. Our main theoretical contribution in this section is to identify conditions on the model parameters (n,p,s) and on the size of the seed set a_0 (a small set of initially pre-matched pairs of nodes) such that percolation graph matching (PGM) algorithm succeeds with high probability. For this, we rely on recent advances in the analysis of bootstrap percolation in the $G(n,p)$ random graph by Janson et al. [55]. We briefly summarize their model and key results here.

2.2.1 Bootstrap Percolation Theory

Janson et al. proposed a method for the analysis of infection spread in a network and established several results about the process. The described bootstrap percolation algorithm starts with an initially infected set and, at each time step, it takes exactly one seed and increases the infection mark counter of its neighbours.

Percolation theory is the study of the presence of large (or infinite) clusters in random environments, such as lattices with missing nodes or links, or random graphs. In bootstrap percolation, we study systems where a node is part of a cluster only if it has at least r neighbors that belong to the cluster. This more restrictive notion of inclusion can capture, for example, the spread of influence through a social network, where an individual is convinced of an idea only if she hears this idea from several acquaintances.

In a seminal paper [55], Janson et al. succeed in analyzing this process precisely for the Erdős-Rényi ($G(n,p)$) random graph. They stated the following results for $G(n,p)$ infection spread with a threshold r. For given r, n, and p define,

$$a_c := \left(1-\frac{1}{r}\right)\left(\frac{(r-1)!}{np^r}\right)^{1/(r-1)}, \tag{2.11}$$

$$\tag{2.12}$$

They analyzed the process and estimated the size of the final active/infected set a^* depending on the size of the initially active set a_0.

Theorem 2.12 (Janson [55])**.** *Suppose* $r \geq 2$ *and* $n^{-1} \ll p \ll n^{-1/r}$.

- *If* $a_0/a_c \to \alpha < 1$*, then* $a^* = \frac{r}{r-1}(\phi(\alpha) + o(1))a_c$ *w.h.p., where* $\phi(\alpha)$ *is the unique root in* $[0,1]$ *of*

$$r\phi(\alpha) - \phi(\alpha)^r = (r-1)\alpha. \tag{2.13}$$

[For $r = 2$*,* $\phi(\alpha) = 1 - \sqrt{1-\alpha}$*.]*

- *If* $a_0/a_c \geq \alpha > 1$*, then* $a^* = n - o(n)$ *w.h.p.; in other words, w.h.p. the process almost percolates.* [5]

We use this theorem to analyze the percolation-based matching algorithm in the $BiG(n,p;s)$ graph model. Although the criterion for propagation is the same in the graph matching process and in the bootstrap percolation ($\geq r$ neighbors infected), the objects of interest in our algorithm are *pairs of nodes* rather than individual nodes as in the Janson et al. model. In other words, in our algorithm, a node pair is matched if it has at least r neighboring node pairs that are already matched. See the details of the algorithm in the next section.

One key result in the section is establishing an equivalence between the percolation process over node pairs for matching and bootstrap percolation, which makes the machinery of [55] available to analyze this process. One subtlety concerns matching errors: They make the process hard to analyze; and they can propagate, thus reducing the quality of the matching. To conclude that the algorithm is correct, we need to show two facts: (i) that the matching process percolates and touches "most" nodes, and (ii) that the algorithm matches nodes correctly.

2.2.2 Percolation Graph Matching Algorithm

We now describe the graph-matching algorithm that addresses the network alignment problem with side information, whose analysis is the main contribution of this section. The matching algorithm has access only to the structure of the two graphs, i.e., it sees *unlabeled* versions of $G_{1,2}$. Its purpose is to find a correct matching π_0

We now describe our PGM algorithm more formally. The input of the algorithm is the following:

- Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$;
- A seed set A_0 of size a_0, consisting of tuples (i,i) of known pairs of matched nodes.

The algorithm we propose and analyze simply matches any two nodes with at least r neighboring pairs that are already matched. An equivalent description emphasizes the incremental nature of the process: we associate with every pair of nodes ($i \in V_1$, $j \in V_2$) a count of marks

[5] For larger value of p the process percolates to the full set w.h.p.

$M_{i,j}$. At each time step t, the algorithm *uses* exactly one unused but already matched pair (i_t, j_t). This pair adds one mark to each neighboring pair, i.e., to every pair in $N_1(i_t) \times N_2(j_t)$, where $N_{1,2}(i)$ is the neighborhood of the node i in $V_{1,2}$. As soon as any pair gets r marks, it is added to the current map; if for some node i there are several nodes j such that all (i, j) have r marks, one pair is picked at random. The process iterates until there are no more unused pairs.

The set $A(t)$ consists of the map built until time t, and the set $Z(t) \subset A(t)$ consists of matched pairs that have been *used* until t. We construct these sets in the following way:

- At time $t = 0$, $A(0) = A_0$ and $Z(0) = \emptyset$,
- At time step t the algorithm randomly selects a pair $(i_t, j_t) \in A(t-1) \setminus Z(t-1)$ and adds one credit mark to all pairs $(i', j') \in V_1 \times V_2$ such that there exist $(i_t, i') \in E_1$ and $(j_t, j') \in E_2$ (cf. Fig. 2.4).

  If a pair (i', j') has more than r marks then it is added to the map $A(t)$; furthermore, all other candidates (i'', j') and (i', j'') are permanently removed from consideration.

  Let $\mathcal{N}(A(t))$ be the set of pairs with r marks, which are added to the map at time t. Then

  $$A(t) = A(t-1) \cup \mathcal{N}(A(t))$$

  and

  $$Z(t) = Z(t-1) \cup \{(i_t, j_t)\}.$$

  Note that $a(t) \geq z(t) = t$, where $a(t) = |A(t)|$ and $z(t) = |Z(t)|$.

The process stops when $A(t) \setminus Z(t) = \emptyset$, which happens when all pairs from the map $A(t)$ are used. Denote this time step by $T = \min(t \geq 0 \text{ s.t. } A(t) \setminus Z(t) = \emptyset)$. The final map is $A^* = A(T) = Z(T)$ and its size is $a^* = T$.

The role of the parameter r is important: it controls the amount of evidence in favor of a pair of nodes, before these nodes are matched permanently. There is a tradeoff between two types of errors. If r is chosen too low, the probability of a false match increases. If r is chosen too high, then the algorithm may simply run out of candidate pairs to match and stops early.

2.2.3 Deferred Matching Variant

The algorithm as defined above leads to a tractable probabilistic model, and in particular, can be analyzed using the bootstrap percolation results from [55], as shown below. The basic algorithm greedily matches any candidate pair as soon as it reaches r credits, *even if* $A(t) \setminus Z(t)$ *is not empty*. This is obviously not optimal in most circumstances, as the credits yet to be

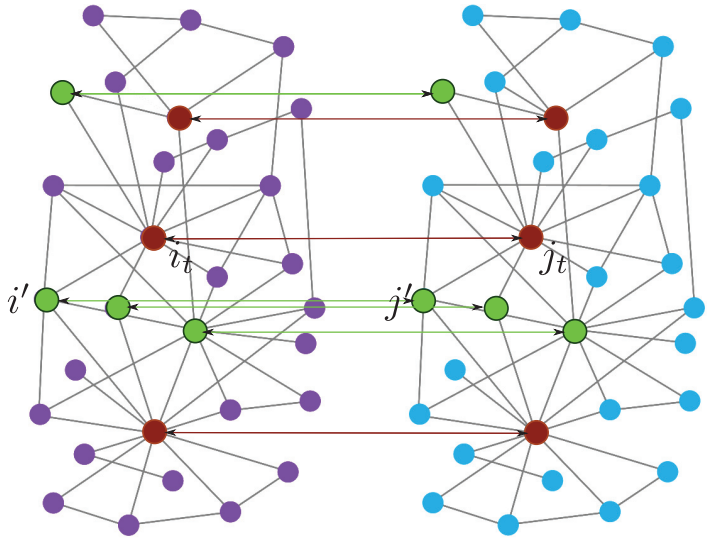

Figure 2.4 – Red nodes are the seeds, green nodes are the set of matched pairs after the first three iterations, for $r = 2$.

generated by the remaining pairs in $A(t) \setminus Z(t)$ might improve the credit counts $M_{i,j}$ and avoid matching errors. There is an easy fix to this, which we describe here; we use this variant of the algorithm in the experiments in Section 2.2.5.

The modified algorithm works as follows. Whenever $A(t) \setminus Z(t)$ is nonempty, we are conservative and continue to attribute credits to candidate pairs, without forming any new couples. Once $A(t) \setminus Z(t)$ is empty, we form exactly one couple (i, j) that has the maximum $M_{i,j}$ of all candidates (provided this is also above the threshold r; otherwise we stop), and add it to $A(t)$; and so forth.

This variant has the advantage of being conservative about matching new couples: it first uses all the available evidence by using all unused pairs before making irreversible decisions. Also, it makes the choice of the parameter r somewhat less important. In particular, if r is chosen too low, the maximum rule ensures that only the best candidate pairs *relative to other candidates* are matched. Our simulation results show that the variant performs well, but exhibits the same phase transition in r as the basic (greedy) approach.

Formally, at each time step t,

- The algorithm processes a matched pair $(i_t, j_t) \in A(t-1) \setminus Z(t-1)$ and adds one credit to every neighboring pair, as in the basic algorithm;
- If $A(t) \setminus Z(t) = \emptyset$, the algorithm takes a pair whose number of credits is maximal and at least r, and adds it to $A(t)$; if there are several such pairs it picks one at random.

The algorithm stops when there are no more pairs with at least r marks.

Our experiments show that this optimization decreases the error rate in certain scenarios, but exhibits similar threshold behavior in the seed set size as the basic version. For more details see Section 2.2.5.

2.2.4 Performance of PGM

Now we analyze the performance of PGM with respect to the initial number of seeds a_0. But first we state the important properties of the propagation process.

Properties of the Propagation Process

Let $E(i,i')$ denote the event that the edge (i,i') is present in G; and $E_1(i,i')$ and $E_2(j,j')$ are the events that edges (i,i') and (j,j') occur in G_1, G_2, respectively.

Observation 2.13. *Since the master graph G is $G(n,p)$, the unconditional edge probability*

$$\mathbb{P}\left(E_1(i,i')\right) = \mathbb{P}\left(E_2(j,j')\right) = ps.$$

But since G_1 and G_2 are sampled from the same generator,

$$\mathbb{P}\left(E_1(i,i')|E_2(i,i')\right) = s.$$

For the convenience of notation, we omit the reference to the graph when it is clear from the context, and we refer to the nodes of G_1 by index i and to the nodes of G_2 by index j. We write $E_1(i,i_t)$ as $E_{i,t}$ and $E_2(j,j_t)$ as $E_{j,t}$. By $i=j$ we mean that i and j correspond to the same node of G.

Let $I_{i,j}(t)$ be an indicator of the event that a pair (i,j) received a mark at time step t, as a result of using a pair (i_t,j_t). This is equivalent to the event that there exist edges $(i,i_t) \in G_1$ and $(j,j_t) \in G_2$. Hence its probability is

$$\mathbb{P}\left(I_{i,j}(t)=1\right) = \mathbb{P}\left(E_{i,t},E_{j,t}\right).$$

We state the following lemma about the increments at time t, conditional on no matching errors so far:

Lemma 2.14. *Conditional on $i_\tau = j_\tau$ for all $\tau \le t$*

1. $\mathbb{P}\left(I_{i,j}(t)=1\right) = \begin{cases} (ps)^2, i \neq j \\ ps^2, i = j \end{cases}$
2. *For fixed t, the $\{I_{i,j}(t)\}_{i,j,i\neq j}$ are not independent.*
3. *For fixed t, the $\{I_{i,i}(t)\}_i$ are independent.*

4. *For fixed $t_1 \neq t_2$, $t_{1,2} \leq t$ and any i, j, the $I_{i,j}(t_1)$ and $I_{i,j}(t_2)$ are independent.*

Proof. Conditional on $i_\tau = j_\tau$ for all $\tau \leq t$

1. If at time t, a seed is mapped correctly, the nodes i_t and j_t are sampled from the same node of G, so by Observation 2.13,

$$\mathbb{P}\left(I_{i,j}(t)=1\right) = \mathbb{P}\left(E_{i,t}, E_{j,t}\right) = \begin{cases} (ps)^2, i \neq j \\ ps^2, i = j \end{cases}$$

2. For $i \neq j$ and $i_1 \neq j$:

$$\mathbb{P}\left(I_{i,j}(t)=1 | I_{i_1,j}(t)=1\right) = \mathbb{P}\left(E_{i,t}, E_{j,t} | E_{i_1,t}, E_{j,t}\right) = \mathbb{P}\left(E_{i,t} | E_{i_1,t}\right) = \mathbb{P}\left(E_{i,t}\right) = ps$$

3. For $i = j$ and $i_1 = j_1$ ($i_1 \neq i$):

$$\mathbb{P}\left(I_{i,j}(t)=1 | I_{i_1,j_1}(t)=1\right) = \mathbb{P}\left(E_{i,t}, E_{j,t} | E_{i_1,t}, E_{j_1,t}\right) = \mathbb{P}\left(E_{i,t}, E_{j,t}\right) = (ps)^2$$

4. For $t_1 \neq t_2$:

$$\mathbb{P}\left(I_{i,j}(t_1)=1 | I_{i,j}(t_2)=1\right) = \mathbb{P}\left(E_{i,t_1}, E_{j,t_1} | E_{i,t_2}, E_{j,t_2}\right) = \mathbb{P}\left(E_{i,t_1}, E_{j,t_1}\right),$$

because for the case $i \neq j$, $\{E_{i,t_2}, E_{j,t_2}\}$ and $\{E_{i,t_1}, E_{j,t_1}\}$ are disjoint, and for the case $i = j$, $E_{i,t_2} \neq E_{i,t_1}$.

□

Clause 1 of the Lemma 2.14 says that a correct pair has a probability of collecting a new marker that is larger by a factor of $1/p$ than the probability of a wrong pair collecting a marker, as long as the pairs generating the credits are correct. While there are many more incorrect pairs than correct pairs ($\Theta(n^2)$ vs n), this difference in the marker rates for correct and wrong pairs is the reason why the algorithm can work well, provided the factor $1/p$ is large enough.

Clause 2 states that markers obtained for two different pairs with a node in common are not independent. Given that a pair (i, j) gets a mark, the event that another pair (i_1, j) also gets a mark is more likely. Clause 3 is key for further analysis of the process. It states that correctly mapped pairs obtain marks independently. Thus, if a pair (i, i) got a mark at time t, it does not correlate with (j, j) getting a mark. Clause 4 asserts that each seed spreads its marks independently. In other words, at a time step t, a pair gets a mark independently of other time steps.

The count $M_{i,j}(t)$ is the number of marks of (i,j) at time t:

$$M_{i,j}(t) = \sum_{s=1}^{t} I_{i,j}(s).$$

Under the conditions of Lemma 2.14, each $M_{i,j}(t)$ is the sum of i.i.d. Bernoulli random variables, so it is either a $\mathsf{Bi}\left(n,(ps)^2\right)$ (Binomial) for $i \neq j$ or a $\mathsf{Bi}\left(n,ps^2\right)$ for $i = j$. In the following section, we develop conditions when PGM does not match wrong pairs (w.h.p.).

Main Theorems

Let $r \geq 4$, and note that ps^2 is the probability of an edge being sampled in both G_1 and G_2 or, equivalently, the probability of an edge to be contained in the intersection of the edge sets $E_1 \cap E_2$. Recall that $a_c = \left(1-\frac{1}{r}\right)\left(\frac{(r-1)!}{n(ps^2)^r}\right)^{\frac{1}{r-1}}$. We show that a_c is the critical value of the initial size of the seed set and prove theorems stated in the contribution section.

Theorem 2.15 (Subcritical regime)**.** *Fix $\varepsilon > 0$. For $n^{-1} \ll ps^2 \ll s^2 n^{-\frac{3}{r}-\varepsilon}/\log n$, if $a_0/a_c \leq \alpha < 1$, the PGM algorithm stops with $a^* \leq \frac{r}{r-1}a_c$ w.h.p. In particular $a^* = (\phi(\alpha)+o(1))\frac{r}{r-1}a_c \leq \frac{r}{r-1}a_0$, where $\phi(\alpha)$ is the unique root in $[0,1]$ of $r\phi(\alpha)) - \phi(\alpha)^r = (r-1)\alpha$.*

Now we consider what happens above the threshold $a_0 > a_c$.

Theorem 2.16 (Supercritical regime)**.** *Fix $\varepsilon > 0$. For $n^{-1} \ll ps^2 \ll s^2 n^{-\frac{3}{r}-\varepsilon}/\log n$, if $a_0/a_c \geq \alpha > 1$ the algorithm propagates, and the size of the final mapping is $a^* = n - o(n)$ w.h.p.*

We discuss the implications of this phase transition and the scaling of the main parameters in more detail in Subsection 2.2.4.

Proof Sketch and Bootstrap Percolation

We briefly outline the main steps of the proof and provide full details in the next subsection.

Our main goal is to prove that the couple formation process $A(t)$ defined in the previous section can be analyzed using the bootstrap percolation model introduced in [55]. In summary, in [55] authors analyze a process where, at every time step t, objects collect a credit with probability p, independently of everything else. We want to analyze the PGM algorithm within the $BiG(n,p;s)$ model. However, our object of interest is not an individual node, but a pair (i,j).

At every time-step, one pair spreads credits to other node pairs. However, we do not have the critical feature that makes the analysis of [55] tractable: as shown in Lemma 2.14, the credit increments $I_{i,j}(t)$ are not equiprobable, and they are not independent. Therefore, the results of [55] cannot be applied directly.

Fortunately, the specific structure of the process of increments over pairs reveals a way out. The key observation is that the credits of *correct* pairs $M_{i,i}(t)$ are in fact independent of each

other. Another observation of Lemma 2.14 is that correct pairs are more likely to obtain a mark. Thus, the (small) subset of pairs of the form (i,i) within all the possible pairs $V \times V$ can be analyzed using the bootstrap percolation framework.

Therefore, we first consider the event X that at any time t, a wrong pair (i,j) has collected at least r credits without either of the "competing" correct pairs (i,i) and (j,j) having collected r credits. In the case of event X, it is possible (but not guaranteed) that a matching error has occurred.

In Lemma 2.17 below, we show that $\mathbb{P}(X) \to 0$, under appropriate conditions. Under these conditions, the matching algorithm does not make wrong matches (w.h.p.) and is suitable for further analysis.

It remains to be shown whether the algorithm percolates. For this, it is conservative to only consider the credits $M_{i,i}$ attributed to correct pairs, as the probability of percolating can only increase if additional pairs are added into the system.

As the correct counts are independent binomials, it is then straightforward to map the problem into the bootstrap percolation framework.

Proofs of Theorems 2.15 and 2.16

In this section, we use the results from [55] to formulate our key results. We show a sharp face transitions in the final map size a^* depending on $a_0 < a_c$ or $a_0 > a_c$.

A key lemma bounds the probability that no error happens in the matching process. An error may occur if at some time step, a bad pair (i,j) collects r marks before its adjacent good pairs (i,i) and (j,j) have collected more than r marks. If such errors are very rare, then we can focus only on correctly mapped pairs in the analysis of $A(t)$. Let $X_{i,j}(t)$ denote the event that the algorithm made an error at time step t by mapping a pair $(i,j), i \neq j$, where $r \leq t \leq n$. The probability of this event is

$$\mathbb{P}\left(X_{i,j}(t)\right) \leq \mathbb{P}\left(M_{i,j}(t) = r, M_{i,i}(t) \leq r, M_{j,j}(t) \leq r\right)$$

Denote by $X = \bigcup_{t, i \neq j} X_{i,j}(t)$ the event that at any time-step t an error happened.

Lemma 2.17. *If $p \ll n^{-\frac{3}{r}-\varepsilon}/\log n$ (with $r \geq 4$), then $\mathbb{P}(X) \to 0$ with $n \to \infty$,*

Proof. First we bound the probability of mapping a wrong pair (i,j) $(i \neq j)$ at time step t, conditional on no wrong used pairs up to time $t-1$. Note that conditioning on a correct used pair $i_\tau = j_\tau$ at time τ implies that this pair was correctly matched at some time τ' before τ, which in turn ascertains that for this pair, the correct count $M_{i_\tau,i_\tau}(\tau')$ "won" over all wrong counts $M_{i',i_\tau}(\tau')$ and $M_{i_\tau,j'}(\tau')$. Therefore, conditional on $i_\tau = j_\tau$ for $\tau < t$, correct counts

are stochastically (slightly) larger, and wrong counts stochastically smaller. In the following argument, we are conservative in ignoring this bias in bounding the probability of future errors.

$$\begin{aligned}\mathbb{P}\left(X_{i,j}(t)\right) &\leq \mathbb{P}\left(M_{i,j}(t) = r, M_{i,i}(t) \leq r, M_{j,j}(t) \leq r\right) \\ &\leq \mathbb{P}\left(M_{i,j}(t) = r, M_{i,i}(t) \leq r\right)\end{aligned}$$

We split our proof into two cases: for earlier steps t we upper-bound $\mathbb{P}\left(X_{i,j}(t)\right)$ by $\mathbb{P}\left(M_{i,j}(t) = r\right)$ and show that $\mathbb{P}\left(M_{i,j}(t) = r\right) \to 0$ or, equivalently, we show that for earlier steps t the wrong pairs do not get r marks w.h.p. For later steps we upper-bound $\mathbb{P}\left(X_{i,j}(t)\right)$ by $\mathbb{P}\left(M_{i,i}(t) \leq r\right)$ and demonstrate that $\mathbb{P}\left(M_{i,i}(t) \leq r\right) \to 0$ or, equivalently, we show that the pair (i,i) has collected more marks than r w.h.p. Then we take a union bound for all i, j and t to upper-bound $\mathbb{P}(X)$.

Fix an arbitrary $\varepsilon > 0$.

- Early steps t: let $t(ps)^2 \leq n^{-\frac{3}{r}-\varepsilon}$ and let X_1 be an event that an error happens early (for all i, j and t satisfying the condition above)

$$\begin{aligned}\mathbb{P}(X_1) &\leq \sum_{i,j,t} \mathbb{P}\left(X_{i,j}(t)\right) \\ &\leq \sum_{i,j,t} \mathbb{P}\left(M_{i,j}(t) = r\right) \\ &= \sum_{i,j,t} \mathbb{P}\left(\mathsf{Bi}\left(t,(ps)^2\right) = r\right) \\ &= \sum_{i,j,t} \binom{t}{r} (ps)^{2r}(1-(ps)^2)^{t-r} \\ &\leq \sum_{i,j,t} (t(ps)^2)^r \\ &\overset{(a)}{\leq} n^3(n^{-\frac{3}{r}-\varepsilon})^r = n^{-r\varepsilon}, \text{ thus} \\ \mathbb{P}(X_1) &\to 0\end{aligned}$$

  where (a) follows from the condition on t.

- Later steps t: let $t(ps)^2 > n^{-\frac{3}{r}-\varepsilon}$, equivalently, $t > \left(\frac{n^{-\frac{3}{r}-\varepsilon}}{(ps)^2}\right)$ and let X_2 be an event that an

error happens at later steps (for all i, j and corresponding t).

$$\begin{aligned}
\mathbb{P}(X_2) &\leq \sum_{i,j,t} \mathbb{P}\left(X_{i,j}(t)\right) \\
&\leq \sum_{i,j,t} \mathbb{P}\left(M_{i,i}(t) \leq r\right) \\
&= \sum_{i,j,t} \mathbb{P}\left(\mathsf{Bi}\left(t, ps^2\right) \leq r\right) \\
&\overset{(a)}{\leq} \sum_{i,j,t} \exp\left(r - tps^2/2\right) \\
&\overset{(b)}{\leq} n^3 \exp\left(r - \frac{n^{-\frac{3}{r}-\varepsilon}}{2p}\right) \\
&= \exp\left(3\log n + r - \frac{1}{2n^{\frac{3}{r}+\varepsilon}p}\right), \text{ thus} \\
\mathbb{P}(X_2) &\to 0, \text{ if } p \ll n^{-\frac{3}{r}-\varepsilon}/\log n
\end{aligned}$$

where (b) follows from the condition on t and (a) uses the following Chernoff bound for the left tail of the binomial:

$$\mathbb{P}\left(X \leq (1-\sigma)\mu\right) \leq \exp\left(\frac{-\sigma^2\mu}{2}\right).$$

Here X is $\mathsf{Bi}\left(t, ps^2\right)$, $\mu = tps^2$ and $\sigma = 1 - \frac{r}{tps^2}$ (to make $(1-\sigma)\mu = r$). Then,

$$\begin{aligned}
\mathbb{P}\left(\mathsf{Bi}\left(t, ps^2\right) \leq r\right) &\leq \exp\left(\frac{-(1-\frac{r}{tps^2})^2 tps^2}{2}\right) \\
&= \exp\left(-\frac{tps^2}{2} + r - \frac{r^2}{2tps^2}\right) \\
&\leq \exp\left(r - tps^2/2\right)
\end{aligned}$$

Taking a union bound we obtain $\mathbb{P}(X) \leq \mathbb{P}(X_1) + \mathbb{P}(X_2) \to 0$, if $p \ll n^{-\frac{3}{r}-\varepsilon}/\log n$. □

Therefore for $p \ll n^{-\frac{3}{r}-\varepsilon}/\log n$, Lemma 2.17 guarantees that w.h.p. we need only consider the evolution of the correct counts $\{M_{i,i}\}$, to which we can apply the results of [55] directly.

Proof. [Theorems 2.15 and 2.16]

The PGM process restricted to correct pairs (i,i) is isomorphic to the bootstrap percolation process for a $G(n,h)$ random graph, with $h = ps^2$. Consider the two events $\{\{M_{i,i}\}$ percolates $\}$ and $\overline{X}$. In the supercritical case, by virtue of Theorem 2.12 and Lemma 2.17, both events occur with high probability. Therefore, PGM percolates correctly and to a set of size $n - o(n)$ w.h.p.

In the subcritical case, the process $\{M_{i,i}\}$ does not percolate. As $\mathbb{P}(X) \to 0$, the full PGM process over all pairs does not percolate either by virtue of Lemma 2.17. □

Interpretation of Results

Here we look into more details on the parameters of the algorithm. In particular, we consider how the threshold a_c scales with respect to r, p and s, and we elaborate on what happens near the bounding conditions on $h = ps^2$.

The parameter r controls a tradeoff between matching errors and percolation blocking. If r is too low, then a wrong pair (i, j) might accumulate r credits before the correct pairs (i, i) and (j, j) do; if r is too high, the process might not percolate, and most nodes do not get matched. Note that r has to be at least 2 for the algorithm to work: For $r = 1$, the algorithm would match pairs of nodes with only one mapped neighbor, which would necessarily lead to ambiguity, except in degenerate cases.

The lower bound $\frac{1}{n} \ll h$ simply ensures that the intersection of the two graphs has a giant component, without which the algorithm cannot percolate. The upper bound $h \ll \frac{s^2 n^{-\frac{3}{r}-\varepsilon}}{\log n}$ is more subtle. Of course, if h exceeds the upper bound, the algorithm still percolates, but it will make errors. This is because the ratio in the probabilities of generating correct and wrong credits is not large enough to guarantee $\overline{X}$. As expected, the threshold a_c decreases with increasing p and s, hence denser graphs require smaller seed sets. For most scenarios of practical interest, r would be a constant. For example, if s is a constant, and the mean degree nh grows as n^δ, with $0 < \delta < 1$ a constant, then there is a constant r that satisfies the upper bound $h \ll s^2 n^{-\frac{3}{r}-\varepsilon}/\log n$. In this case, the seed set threshold scales as $a_c = \Theta(n^{1-\delta\frac{r}{r-1}})$. If the average degree grows, but less than a power law, then a_c is closer to linear. For example, suppose the mean is $nh = \Theta(\log n)$ (which is the threshold for the disappearance of symmetry and of isolated vertices [24]), then a_c scales as follows: $a_c = (1 - \frac{1}{r})(r-1)!^{\frac{1}{r-1}} n(\log n)^{-r/r-1}$. With $r = 4$ this is $a_c = \Theta(n \log^{-\frac{4}{3}} n)$.

2.2.5 Simulation Results

In this section, we test the PGM algorithm over real and artificial graphs, with two goals: to validate the phase transitions predicted by theory, and to check how well the algorithm performs on real networks.

To evaluate the performance of the algorithm, we use two metrics: The first is the size of final the map a^* (the total number of mapped pairs), which says how far the algorithm propagates. The second is the error rate, i.e., the fraction of wrong pairs in the map. Recall that the error rate is $\frac{|(i,j):i\neq j,(i,j)\in A^*|}{a^*-a_0}$.

The following ground-truth network graphs are considered:

- Erdös-Rényi random graph $G(n,p)$;
- Slashdot social network;
- EPFL e-mail exchange network;
- Geometric random graph $G_{geom}(n,d)$.

We run the deferred matching version of the algorithm (see Section 2.2.3) with $r = 2$; however the results are qualitatively similar for the basic version. For the $G(n,p)$, Slashdot and $G_{geom}(n,d)$ random graph, we use the $BiG(G;s)$ sampling model: each edge appears in the observed network with probability s. The experiment with the EPFL network is in some sense more challenging, because the two networks to be matched are in fact different observations of the social interactions within an organization at two different points in time. Figures 2.5 - 2.12 show the dependence of the performance metrics on the size of the seed set a_0. Each figure contains 3 curves for different values of the sampling parameter, which is either the sampling parameter s, or an estimate of s in the case of the EPFL dataset. The value of the parameter s^2 determines the size of the overlap of the observed networks: the intersection of the edge sets of the two graphs have a size proportional to s^2. We average all the results over 10 realizations.

$BiG(n,p;s)$ Model

To support our results, we first simulate the $BiG(n,p;s)$ graph matching model exactly. Specifically, in this model, the generator graph G is an Erdös-Rényi $G(n,p)$ graph with $n = 100000$ and $p = 20/n$.

We observe that when the size of the seed set is sufficiently large, the algorithm propagates to the complete mapping (see Figure 2.5). We also see the sharp phase transitions predicted in Theorems 2.15 and 2.16. Furthermore, the theoretically obtained threshold a_c appears very precise. According to the definition (1.1) of a_c, for the first curve, $s = 0.9$ the critical size of the seed set a_c is 191, for $s = 0.8$ the a_c is 305 and for $s = 0.7$ the a_c is 520. We can see that the observed transitions are close to these values. To highlight this fact, we normalize the x-axis by the computed a_c. In Figure 2.6, we observe that, after re-scaling, all the curves look essentially the same. The number of wrong pairs in all the experiments is negligible (~ 0.0001 of n) hence we do not plot it.

To confirm that deferred version does not change the observed phenomenons, we also ran the analogous experiments with the basic version of the PGM and observe identical threshold behavior.

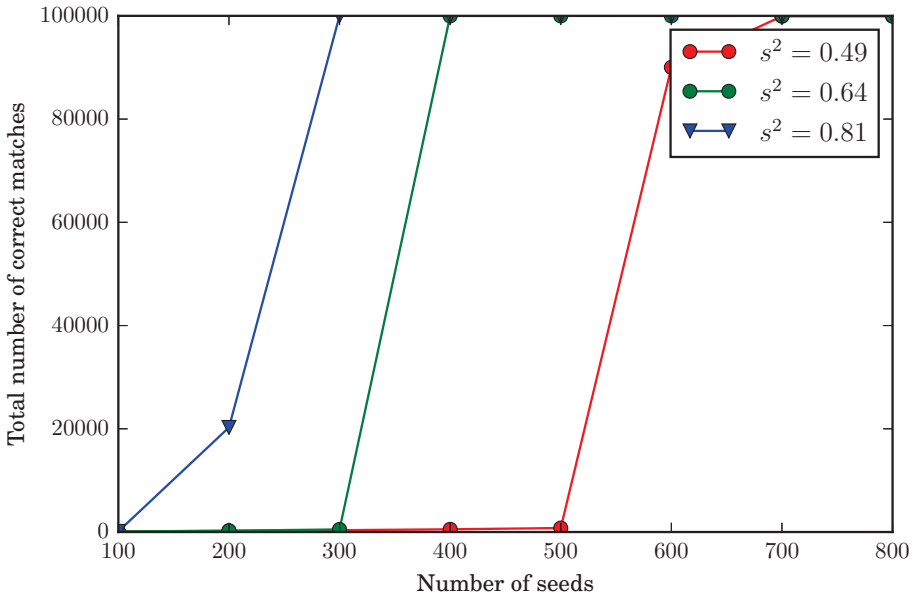

Figure 2.5 – Total number of correctly matched pairs vs number of seeds for the PGM algorithm over $G(n,p)$ with $n = 10^5$ and $p = 20/n$.

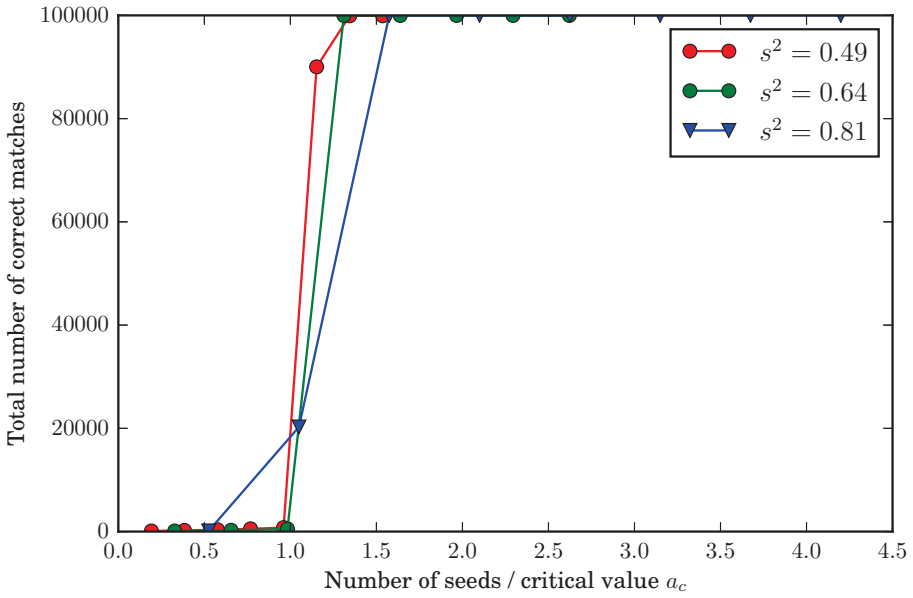

Figure 2.6 – Total number of correctly matched pairs vs number of seeds for the PGM algorithm over $G(n,p)$ with $n = 10^5$ and $p = 20/n$. The x-axis is rescaled according to the value of a_c.

Real Networks: Slashdot and E-mail Graphs

In the second set of experiments, we run PGM over large-scale social networks. First, we run the algorithm over real friend/foe links between Slashdot users [75] obtained in November 2008 (cf. Table 2.3).

Nodes	77360
Edges	546487
Number of components	1
Average clustering coefficient	0.0555
Diameter (longest shortest path)	10

Table 2.3 – Slashdot dataset statistics.

To generate two observations of the network, we resort to edge sampling. In this model, when $s = 0.9$, the overlap of the two networks is less than 63000 nodes; when $s = 0.8$, the overlap is about 49000 nodes; when $s = 0.7$, it is 38000 nodes.

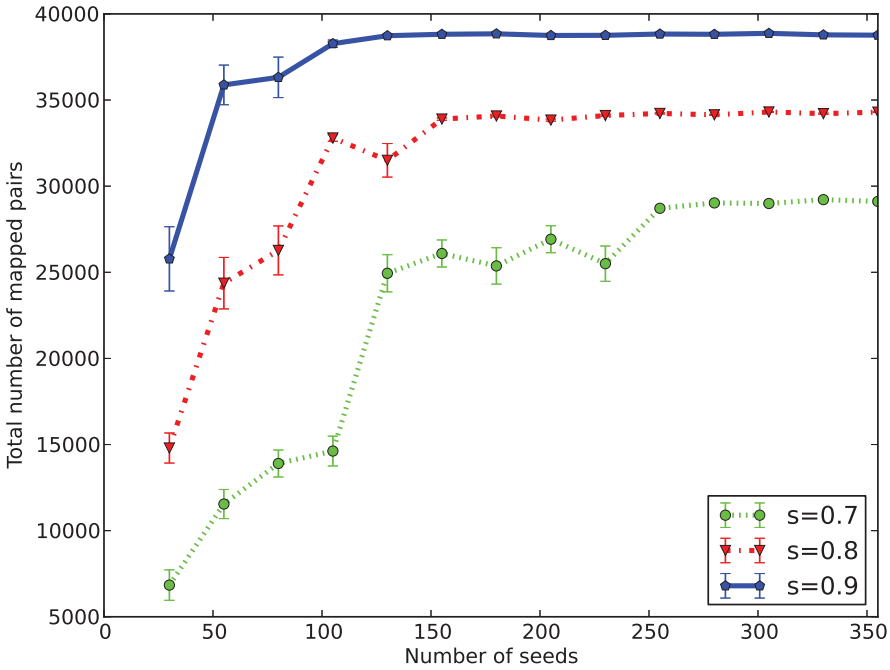

Figure 2.7 – Total number of mapped pairs vs number of seeds for the PGM algorithm over the Slashdot network.

The results suggest phase transitions in the size of the final mapping, albeit less sharp than for $G(n;p)$ (see Figure 2.7). We also see that if the algorithm propagates (supercritical case), the error rate is encouragingly small (see Figure 2.8). For example, for $s = 0.9$, it is enough to

have 150 seeds (which is 0.2% of all nodes) for the algorithm to propagate over the majority of the graph. Figure 2.8 shows that the error rate drops rapidly with a_0.

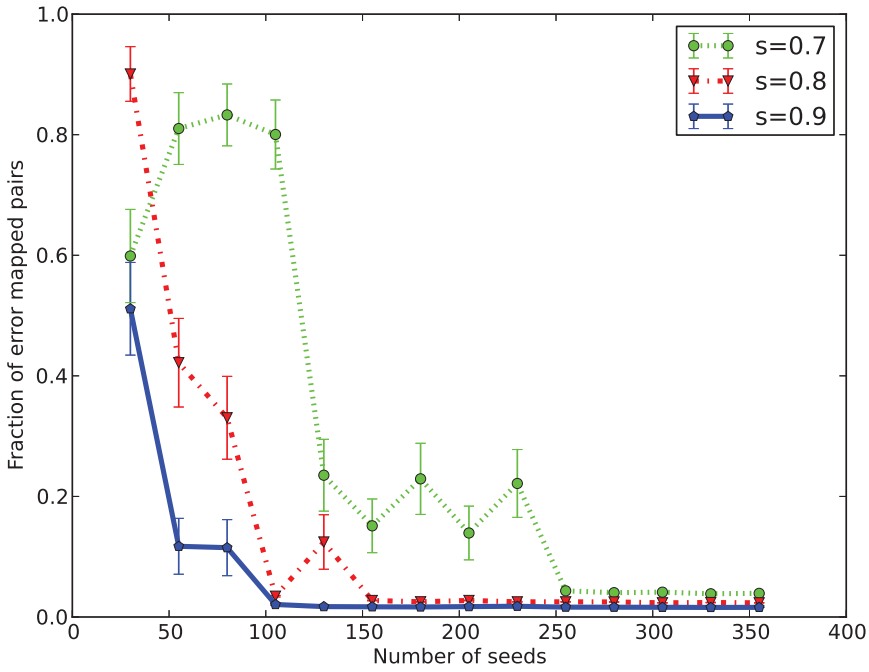

Figure 2.8 – Error rate vs number of seeds for the PGM algorithm over the Slashdot network.

Second, we obtained snapshots of the e-mail traffic on the EPFL campus for different time periods (the week numbering starts at the beginning of year). Each node corresponds to an e-mail account, and an undirected edge means that at least one e-mail was sent between two accounts. The experiment is more realistic in the sense that we do not rely on the $BiG(G;s)$ sampling model to generate two similar graphs $G_{1,2}$, but instead these graphs correspond to the real traffic patterns in two different time periods.

The challenge for the algorithm is that in the considered graphs not only the edge sets are different, but so are the vertex sets. In other words, the PGM does not match the vertex sets of two graphs anymore, instead it identifies common subsets and matches them. If the two graphs are different enough the PGM cannot separate the nodes that are not presented in both graphs and tries to match them thus increasing the error rate. Another challenge is that the graphs are quite sparse, the average degree is about 7. The three curves demonstrate the behavior of the algorithm on the e-mail exchanges graphs for the following periods:

- G_1 is a graph of e-mails sent between weeks 3 and 17 and G_2 is a graph of e-mails sent between weeks 8 and 12. Each graph contains approximately 60000 nodes and 230000 edges. The intersection graph has 50000 nodes and 160000 edges.

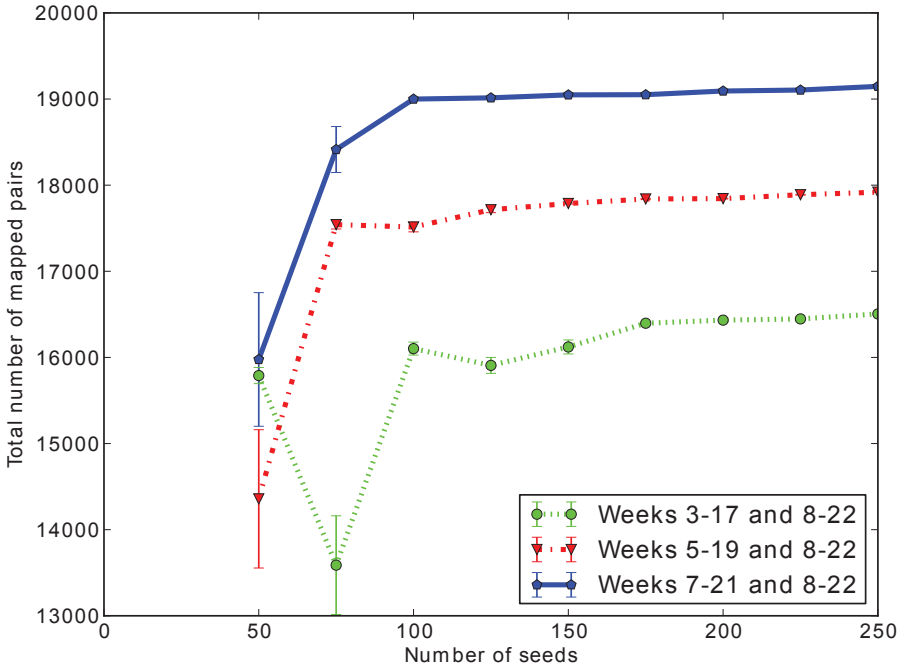

Figure 2.9 – Total number of mapped pairs vs number of seeds for the PGM algorithm over the EPFL contact network.

- For weeks 5-19 and 8-12, respectively: Each graph contains approximately 61500 nodes and 231000 edges. The intersection graph has 54000 nodes and 185000 edges.
- For weeks 7-21 and 8-12, respectively: Each graph contains approximately 61500 nodes and 231000 edges. The intersection graph has 59000 nodes and 207000 edges.

The results reveal similar phase transitions on the size of the final mapping and error rate as for those in Slashdot(see Figures 2.9 and 2.10). Moreover, we observe that given enough seeds the algorithm correctly identifies nodes that are present in both graphs thus suggesting that the PGM is robust to partial node overlap. This was confirmed later in [60].

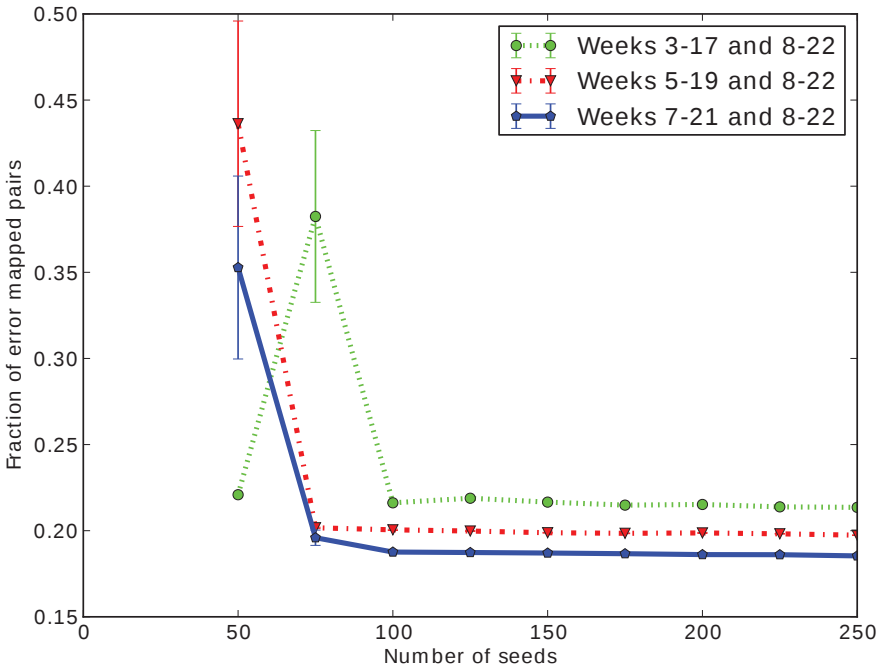

Figure 2.10 – Error rate vs number of seeds for the PGM algorithm over the EPFL contact network.

Random Geometric Graph $G_{geom}(n,d)$

The performance of our proposed PGM algorithm is surprisingly good over both the $G(n,p;s)$ random graphs and over real social networks. We conjecture, however, that its success relies in part on the compactness of these graphs, which ensures that even with a relatively small number of seeds, every node in the network is close to some seeds, which allows to "triangulate" the nodes.

To illustrate this, we report on an experiment where the generator graph G is a random geometric graph $G_{geom}(n,d)$. A random geometric graph is a random undirected graph which is generated by placing vertices uniformly at random on the unit square $[0,1)^2$. Two vertices u and v are connected if and only if the distance between them is at most d [90]. The typical distance in a supercritical random geometric graph scales as $n^{1/2}$, in contrast to the logarithmic distance in $G(n,p)$ and other "small-world" networks.

Figures 2.11 and 2.12 show the experiment for $G_{geom}(n = 10000, d = 0.01)$. We observe that the algorithm does not percolate, and that it has a very high error rate within the map. While a complete understanding of the limits of percolation-based graph matching is lacking, this does suggest that PGM performs better with compact networks.

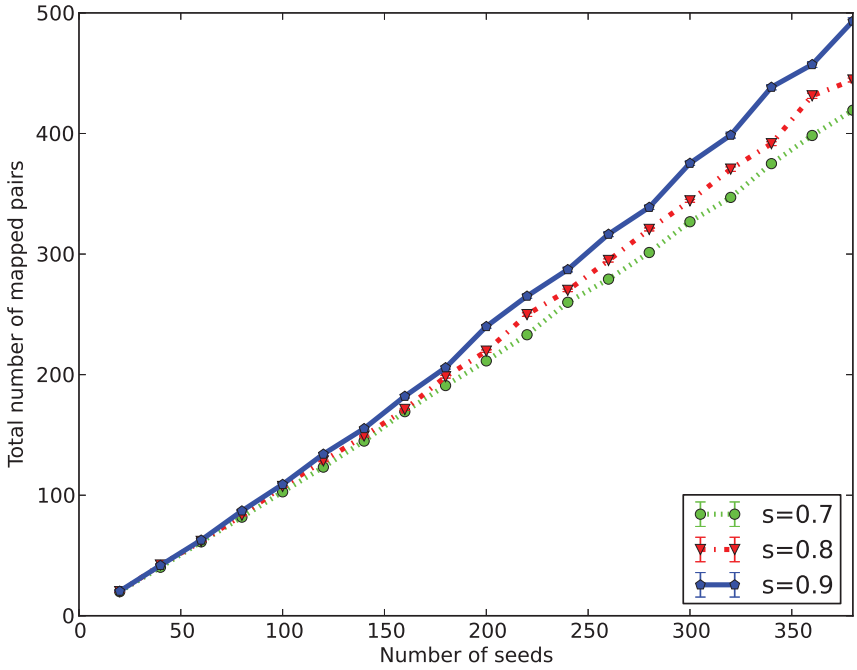

Figure 2.11 – Total number of mapped pairs vs number of seeds for the PGM algorithm over $G(n,d)$ random geometric graph model where $n = 10000$ and $d = 0.01$.

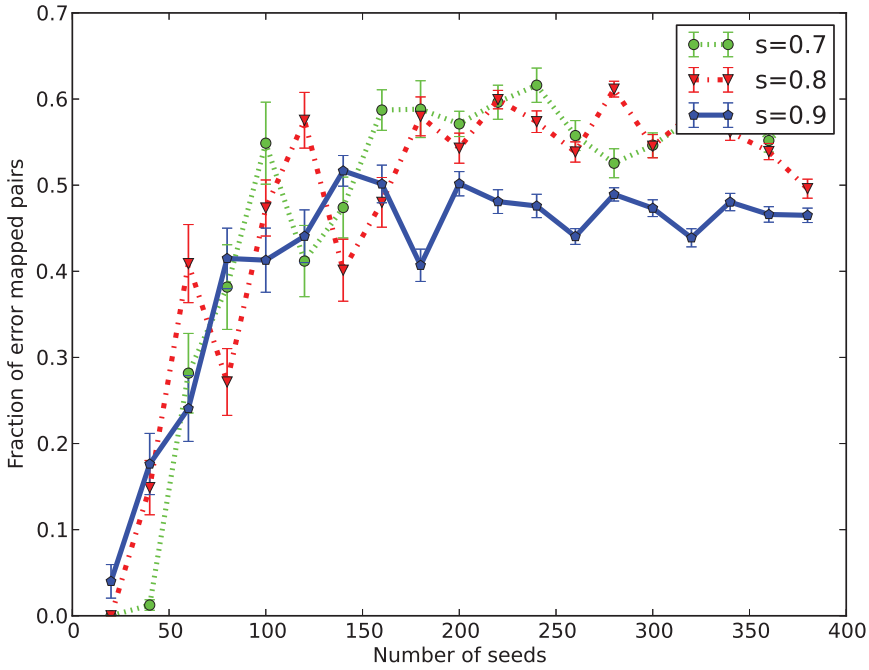

Figure 2.12 – Error rate vs number of seeds for the PGM algorithm over $G(n,d)$ random geometric graph model where $n = 10000$ and $d = 0.01$.

2.2.6 Scalability Optimization

In this section, we describe two optimizations of PGM which enable us to push the sizes of considered graphs (sampled from $BiG(n,p;t,s)$ model) to millions of nodes. We introduce time and space complexity improvements that involve embedding efficient datastructures for sequential implementation and adapting the PGM for parallel implementation. These two improvements address different bottlenecks of the algorithm, by thus assuring full flexibility of the experiments. The parallel implementation is less flexible; for example, it does not maintain a deferred version of PGM, however it enables running PGM for larger graphs faster.

Efficient Datastructures

Here we describe optimizations of a sequential implementation that allows to address the main bottlenecks of the first implementations of the PGM. The improvements are implemented joined with Jefferson Elbert.

One of the drawbacks of the straightforward approach described in Sections 2.2.2 and 2.2.3 is maintaining (and operating) the structure that contains all the accumulated marks of pairs that are not matched up to the current step t. In the basic version, we have to store information about all the pairs $(i,j) : i \in V_1, j \in V_2$ such that (i,j) has less than r number of marks. In the deferred version, we store information about all the pairs $(i,j) : i \in V_1, j \in V_2$ such that (i,j) is not mapped yet. This structure can take $O(n^2)$ space and takes a long time to operate. We need to maintain two basic operations: take a pair with maximal number of marks (or all the pairs with marks above r) and update a number of marks of a given pair.

To optimize the first operation, we store the current candidate-mappings as a hash-map of lists grouped by r (here r is a key and list of pairs with r marks is a value). We use the fact that on average we have 10-20 of such groups, as r does not grow too large. Then it takes constant time to select a best match (any element of the list with max key). Another "expensive" operation is to locate a pair in a list to increment its number of marks, while processing a current seed. To optimize this step we maintain another structure that is a map, with key being a pair and value being a reference to the position of the pair in the group from above.

We introduce a third improvement, that is a map that stores positions of pairs grouped by left and right entities (instead of a map where the whole pair is a key we use two nested maps with a left entity as a first key and a right entity as a second key and another two nested maps for a reversed search). This improvement enables us quickly locate a pair by its left and right unit. Obviously these structures need tedious maintenance of cross consistency. However, with this implementation, we were able to fully align two graphs of size 10^6 with 3000 seeds in around 2 hours.

MapReduce Implementation

This implementation was done during a semester project by Jéremy Weber, co-supervised by Ehsan Kazemi and myself.

As described above, operating over large amount of candidates can be time expensive, and as currently there are networks of billions of nodes, the capacity of a single machine is not sufficient for our purposes. The key intuition for this optimization is that the main steps of PGM are easy to adapt for parallelization. Indeed, spreading marks from each seed is independent of spreading marks from any other seeds; collecting marks, as well, can be grouped into "word-count" type of jobs.

Using this observation, we implement a PGM as a sequence of four map-reduce jobs that include formatting of the input graphs for Hadoop.

We iterate the following main steps of the program, where each step is a map-reduce job:

- we spread marks from all current generation seeds, that are not used pairs with more than r marks (parallel)
- we group and count marks of all the pairs that were given at previous step (parallel)
- we extract all the pairs with more than r marks (parallel).

As it is shown in [55], there are only few generations of the percolation process until it stops. Hence we need only limited number of iterations of these steps.

The improved versions of PGM from [60] such as ExpandWhenStuck are implemented as well. As a result, from less than 20 seeds we are able to align graphs sampled from $BiG(n;p,s)$ with 10 million nodes on the Hadoop Cluster in less than 20 min.

3 Network Assembly

In this chapter we address the second scenario of the network reconstruction problem. For an input we assume multiple, possibly noisy, small observations extracted from a master-graph. These observations are called *patches*. According to Definition 1.5, which defines the problem in its most general form, the graph-assembly problem takes as input a finite collection of patches and puts these pieces together in an assembled graph $\hat{G}$. For example, if, to anonymize a phonecall network, the data provider shared only unlabeled egonets [21] then, assembling these patches back to the original network means de-anonymizing this scheme.

We consider two scenarios: First, we assume some restriction on the number and structure of the patches (patches being egonets) and full ambiguity in labels of the nodes (By full ambiguity we mean that the labels of vertices in each patch bear no resemblance at all to their original labels in the master graph). In Section 3.1 we study the feasibility of an assembly for a master graph generated via the $G(n, p; q)$ model. The second scenario has no assumption about the form of the patches, however it relies on meaningful labels from some small label set (labels of nodes in patches are the same as those of nodes of the master graph). We see that, even if the label set is extremely small compared to the size of the master graph, the assembly is still possible. We propose a practical algorithm for network assembly and evaluate it in Section 3.2.

In both scenarios, we try to identify small specific subgraphs that serve as markers or gluing points for merging patches together. In the first version of the problem, these subgraphs are unique through the whole master graph and they uniquely identify some nodes across egonets. In the second scenario, we consider rare specific subgraphs and use their frequency to compute a graph similarity. We call these *seed-subgraphs* by analogy with seeds in the network alignment problem.

3.1 Feasibility of Network Assembly from Ambiguous Patches

In this section, we consider a specific variation of this problem, where each patch is created by extracting the egonet around each vertex in the master graph. Recall, the *egonet*, or 1-egonet of a vertex i in a graph G, denoted G_i, is the induced subgraph generated by i and its neighbors in G — we say that i is the *center* of this egonet. We assume that, for each egonet in the patch collection, the identity of i is either kept intact or somehow inferable, but all other identities are removed.

Recall that we deal with an *unlabeled egonet collection* $\mathscr{P} = \{G_i = f_i(H_i)\}_{i\in[n]}$ of a master graph G, where each patching function $f_i : V(H_i) \to [|V(G_i)|]$ is a bijection such that $f_i(i) = 1$. Clearly, we want a guarantee to have the assembled graph $\hat{G}$ equal to the master graph G. Indeed, if $\mathscr{P} = \{G_i\}_{i\in[n]}$ is an unlabeled egonet collection of G — that is, for every $i \in [n]$, $G_i = f_i(G_i)$ — it is not difficult to see that $(G, \{f_i^{-1}\})$ is a valid assembly of $\mathscr{P}$. The interesting theoretical question is whether G is the only graph for which there is such an assembly.

3.1.1 $G(n, p; q)$ Model

For some random graphs models it was shown that egonet patches should be relatively large to contain enough information to make assembly feasible [83, 84]. However in real networks, neighborhoods of nodes are more complex and highly connected (i.e., have high clustering coefficient), thus containing richer structure. For example, in social networks, friends of any given person are more likely to know each other. This behavior is called triadic closure [100]. We suggest here that highly clustered networks are easier to assembly and address the question of how a clustering coefficient is correlated with the feasibility of assembly.

For this purpose, we introduce the $G(n, p; q)$ random graph model and analyze its properties. The graph $G \sim G(n, p; q)$ is obtained by taking $V = [n]$ as a vertex set and generating a random edge set E over V according to the following steps,

1. An intermediate Erdős-Rényi graph over V is generated: an undirected edge is drawn between each pair of vertices with probability p, independently of the other pairs. We denote the obtained graph by $G_p(V, E_p)$.

2. For each connected triple (u, g, v) in G_p, i.e., a triple of vertices such that $(u, g), (g, v) \in E_p$, we add an edge (u, v) to E by closing the triangle with probability q, independently for each triple. In this case, we say that g is a *generator* for the q-edge (u, v). All q-edges are allowed to co-exist with previous p-edges if they overlap with them. If any q-edge has more than one generator, the resulting q-edges are collapsed. The obtained graph is denoted $G = (V, E)$.

For convenience, we denote by $P_e = \mathbf{1}_{\{e\in E_p\}}$ the indicators of the edges in G_p, with $Q_e = \mathbf{1}_{\{e\in E\}}$ being the indicators of the edges in our final graph G. We refer to the edges in E_p as *p-edges*

and the edges in E as *q-edges*.

Define the set of independent $\mathsf{Be}(q)$ [1] random variables $\{T_t\}_{t\in V\times\binom{V}{2}}$ and put $T_{u,g,v} = T_{v,g,u}$. Let t be a connected triple (u,g,v) in G_p, i.e., a pair of incident edges $(u,g),(g,v) \in E_p$. The idea is that, for each such connected triple, we apply triadic closure with probability q. Thus the edge $(u,v) \in E$ if and only if $T_{u,g,v} = 1$ for at least one $g \in V \setminus \{u,v\}$ that is connected to both u and v by p-edges. We define

$$S_e = S_{(u,v)} = \{g \in V \,:\, P_{u,g}P_{g,v}T_{u,g,v} = 1\}$$

the set of generators for an edge (u,v). Thus, there is an edge $e \in E$ iff it has at least one generator.

Some additional useful definitions are as follows: for any $u \in V$, the *neighborhood* N_u of u is the set of vertices adjacent to u in G (thus, via q-edges), with $d_u = |N_u|$ its degree, and the *p-neighborhood* N_u^p of u is the set of vertices adjacent to u in G_p (via p-edges).

Remark 1. *The following facts hold:*

1. *For any $e \in \binom{V}{2}$, the size of the set of generators is $|S_e| = Bi(n, p^2q)$;* [2]
2. *For any $e \in \binom{V}{2}$, the probability of an edge is $Q_e = Be\left(1-(1-p^2q)^n\right)$.* [3]

We show some key properties of this model. Let c_u be the clustering coefficient of node u.

Proposition 3.1. *Let $u \in V$ be arbitrary. If $np \to \infty$, $n^2p^3 \to 0$ and q is fixed, then:*

- $\mathbb{E}[|E|] \simeq \frac{n^3p^2q}{2}$;
- $\mathbb{E}[d_u] \simeq (np)^2q$;
- $\mathbb{E}[c_u] \simeq \frac{q}{np}$.

Proof. We begin by noting that our hypothesis implies $np^2 \to 0$. In this case, $(1-(1-p^2q)^n) \simeq np^2q$ and, therefore, $\mathbb{E}\left[Q_{u,v}\right] \simeq np^2q$ for any u,v.

The first two statements are easily derived by using this fact, after applying the linearity of expectation to $|E_q| = \sum_{u,v} Q_{u,v}$ and $d_u = \sum_x Q_{u,x}$, respectively. For the third statement, note that

[1] $\mathsf{Be}(q)$ is a Bernoulli random variable with probability q.

[2] $\mathsf{Bi}(n,p)$ is a Binomial random variable with n trials each with probability p.

[3] Since $n \to \infty$, we frequently omit constant subtractions and replace $n-1, n-2, \ldots$ with n without comment, provided this does not affect the result.

c_u can be written as $c_u = \dfrac{\sum_{x,y} Q_{x,u}Q_{u,y}Q_{x,y}}{\sum_{x,y} Q_{x,u}Q_{u,y}} = \dfrac{N}{D}$. Using a first order Taylor expansion (assured by the concentration of N and D around their non-zero means, see [29], pages 240-245):

$$\mathbb{E}[c_u] = \frac{\mathbb{E}[N]}{\mathbb{E}[D]} + o_p\left(\frac{\mathbb{E}[N]}{\mathbb{E}[D]}\right),$$

where o_p means convergence in probability. Below we show in details that the numerator $\mathbb{E}[N]$ is asymptotically equal to $(npq)^3/2$. Analogously the denominator $\mathbb{E}[D]$ is asymptotically equal to $\mathbb{E}[D] = (n^2p^2q)^2/2$, but we omit lengthy calculations. These statements imply our result.

It is enough to determine the functions $f_1(n,p,q), f_2(n,p,q) \sim (npq)^3/2$ such that $f_1 \leq \mathbb{E}[N] \leq f_2$. An analogous procedure, which we omit, can be performed for the denominator as well.

Denote $I_{a,b,c} = I_{a,b,c}(x,y) = P_{a,u}P_{a,x}P_{b,u}P_{b,y}P_{c,x}P_{c,y}T_{u,a,x}T_{u,b,y}T_{x,c,y}$, for $a \neq u,x; b \neq u,y; c \neq x,y$. This enables us to write

$$\begin{aligned}
\mathbb{E}[N] &= \sum_{(x,y)} \mathbb{P}\left(Q_{x,u}Q_{u,y}Q_{x,y} = 1\right) \\
&= \sum_{x,y\neq u} \mathbb{P}\left(\bigoplus_{a,b,c} P_{a,u}P_{a,x}P_{b,u}P_{b,y}P_{c,x}P_{c,y}T_{u,a,x}T_{u,b,y}T_{x,c,y} = 1\right) \\
&= \sum_{x,y\neq u} \mathbb{P}\left(\bigoplus_{a,b,c} I_{a,b,c} = 1\right)
\end{aligned}$$

Note that, if $x, y \neq u$, then $\mathbb{P}\left(I_{a,b,c} = 1\right)$ can take three possible values: p^3q^3 if $a = b = c$, p^5q^5 if $a = b \neq c$ (or the two other symmetric cases), and p^6q^6 if $a \neq b \neq c \neq a$.

For the right inequality, note that $\bigoplus I_{a,b,c} = 1$ iff $\sum I_{a,b,c} \geq 1$. Union bound yields:

$$\begin{aligned}
\mathbb{E}[N] &= \sum_{x,y\neq u} \mathbb{P}\left(\sum_{a,b,c} I_{a,b,c} \geq 1\right) \\
&\leq \sum_{x,y\neq u} \sum_{a,b,c} \mathbb{P}\left(I_{a,b,c} \geq 1\right) = \sum_{x,y\neq u} \sum_{a,b,c} \mathbb{P}\left(I_{a,b,c} = 1\right) \\
&\leq \sum_{x,y\neq u} \Big(\sum_{a=b=c} \mathbb{P}\left(I_{a,a,a} = 1\right) + \sum_{\substack{a=b\neq c \\ \text{+symm cases}}} \mathbb{P}\left(I_{a,a,c} = 1\right) + \sum_{a\neq b\neq c\neq a} \mathbb{P}\left(I_{a,b,c} = 1\right) \Big) \\
&\leq \binom{n}{2}(np^3q^3 + 3n(n-1)p^5q^3 + n(n-1)(n-2)p^6q^3) \\
&\simeq \frac{(npq)^3}{2}(1 + np^2 + n^2p^3) \simeq \frac{(npq)^3}{2}
\end{aligned}$$

where, in the last equation, we use the fact that $np^2, n^2p^3 \to 0$.

Now, for the left inequality, we drop some terms from the binary sum, and manipulate a bit further to obtain

$$\begin{aligned}
\mathbb{E}[N] &= \sum_{x,y\neq u} \mathbb{P}\left(\bigoplus_{a,b,c} I_{a,b,c} = 1\right) \\
&\geq \sum_{x,y\neq u} \mathbb{P}\left(\bigoplus_{\substack{a\neq u,x,y \\ a=b=c}} I_{a,a,a} = 1\right) \\
&= \sum_{x,y\neq u} 1 - \mathbb{P}\left(\bigoplus_{\substack{a\neq u,x,y \\ a=b=c}} I_{a,a,a} = 0\right)
\end{aligned}$$

Now, note that, for $a \neq a'$, $I_{a,a,a}$ and $I_{a',a',a'}$ are independent, as they do not share any random variables. Recall that $I_{a,a,a} = \mathsf{Be}\left(p^3 q^3\right)$. Then, we have:

$$\begin{aligned}
\mathbb{E}[N] &\geq \sum_{x,y\neq u} 1 - \mathbb{P}\left(\bigoplus_{\substack{a\neq u,x,y \\ a=b=c}} I_{a,a,a} = 0\right) \\
&= \sum_{x,y\neq u} 1 - (1-p^3q^3)^{n-3} \simeq \frac{(npq)^3}{2}
\end{aligned}$$

Thus $\mathbb{E}[N] = \mathbb{E}(\sum_{x,y} Q_{x,u}Q_{u,y}Q_{x,y}) \simeq \frac{(npq)^3}{2}$

See example of the dominating case where $g = a = b = c$ in Figure 3.1.

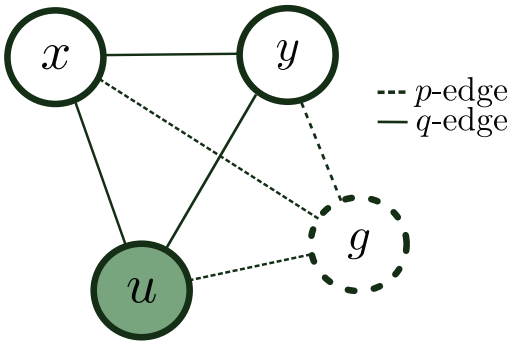

Figure 3.1 – Edges of neighborhood of u.

□

Consider for comparison an Erdős-Rényi random graph $G(n, p')$ with the same expected density. It has an edge probability $p' = np^2q$, average degree of $(np)^2q$, and its expected clustering coefficient is therefore np^2q. For the considered regime we have $n^2p^3 \to 0$ and q is fixed,

hence $c_u(G(n,p')) \ll c_u(G(n,p;q)$, precisely $np^2q \ll q/np$.

Another interesting feature of the $G(n,p;q)$ model is that, for a rather general regime of p, all edges have a very limited number of generators.

Lemma 3.2. *For $np \to \infty, n^5p^6 \to 0$ and fixed q, w.h.p., all edges have at most two generators.*

Proof. It is enough to show that the expected number of edges with three or more generators vanishes, as this implies the result by the first moment method.

Let p_k denote the probability that an arbitrary edge (u,v) has precisely k generators. Recall from Remark 1 that the generator set of an edge has size $\mathsf{Be}\left(n,p^2q\right)$. This implies

$$p_k = \binom{n}{k}(p^2q)^k(1-p^2q)^{n-k} \leq (np^2q)^k.$$

For any edge e, the probability that it has at least 3 generators is given by

$$\begin{aligned}\mathbb{P}\left(|S_e| \geq 3\}\right) &= \sum_{k\geq 3} p_k \leq \sum_{k\geq 3}(np^2q)^k \\ &= (np^2q)^3\frac{1}{(1-np^2q)} \simeq (np^2q)^3\end{aligned}$$

where the last steps follow from the convergence of the geometric series for large enough n — note that our hypothesis implies that $np^2 \to 0$. Finally,

$$\mathbb{E}\left[\{(u,v) : |S(u,v)| \geq 3\}\right] = \sum_{u,v}\mathbb{P}\left(|S_e| \geq 3\}\right) \leq \sum_{u,v}(np^2q)^3 = n^2(np^2q)^3 \simeq n^5p^6q^3 = o(1).$$

□

One observation from the proof is that the expected number of edges with two generators is $\mathbb{E}\left[\{(u,v) : |S(u,v)| = 2\}\right] = n^4p^4q^3$ that is a negligible fraction of the total number of edges.

From now on, we consider the following more restrictive regimes on p and q: $(np)^5p \to 0$, fixed q, and $npq^2 = 12\log n + \omega(1)$. These constraints imply an average degree d of $\Omega(\log^2 n)$, while still being sparse enough to allow a characterization of local structures (the common-neighbor subgraphs over all edges) that are central in the reassembly process.

3.1.2 Assembly of Noiseless Egonets

Our goal of this section is to demonstrate that for a certain regime of the parameters p and q, it is feasible to reassemble a collection of noiseless 1-hop egonets extracted from a $G(n,p;q)$ random graph. For this, we characterize a number of properties that this random graph pos-

sesses with a high probability, and these properties naturally lead to a very intuitive algorithm for reassembling the given egonets.

The intuition behind the algorithm is as follows. Let us assume for a moment that the edges of the master graph are uniquely labeled, and that this labeling is preserved through patch generation process — that is, edges in egonets that correspond to the same edge on the master graph are given the same label. In this case, it is straightforward to re-identify the nodes. For instance, if the edge (u, v) is assigned the unique label 35, then there is an edge labeled 35 in the egonet of u, which means its other endpoint must be v, since no edge to another node is assigned the label 35. Analogously, we can identify u on the egonet of v.

This observation means that the problem can be solved, as long as we can assign such a consistent labeling to edges between all egos and its respective neighbors. However, we must assign these labels by looking only at the structure of the egos and nothing else. Fortunately, under the condition that either u or v is the ego-center, any edge (u, v) has a structural feature that is preserved by the egonet extraction process. This feature is the *induced subgraph of common neighbors* of u and v, which we denote by $H_{u,v}$. Note that this feature is symmetric by nature, thus $H_{u,v} \sim H_{v,u}$ (From here $\sim$ means graph isomorphism). As the main result of this section, we show that, for a $G(n, p; q)$ random graph, any two edges have non-isomorphic subgraphs of common neighbors. Therefore this feature acts as a fingerprint for all edges in a graph.

Further we formulate the main result of this section, provide key lemmas where we show that all the edges of G have "unique edge fingerprints" across the patches and, at last, we prove the theorem by providing a particular assembly algorithm.

Theorem 3.3. *Let G be a $G(n, p; q)$ random graph, with $(np)^5 p \to 0$, fixed q and $npq^2 = 12 \log n + \omega(1)$ and let $\mathscr{P} = \{G_i\}_{i \in [n]}$ be an anonymized egonet collection extracted from G. There exists an assembly algorithm that builds $\hat{G}$ from the input $\mathscr{P}$ and $V(\hat{G}) = V(G)$ and $E(\hat{G}) = E(G)$.*

Structural Properties of Patches

In the section, we characterize a number of additional properties of the graphs generated by $G(n, p; q)$ random graph model. To determine when all edges indeed have unique (up to isomorphism) subgraphs of common neighbors, we must first characterize the structure of these subgraphs. We start by determining the node set of $H_{u,v}$, which we call $N_{u,v}$.

Lemma 3.4. *If G is sampled from $G(n, p; q)$ with $np \to \infty$, $(np)^5 p \to 0$, q is fixed, then for any fixed $u, v \in G$ such that u is adjacent to v, the following statements hold w.h.p.:*

- *For each $x \in N_{u,v}$, there exists $g \in S(u, v) \cap S(u, x) \cap S(v, x)$ – i.e., all the edges of the uxv triangle have at least one common generator;*
- $|N_{u,v}| = Bi\left(n, |S(u, v)| pq^2\right)$ *and* $\mathbb{E}\left[|N_{u,v}| \middle| |S(u, v)|\right] = |S(u, v)| npq^2$.

See Figure 3.2 for an illustration of $N_{u,v}$.

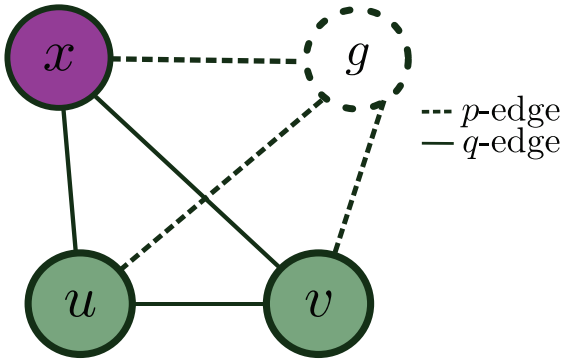

Figure 3.2 – If $x \in N_{u,v}$ is a common neighbor of u and v then all three edges $(x,u),(x,v)$ and (u,v) have a common generator g.

Proof. The proof is similar to the proof of the Proposition 3.1, we give it later in Section 3.1.7. □

We note here that the probability of a node x to be connected to u,v is $(1+o(1))n^2p^4q^2$ (see 3.1.7), thus summing through all possible subgraphs $H_{u,v}$ there is a negligible fraction of nodes that are connected to some u,v through other generators than $S(u,v)$. These nodes do not affect further computations, thus we omit these for simplicity.

We now characterize edges between the nodes of the neighborhoods $N_{u,v}$. For any $x,y \in N_{u,v}$, by Lemmas 3.2 and 3.4 there exist $g_1,g_2 \in S(u,v)$ such that $P_{g_1,x}=1$ and $P_{g_2,y}=1$. Consider two cases:

$$\begin{cases} g_1=g_2=g, & \text{then } xy\in E \text{ if } T_{xgy}=1 \text{ or if } P_{g_3,x}P_{g_3,y}T_{xg_3y}=1 \text{ for some } g_3.\\ & \text{Hence } \mathbb{P}\left(Q_{x,y}=1 | x,y\in N_{u,v}, g_1=g_2\right)=q+np^2q.\\ g_1\neq g_2, & \text{then } xy\in E \text{ if } P_{g_1,y}T_{xg_1y}=1 \text{ or if } P_{g_2,x}T_{xg_2y}=1 \text{ or if } P_{g_3,x}P_{g_3,y}T_{xg_3y}=1 \text{ for some } g_3.\\ & \text{Hence } \mathbb{P}\left(Q_{x,y}=1 | x,y\in N_{u,v}, g_1\neq g_2\right)=2pq+np^2q. \end{cases}$$

Corollary 3.5. *Under the conditions of Lemma 3.4, w.h.p., one of these cases holds:*

1. $|S(u,v)|=1$ *and* $H_{u,v}$ *is a single Erdős-Rényi graph* $G(Bi\left(n,pq^2\right),q)$*;*
2. $|S(u,v)|=2$ *and* $H_{u,v}$ *consists of two components, each is an Erdős-Rényi graph* $G(Bi\left(n,pq^2\right),q)$ *and edges between nodes of different components exist with probability* np^2q*.*

We note that the exact probability of an edge in the considered Erdős-Rényi graph is $q+np^2q$; however we omit the negligible part for readability, since it does not affect further computations. Also, we observe, as $np^2q \ll q$ in the latter case, that the two Erdős-Rényi

graphs have dense structure, but are very loosely connected. See Figure 3.3, for the example of a subgraph $H_{u,v}$ induced by common neighbors of an edge (u,v) with one generator g .

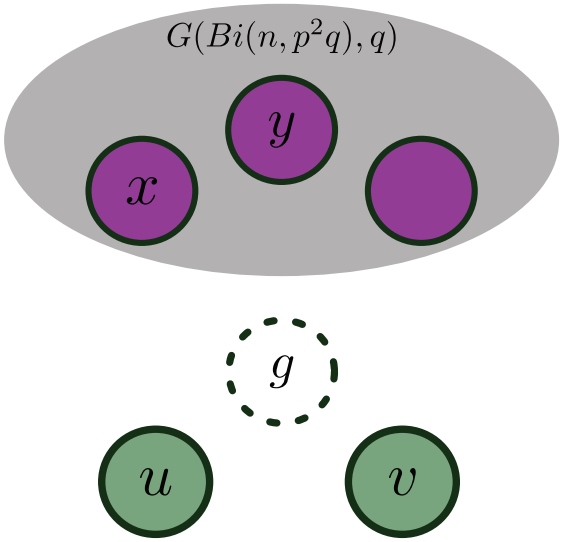

Figure 3.3 – A subgraph $H_{u,v}$ induced by common neighbors of an edge (u,v) with one generator g.

Uniqueness of Edge Fingerprints

We are now ready to prove our key result of this section about uniqueness of common-neighbors subgraphs.

Theorem 3.6. *Let G be a $G(n,p;q)$ random graph, with $(np)^5 p \to 0$, fixed q and $npq^2 = 12\log n + \omega(1)$. Then, w.h.p., for any pairwise adjacent nodes u,v and i,j, either $\{u,v\} = \{i,j\}$ or $H_{u,v}$ is not isomorphic to $H_{i,j}$.*

Proof. Denote by W the number of quadruples (u,v,i,j), with u and v adjacent, i and j are adjacent and $(i,j) \neq (u,v),(v,u)$, such that $H_{u,v}$ is isomorphic to $H_{i,j}$. By the first moment method, it is enough to show that $\mathbb{E}[W] \to 0$. We note that $\mathbb{E}[W] = \sum_{u,v,i,j} \mathbb{P}\left(H_{u,v} \sim H_{i,j}\right)$.

Now, we fix u,v,i and j and split our analysis into several cases. We consider the most complex case in detail and omit lengthy and similar computations for the other cases.

1. $|S(u,v)| = |S(i,j)| = 1$:
   (a) $S(u,v) = S(i,j) = \{g\}$, where g is the common generator of (u,v) and (i,j):
      i. $u = i$ and $v \neq j$ (or, analogously, $u \neq i$ and $v = j$):
      Note that, by Lemma 3.4 any vertex $x \in V$ belongs to both $H_{u,v}$ and $H_{u,j}$ according to the following criteria:

- $x \in H_{u,v}$ iff $P_{g,x}T_{u,g,x}T_{v,g,x} = 1$ (w.p pq^2)
- $x \in H_{u,j}$ iff $P_{g,x}T_{u,g,x}T_{j,g,x} = 1$ (w.p pq^2)[4]

Let J be a common super-graph of $H_{u,v}$ and $H_{u,j}$, such that it is induced by $\{x \in V : P_{g,x}T_{u,g,x} = 1\}$. By the definition, any node that is not in J cannot belong to neither $H_{u,v}$ or $H_{u,j}$. Note that, by reasoning analogous to those in Lemma 3.4 and in Corollary 3.5 the graph J is an Erdős-Rényi random graph $G(\mathsf{Bi}(n,pq),q)$ (each node $x \in V$ satisfies $P_{g,x}T_{u,g,x} = 1$ independently with probability pq; and any two nodes $x, y \in J$ are adjacent if and only if $T_{x,g,y} = 1$, that happens with probability q, for all pairs independently).
Furthermore, each node $x \in J$ belongs to $H_{u,v}$ or $H_{u,j}$ if $T_{v,g,x} = 1$ or $T_{j,g,x} = 1$, and these conditions hold independently with probability q. This means $H_{u,v}$ and $H_{u,j}$ are obtained from J by sampling each node independently with probability q. To bound the probability of these two graphs being isomorphic, we use the Lemma 3.10 (see Section 3.1.7).
We put $m = \mathsf{Bi}(n,pq)$ and $t = q$ into the Lemma 3.10 and fix $0 < \delta < 1$ and obtain:

$$\mathbb{P}\left(H_{u,v} \sim H_{u,j}\right) \leq \exp\left(m\log m - m^2 c\right) + 2\exp\left(-\frac{\delta^2 mq}{2}\right)$$

for $c = (1-\delta)^2 q^2(1-q)\log c_1$ and $c_1 = (q^2 + (1-q)^2)^{-1} \in (1,2]$. Note, however, that m is a random variable. We can apply the Chernoff bound (see Appendix A.1) to bound $npq(1-\delta) \leq m \leq npq(1+\delta)$. Thus:

$$\mathbb{P}\left(H_{u,v} \sim H_{i,j}\right) \leq \exp\left(npq(1+\delta)\log npq(1+\delta) - (npq)^2(1-\delta)^2 c\right) + 4\exp\left(-\frac{\delta^2 npq^2}{3}\right).$$

There is a negligible fraction of nodes that is connected to u, v with probability $(np^2q)^2$ through other generators, however we ignore these since only a negligible fraction of edges are adjacent to these nodes.

ii. $u \neq i$ and $v \neq j$. The case is analogous to the previous case. We observe that both $H_{u,v}$ and $H_{i,j}$ are obtained by node sampling the graph $J' = N_s^p$; where J' is an Erdős-Rényi random graph $G(\mathsf{Bi}(n,p),q)$ and each node is sampled with probability q^2 to $H_{u,v}$ and $H_{i,j}$. We apply again Lemma 3.10 (see Section 3.1.7):

$$\mathbb{P}\left(H_{u,v} \sim H_{i,j}\right) \leq \exp\left(npq^2(1+\delta)\log npq^2(1+\delta) - (npq)^2(1-\delta)^2 c\right) + 4\exp\left(-\frac{\delta^2 npq^2}{3}\right).$$

[4]The probability of x being connected to u, v through other generator(s) g' is $o(pq^2)$.

(b) $S(u,v) \neq S(i,j)$. Denote $S(u,v) = \{g_1\}$ and $S(i,j) = \{g_2\}$. Since $N^p_{g_1} \cap N^p_{g_2} = \emptyset$ holds w.h.p, it also holds that $N_{u,v} \cap N_{j,i} = \emptyset$ (by Lemma 3.4). Then, by a reasoning similar to that in Lemma 3.10, $\mathbb{P}\left(H_{u,v} \sim H_{i,j}\right) \leq m!(q^2 + (1-q)^2)^{\binom{m}{2}}$.

2. $|S(u,v)| = 2$. Denote $S(u,v) = \{g_1, g_2\}$. In this case, $H_{u,v}$ consists of two weakly connected Erdős-Rényi graphs $H^1_{u,v} \cup H^2_{u,v}$, and similarly for $H_{i,j}$. We can separate these components by using the fact that the number of edges between components is at most constant, while each node inside the component has $\omega(1)$ degree. Thus we can scan through all such combinations of edges to identify two components. Then we can consider isomorphism of each of the components and reduce the problem to the previous case. Thus, we stochastically upper-bound this case by the previous case.

Using the loosest bound of the previous cases, we can bound $\mathbb{E}[W]$:

$$\begin{aligned}\mathbb{E}[W] &\leq n^4\left(\exp\left(npq(1+\delta)\log npq(1+\delta) - (npq)^2(1-\delta)^2 c\right) + 4\exp\left(-\frac{\delta^2 npq^2}{3}\right)\right)\\ &= \exp\left(4\log n + npq(1+\delta)\log npq(1+\delta) - (npq)^2(1-\delta)^2 c\right)\\ &\quad + 4\exp\left(4\log n - \frac{\delta^2 npq^2}{3}\right)\end{aligned}$$

the last summand dominates and thus the whole sum goes to 0 if $p \geq \frac{12\log n + \omega(1)}{\delta^2 q^2 n}$. Note that if $q \in (0,1)$ and $\delta < 1$, c is constant. □

3.1.3 Feasibility of Egonet Assembly

The results leading to Theorem 3.6 enable us to analyze a simple assembly algorithm that works as follows. Let $\mathscr{P} = \{G_i\}_{i\in[n]}$ be an unlabeled egonet collection of a graph $G = (V,E)$, that is the master-graph we want to obtain at the end of the assembly process; and also assume that all edges in G have unique fingerprints, that is, for any two distinct edges $(u,v) \in E$ and $(i,j) \in E$, the subgraphs $H_{u,v}$ and $H_{i,j}$ are not isomorphic.

G must have $[n]$, the index set of $\mathscr{P} = \{G_i\}_{i\in[n]}$, as its node set, so we begin by setting $[n]$ as the vertex set of our assembled graph $\hat{G}$. To construct its edge set $E(\hat{G})$, we choose a node $u \in [n]$. We know that u is present in the egonet G_u and has been assigned the label 1 in G_u. Take a node $j \in G_u$ other than 1. Edge $(1,j)$ is the image of some edge (u,v) in G, and the subgraph of G_u induced by 1, j and their common neighbors is a relabeled version of $H_{u,v}$. Extract this fingerprint from G_u and search for a second edge, in a different egonet, with an isomorphic fingerprint. Since the fingerprints of the edges in G are unique, there will be exactly one such edge, say $(1,k)$ on the egonet $G_{u'}$, and both of them must have originated from the same edge on the master graph. The labels of this edge must be the egonet centers u and u' of the two matching edges, so we add the edge (u,u') to $E(\hat{G})$. In this way we defined $a_u(j) = u'$ and $a'_u(k) = u$. Repeat this for all egonets until they are exhausted, at which point the algorithm

terminates. We will call this the *fingerprint assembly algorithm*.

If all edges in G have unique fingerprints, this algorithm will always reassemble G correctly:

Proof. [Theorem 3.3]

By the generation process of the patch collection, every edge (u, v) in the master graph is center-incident in exactly two patches: the one centered at u and the one at v. It is straightforward to see that if we can correctly collapse all the center-incident edges in all the patches (and ignoring all the other edges), this reveals the original graph G. Therefore, if the edge fingerprint given by the isomorphism class of $H_{u,v}$ is unique for each (u, v), fingerprint assembly succeeds.

□

Note that the fingerprint assembly algorithm requires $\binom{|E|}{2}$ checks for small-graphs isomorphism. This is, in general, a computationally expensive procedure even after recent improvements, with the best known algorithm having quasi-polynomial time complexity [13]. With an oracle for the graph isomorphism problem, the average case complexity of this algorithm drops to around $|E|(npq^2) + |E|^2$, from the subgraph extraction process and the checks for graph isomorphism, respectively. Any technique for optimizing the graph isomorphism step, such as applying approximate graph isomorphism techniques, can be used to reduce its running time. Additionally, if this step is solved by constructing an isomorphism whenever possible, one can use the information given by this isomorphism to further reduce the number of fingerprints comparisons.

Our implementation of this algorithm uses canonical labeling methods to check for subgraph isomorphism. A *canonical labeling* is a labeling of the graph's vertices that uniquely captures the structure of the graph, and two graphs are isomorphic if and only if their canonical forms are precisely equal. The problems of canonization and isomorphism are similar in both theory and algorithm design, even though it is not known whether they are poly-time equivalent [9].

Our implementation has an additional optimization step: Instead of searching through all edges in the egonets looking for edges with isomorphic fingerprint, we convert each fingerprint to an integer value. These integers are extracted from the canonical form of the fingerprint and are therefore graph invariants. Afterwards these edges are stored in a hash map where we use the corresponding integer fingerprints as the search key; thus, reducing the pairwise search for isomorphic fingerprints to a scan over the hash map for edges with matching keys. This optimization reduces the algorithm complexity from $\binom{|E|}{2}$ checks for isomorphism to $|E|$ calculations of canonical forms, at the cost of $|E|$ additional graph-to-integer conversions. Although eventual hash collisions can in principle insert noise in our fingerprint comparison, we do not expect such collisions to be frequent. Additional graph invariants,

such as number of edges, can also be extracted from the fingerprints to disambiguate even further in case of eventual collisions, but we choose not to use them in our implementation.

We implement the algorithm by using the canonical labeling procedure from the Bliss library [56]. This library provides us with a hash calculation procedure, that we use to convert fingerprints to integer values. Additional collisions can result from this, and the same mitigation techniques described previously (number of edges and degree sequences) can also be applied here. We ran a set of experiments for finite graphs sampled from $G(n,p;q)$ model and found out that the algorithm can restore all the edges with precision 1.

We do not focus on developing the most efficient algorithm in this section, hence we do not set up an extensive experiment set with different theoretical and practical models. One of the interesting future directions would be to consider real and artificial noisy data-models and to develop an approximate assembly algorithm for this. Here we are more interested in the feasibility of graph reconstruction from very poor additional information. And the experiments fully support the theoretical results: for graphs sampled from the $G(n,p;q)$ model the edge fingerprints are unique, thus assuring the feasibility of assembly.

3.1.4 Assembly of Noisy Egonets

Recall that in the noisy case our goal is to reconstruct a master graph $\hat{G}$ from the noisy egonets collection $\mathscr{P} = \{G_i = f_i(H_i^*)\}$, where we generate a noisy observation H_i^* by keeping the original node set of H_i but sampling edges independently with probability s and anonymizing the obtained graph with a function f_i.

We cannot rely anymore on the isomorphism of the subgraphs of common neighbors, however we hypothesize that the common neighbors still preserve enough of the structure to be identifiable. In order to show that the hypothesis is true under certain conditions, we prove a result analogous to Theorem 3.6. This result can be expressed in terms of the *edge mismatch* between two graphs.

Definition 3.7 (Edge Mismatch)**.** *Let $H_1(V_1,E_1)$ and $H_2(V_2,E_2)$ be two graphs with $|V_1| = |V_2|$ and let π be a bijection between V_1 and V_2. The* edge mismatch *of π of H_1 and H_2, denoted by $\Delta(H_1,H_2,\pi)$, is given by:* $\Delta(H_1,H_2,\pi) = \sum_{(u,v)\in\binom{V_1}{2}} \mathbf{1}_{\{(u,v)\in E_1 \otimes (\pi(u),\pi(v))\in E_2\}}$

This metric is equivalent to the one from Definition 2.1 with $\alpha = 0$ and conditioning on all the nodes being matched.

Furthermore, for two neighboring nodes u and v, we denote by $H_{u,v}$ the subgraph of G_u induced by the common neighbors of u and v. Note that $H_{u,v}$ and $H_{v,u}$ have the same node sets[5]. However, since both G_u and G_v are noisy egonet-observations, it does not hold in general that $H_{u,v} = H_{v,u}$, which differs from the noiseless case. Rather, $H_{u,v}$ and $H_{v,u}$ are

[5] In the regimes further considered, these graphs remain connected.

both subgraphs of $H_{u,v}$. We also note that the notation is not symmetric anymore meaning that $H_{u,v}$ is a subgraph induced by an egocenter u and its neighbor v, i.e., the first index denotes the center of the egonet.

Lemma 3.8. *Let G be a $G(n,p;q)$ random graph, with $(np)^5 p \to 0$, fixed q and $npq^2 = \frac{32\log n + 16\log(npq^2)+\omega(1)}{s^3}$. Then, w.h.p., for any pairwise-adjacent nodes u,v and i,j, either $\{u,v\} = \{i,j\}$ or $\Delta(H_{u,v}, H_{v,u}, \pi_0) < \Delta(H_{u,v}, H_{i,j}, \pi)$ for any bijection π (with π_0 the identity mapping over N_{uv}).*

Proof. The proof is very similar to the one of Theorem 3.6, we explain the main steps. Analogously to Theorem 3.6, denote by W the number of quadruples (u,v,i,j), with u and v adjacent, i and j are adjacent and $(i,j) \neq (u,v),(v,u)$, such that $\Delta(H_{u,v}, H_{v,u}, \pi_0) \geq \Delta(H_{u,v}, H_{i,j}, \pi)$ for some bijection π. We show that $\mathbb{E}[W] \to 0$.

Fix u, v, i and j and consider only the case $S(u,v) = S(i,j) = \{g\}$, $u \neq i$ and $v = j$ — other cases as broken down in the proof of Theorem 3.6 will be omitted, as they yield stricter bounds. Our goal is to show that $\mathbb{P}\left(\Delta(H_{u,v}, H_{v,u}, \pi_0) \geq \Delta(H_{u,v}, H_{i,v}, \pi) \text{ for some } \pi\right) \to 0$.

Denote $J = \{x \in V \text{ s.t. } P_{g,x} T_{u,g,x} = 1\}$ a common noiseless super-graph of $H_{u,v}$ and $H_{i,v}$. For every node x in J it belongs to $H_{u,v}, H_{i,v}$ independently with probability q. Furthermore, all edges between nodes in these sets will show up in the corresponding graphs with probability s, independently. Thus, all three graphs can be seen as a result of two-step sampling process similar to the $BiG(n,p;t,s)$ model where the master-graph is a subgraph J. See details in Section 3.1.7 and Figure 3.4 for an illustration of this process.

We further assume that $|H_{u,v}| = |H_{i,v}| = m$, as otherwise there are no bijections between the node sets of these graphs and the quadruple is not counted in W by default.

We put $J = G(m,q)$ and $t = q$ into the Lemma 3.11 and fix $0 < \delta < 1$ and obtain:

$$\mathbb{P}\left(\text{Exists } \pi : \Delta(H_{u,v}, H_{v,u}, \pi_0) > \Delta(H_{u,v}, H_{u,j}, \pi)\right) \leq 2 \sum_{k=2}^{\infty} m^k \exp\left(k\left(\log m - \frac{mps^3}{16}\right)\right) + \exp\left(-\frac{\delta^2 m(1-q)}{2}\right)$$

where m^k is a usual upper bound for a number of matchings with k nodes mapped wrongly. Recall that, by Lemma 3.4, $m = |N_{u,v}| = \mathsf{Bi}\left(n, pq^2\right)$. The Chernoff bound (see A.1) implies

$(1-\delta)npq^2 \le m \le (1+\delta)npq^2$, so summing over all quadruples yields

$$\begin{aligned}\mathbb{E}[W] &\le \sum_{u,v,i,j}\left[2\sum_{k=2}^{\infty}\exp\left(k\left(\log npq^2(1+\delta)-\frac{npq^2s^3(1-\delta)}{16}\right)\right)+3\exp\left(-\frac{\delta^2npq^2(1-q)}{3}\right)\right]\\ &\le 2\sum_{k=2}^{\infty}\exp\left(4\log n+k\left(\log npq^2(1+\delta)-\frac{npq^2s^3(1-\delta)}{16}\right)\right)\\ &\quad+3\exp\left(4\log n-\frac{\delta^2npq^2(1-q)}{3}\right)\end{aligned}$$

Under the assumptions, the right side vanishes, which concludes the proof. □

3.1.5 Feasibility of Noisy Egonets Assembly

Based on Lemma 3.8 we build a variation of the fingerprint assembly algorithm that can be used to assemble a collection of noisy egonets $\mathscr{P} = \{G_i\}_{i\in[n]}$. The *noisy-fingerprint algorithm* takes $\{G_i\}$ as input and proceeds like the fingerprint assembly algorithm, except for the following modification: For each egonet G_u and each node $j \neq 1$ in G_v, we match it to an edge $(1,k)$ on an egonet $G'_{u'}$, but we change the criteria "both fingerprints match exactly" to the criteria "edge mismatch between both fingerprints is minimized".

Just like in the noiseless scenario, this algorithm is able to completely assemble the master-graph G.

Theorem 3.9. *Let G be a $G(n,p;q)$ random graph, with $(np)^5p \to 0$, fixed q and $npq^2 = \frac{32\log n+16\log(npq^2)+\omega(1)}{s^3}$ and let $\mathscr{P} = \{G_i\}_{i\in[n]}$ be an unlabeled noisy-egonet collection extracted from G. If $\hat{G}$ is the output graph of the noisy-fingerprint algorithm with input $\mathscr{P}$, then $E(\hat{G}) = E(G)$ w.h.p.*

Proof. The proof proceeds exactly as in the noiseless case, except that the test of isomorphism of the common-neighbor subgraph is replaced with the test for a minimal edge mismatch distance.

□

Unlike the fingerprint-assembly algorithm, the noisy-fingerprint algorithm has no trivial efficient implementation. The main reason is that the subgraph isomorphism subroutine must be replaced by the calculation of a minimum inconsistency between the input subgraphs, which is a computationally expensive task to which there is no known efficient approximation, to the best of our knowledge. A practical approximation that warrants some interest is to use the optimized form of the algorithm that has been implemented with the Bliss library, but use a locality-sensitive hash function over labeled graphs to store all subgraphs in our hash map. This way, the task of searching for graphs with similar topology (i.e., similar

fingerprints) would be reduced to determining entries that are closely located in this hash map.

3.1.6 Discussion

We stated two results that characterize the regimes where complete graph assembly, using only the structure of very small, unlabeled patches is feasible. We have shown how the relatively high transitivity (compared to Erdős-Rényi) of the $G(n,p;q)$ random graph model leads to the existence of features in egonet patches, which can be exploited in the assembly of the egonet collection through a very simple algorithm.

We have shown that an assembly is still feasible if the patches in our collection are noisy observations of egonets. The conditions required on the model's parameters are stronger but only slightly: the lower bounds imposed on the average degree of the intermediate graph G_p differ by a multiplicative penalty of s^{-3} due to the noise parameter s (i.e., s^{-6} in terms of the average degree of G).

It is important to highlight that the focus of this section is not chiefly on particular algorithms for solving the graph assembly problem. Rather, we evaluate the impact of a fundamental network property — its clustering coefficient — on the theoretical feasibility of solving the graph assembly problem. Nevertheless, our approach of generating unique fingerprints for edges, based on their common-neighbor subgraphs, may be relevant for network assembly in practice. We consider a realistic algorithm in the following section.

3.1.7 Auxiliary Results

Proof of Lemma 3.4

Proof. To prove the first statement, it is enough to show that, under stated assumptions and given u and v are adjacent, the expected size of the set $X = \{x \in V : Q_{u,x}Q_{v,x} = 1, S(u,x) \cap S(v,x) \cap S(u,v) = \emptyset\}$ goes to 0. The first moment method then implies that this set has size 0 a.a.s., which is equivalent to the desired result.

We start by using the tower property:

$$\mathbb{E}\left[|X| \;\middle|\; Q_{u,v} = 1\right] = \mathbb{E}\left[\mathbb{E}\left[|X| \;\middle|\; N_u^p, N_v^p, S(u,v), Q_{u,v} = 1\right] \;\middle|\; Q_{u,v} = 1\right]$$

Recall that, for any node g, N_g^p denotes the set of p-neighbors of g that showed up during the construction process of the $G(n,p;q)$ graph. Note that the condition on the inner expectation can be expressed as a function only of random variables of the kinds P_{u*}, P_{v*} and T_{u*v}. By construction, any functions of random variables other than these are independent of this condition.

To bound this inner expectation, fix vertex sets $\mathscr{U}, \mathscr{V}, \mathscr{S}$ with $\mathscr{S} \subseteq \mathscr{U} \cap \mathscr{V}$ and denote current values being fixed event by $\mathscr{T} = \{N_u^p = \mathscr{U}, N_v^p = \mathscr{V}, S(u,v) = \mathscr{S}, Q_{u,v} = 1\}$. Then

$$\mathbb{E}[|X| \,|\, \mathscr{T}] = \sum_{x \notin \{u,v\}} \mathbb{P}(x \in X \,|\, \mathscr{T})$$

Note that $u, v \notin X$, as $S(u,u), S(v,v) = \emptyset$ by construction. For any $x \notin \{u,v\}$, we write

$$\begin{aligned}\mathbb{P}(x \in X \mid \mathscr{T}) &= \mathbb{P}(|S(u,x)| \geq 1, |S(v,x)| \geq 1, S(u,x) \cap S(v,x) \cap S(u,v) = \emptyset \mid \mathscr{T}) \\ &= \mathbb{P}(|S(u,x)| \geq 1, |S(v,x)| \geq 1, S(u,x) \cap S(v,x) \cap \mathscr{S} = \emptyset \mid \mathscr{T})\end{aligned}$$

Rewriting expressions on $S(u,x)$ and $S(v,x)$ in terms of our basic random variables:

$$\begin{aligned}\mathbb{P}(x \in X \mid \mathscr{T}) = \mathbb{P}\Bigg(&\bigoplus_{g' \neq u,x} P_{g',u} P_{g',x} T_{u,g',x} = 1, \bigoplus_{g'' \neq v,x} P_{g'',v} P_{g'',x} T_{v,g'',x} = 1, \\ &\bigotimes_{g \in \mathscr{S}} P_{u,g} P_{v,g} P_{x,g} T_{u,g,x} T_{v,g,x} = 0 \,\Bigg|\, \mathscr{T}\Bigg) \\ = \mathbb{P}\Bigg(&\bigoplus_{\substack{g' \neq u,x \\ g'' \neq v,x}} \big(P_{g',u} P_{g',x} T_{u,g',x} P_{g'',v} P_{g'',x} T_{v,g'',x} = 1, \\ &\bigotimes_{g \in \mathscr{S}} P_{x,g} T_{u,g,x} T_{v,g,x} = 0\big) \,\Bigg|\, \mathscr{T}\Bigg)\end{aligned}$$

where the last step used the condition that $S(u,v) = \mathscr{S}$ and, therefore, $P_{u,g} P_{v,g} = 1$ for any $g \in \mathscr{S}$. Now, we can apply union bound to the latest expression:

$$\begin{aligned}\mathbb{P}(x \in X \mid \mathscr{T}) \leq \sum_{\substack{g' \neq u,x \\ g'' \neq v,x}} \mathbb{P}\Big(&P_{g',u} P_{g',x} T_{u,g',x} P_{g'',v} P_{g'',x} T_{v,g'',x} = 1, \\ &\bigotimes_{g \in \mathscr{S}} P_{x,g} T_{u,g,x} T_{v,g,x} = 0 \,\Bigg|\, \mathscr{T}\Bigg)\end{aligned}$$

The summand has different values depending on g', g''. Let us detail all possible cases:

1. $g' = g'' \in \mathscr{S}$ — since the condition implies $P_{g',u} P_{g',v} T_{u,g',v} = 1$, the event expression reduces to $(P_{g',x} T_{u,g',x} T_{v,g',x} = 1, \otimes_{g \in \mathscr{S}} P_{x,g} T_{u,g,x} T_{v,g,x} = 0)$; the two parts of the expression are mutually exclusive, hence, the event has probability 0;

2. $g' = g'' \in \mathscr{U} \cap \mathscr{V} \setminus \mathscr{S}$ — by the same argument as the previous item, the event expression reduces to $(P_{g',x} T_{u,g',x} T_{v,g',x} = 1, \otimes_{g \in \mathscr{S}} P_{x,g} T_{u,g,x} T_{v,g,x} = 0)$; using independence:

$$\begin{aligned}
&\mathbb{P}\left(P_{g',u}P_{g',x}T_{u,g',x}P_{g'',v}P_{g'',x}T_{v,g'',x}=1,\bigotimes_{g\in\mathscr{S}}P_{x,g}T_{u,g,x}T_{v,g,x}=0\;\middle|\;\mathscr{T}\right)\\
&\quad=\mathbb{P}\left(P_{g'x}T_{u,g',x}T_{v,g',x}=1,\bigotimes_{g\in\mathscr{S}}P_{x,g}T_{u,g,x}T_{v,g,x}=0\;\middle|\;\mathscr{T}\right)\\
&\quad\leq\mathbb{P}\left(P_{g'x}T_{u,g',x}T_{v,g',x}=1\;\middle|\;\mathscr{T}\right)\\
&\quad=\mathbb{P}\left(P_{g'x}T_{u,g',x}T_{v,g',x}=1\right)\\
&\quad=pq^2
\end{aligned}$$

This pattern of manipulation also applies to following cases and will be further omitted;

3. $g'=g''\notin\mathscr{U}\cap\mathscr{V}$ — in this case, either $g'\notin\mathscr{U}$, which implies $P_{g',u}=0$, or $g''\notin\mathscr{V}$, which implies $P_{g'',v}=0$; both facts imply that $P_{g',u}P_{g',x}T_{u,g',x}P_{g'',v}P_{g'',x}T_{v,g'',x}=0$ so the event has probability 0;

4. $g'\neq g'', g'\in\mathscr{U}, g''\in\mathscr{V}$ — the expression reduces to

$$P_{g',x}T_{u,g',x}P_{g'',x}T_{v,g',x}=1,\otimes_{g\in\mathscr{S}}P_{x,g}T_{u,g,x}T_{v,g,x}=0,$$

similarly to case 2, and the probability is bounded by p^2q^2;

5. $g'\neq g'', (g'\notin\mathscr{U}$ or $g''\notin\mathscr{V})$ — as in case 3, the choices of g' and g'' imply that

$$P_{g',u}P_{g',x}T_{u,g',x}P_{g'',v}P_{g'',x}T_{v,g'',x}=0$$

yielding an event of probability 0;

Case 2 will happen for $|\mathscr{U}\cap\mathscr{V}|$ choices of g' and g'', and case 4 will happen for $|\mathscr{U}|\cdot|\mathscr{V}|-|\mathscr{U}\cap\mathscr{V}|$ such choices. Thus

$$\begin{aligned}
\mathbb{P}(x\in X\mid\mathscr{T})&\leq|\mathscr{U}\cap\mathscr{V}|pq^2+(|\mathscr{U}|\cdot|\mathscr{V}|-|\mathscr{U}\cap\mathscr{V}|)p^2q^2\\
&=|\mathscr{U}\cap\mathscr{V}|p(1-p)q^2+|\mathscr{U}|\cdot|\mathscr{V}|p^2q^2
\end{aligned}$$

Since this is valid for any $x\neq u,v$

$$\mathbb{E}\left[|X|\;\middle|\;\mathscr{T},Q_{u,v}=1\right]\leq\binom{n}{2}(|\mathscr{U}\cap\mathscr{V}|p(1-p)q^2+|\mathscr{U}|\cdot|\mathscr{V}|p^2q^2)$$

and,

$$\mathbb{E}\left[|X|\;\middle|\;Q_{u,v}=1\right]=\mathbb{E}\left[(n-2)(|N_u^p\cap N_v^p|p(1-p)q^2+|N_u^p|\cdot|N_v^p|p^2q^2)\right]$$

By linearity of expectation and independence of N_u^p and N_v^p,

$$\begin{aligned}\mathbb{E}\left[|X| \;\middle|\; Q_{u,v}=1\right] &= (n-2)\left(\mathbb{E}\left[|N_u^p \cap N_v^p|\right] p(1-p)q^2 + \mathbb{E}\left[|N_u^p|\right]\cdot\mathbb{E}\left[|N_v^p|\right] p^2q^2\right)\\ &\simeq n\cdot np^2\cdot pq^2 + (np)^2p^2q^2\\ &= np^3q^2 + n^2p^4q^2 = o(1)\end{aligned}$$

To show the second and third statements of the lemma, we use the following argument, for any two events A and B, such that $A \subseteq B$ the following holds:

$$\mathbb{P}(A) = \mathbb{P}(B)\frac{\mathbb{P}(A\cap B)}{\mathbb{P}(B)} = \mathbb{P}(B)\,\mathbb{P}(A|B)\,.$$

Hence,

$$\mathbb{P}\left(Q_{x,u}Q_{x,v}=1 \;\middle|\; Q_{u,v}=1\right) = \mathbb{P}\left(x\in\bigcup_{g\in S(u,v)} N_g^p\right)\cdot\mathbb{P}\left(Q_{x,u}Q_{x,v}\;\middle|\; Q_{u,v}=1\wedge x\in\bigcup_{g\in S(u,v)} N_g^p\right).$$

Thus $|N_{u,v}| = \mathsf{Bi}\left(n, |S(u,v)|pq^2\right)$ and $\mathbb{E}\left[|N_{u,v}| \;\middle|\; S(u,v)\right] = n|S(u,v)|pq^2$. □

Graph Alignment Results

In this section we present several alignment results for different sampling models, that are modifications of the $BiG(n,p;t,s)$ model.

Node Sampling

Let $G(V,E)$ be a realization of an Erdős-Rényi random graph $G(m,p)$, and let $G_1(V_1,E_1)$ and $G_2(V_2,E_2)$ be two samples of G obtained as follows: Each node $u\in V$ is sampled with probability t independently to V_1 and V_2, and E_1 and E_2 are all edges of E whose both endpoints are sampled in V_1 and V_2, respectively.

Lemma 3.10.

$$\mathbb{P}(G_1\sim G_2) \le \exp\left(m\log m - cm^2\right) + 2\exp-\frac{\delta^2 mt}{2}$$

where $c(p,t,\delta) = (1-\delta)^2t^2(1-t)\log\left((p^2+(1-p)^2)^{-1}\right)$ *and* $0<\delta<1$.

Proof. If $|V_1|\neq|V_2|$, this event has probability 0, so we assume $|V_1|=|V_2|=m'$. Denote by V_0 the set of nodes in G that are sampled in both V_1 and V_2, and let $m_1 = |V_1\setminus V_0| = |V_2\setminus V_0|$.

Consider an arbitrary mapping $\pi: V_1\to V_2$. For any pair of nodes $x\in V_1\setminus V_0, y\in V_1$, if π is an isomorphism, then either $(x,y)\in E_1$ and $(\pi(x),\pi(y))\in E_2$, or $(x,y)\notin E_1$ and $(\pi(x),\pi(y))\notin E_2$.

This happens with probability $p^2 + (1-p)^2$, since x is not a fixed point of π. In total, we have approx $m'm_1$ such pairs, and the event above happens independently for each pair, hence $\mathbb{P}(G_1 \sim_\pi G_2) \leq (p^2 + (1-p)^2)^{m'm_1}$.

Denote by $c_1 = (p^2 + (1-p)^2)^{-1}$. In total we have at most $m'!$ mappings from G_1 to G_2, thus

$$\mathbb{P}(G_1 \sim G_2) \leq m'!(c_1)^{-m'm_1} \leq \exp\left(m' \log m' - m'm_1 \log c_1\right)$$

Recall that $m' = \mathsf{Bi}(m,t)$ and $m_1 = \mathsf{Bi}(m, t(1-t))$. Then, $\mathbb{P}\left(m' \leq (1-\delta)mt\right) \leq \exp\left(-\frac{\delta^2 mt}{2}\right)$ by Chernoff bound since $m \to \infty$, and similarly $\mathbb{P}(m_1 \leq (1-\delta)mt(1-t)) \leq \exp\left(-\frac{\delta^2 mt}{2}\right)$. Therefore,

$$\mathbb{P}(G_1 \sim G_2) \leq \exp\left(m \log m - cm^2\right) + 2\exp\left(-\frac{\delta^2 mt}{2}\right),$$

where $c = (1-\delta)^2 t^2 (1-t) \log c_1$. □

Node-Edge Sampling (Noisy Egonets Assembly)

Consider now the following variation of this graph sampling process. First, graphs G, G_1 and G_2 are generated as previously described. Now, graphs $G_{1,1} = (V_1, E_{1,1})$, $G_{1,2} = (V_1, E_{1,2})$ are obtained by sampling edges from E_1 independently with probability s, this sampling also being independent for $G_{1,1}$ and $G_{1,2}$. Similarly, $G_{2,1} = (V_2, E_{2,1})$ is obtained via this edge-sampling process from G_2. This process is illustrated in Figure 3.4.

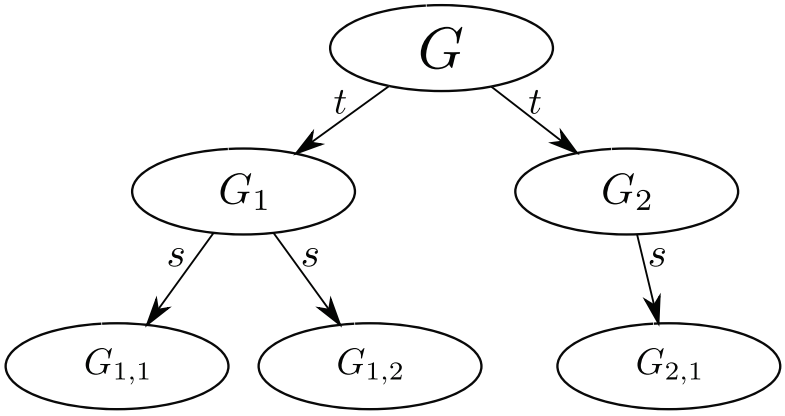

Figure 3.4 – Node-edge sampling process that generates the edge neighborhoods.

Assume $|V_1| = |V_2| = m$, and denote by π_0 the identity mapping over V_1. Denote by D an event that there exists π such that $\Delta(G_{1,1}, G_{1,2}, \pi_0) > \Delta(G_{1,1}, G_{2,1}, \pi)$.

Lemma 3.11. *For* $s \gg \left(\frac{\omega(1)\log m}{m}\right)^{\frac{2}{3}}$ *and* p, t *fixed, then*

$$\mathbb{P}(D) \leq \sum_{k=x+1}^{m} \exp\left(k\left(\log m - \frac{mps}{16}\cdot s^2\right)\right) + \exp\left(-\delta^2 \frac{m(1-t)}{2}\right).$$

for $x = \lceil m(1-t)\rceil$ *and* $0 < \delta < 1$.

Proof. Denote by k the number of nodes u such that $\pi(u) \neq u$ and denote by Π_k a subset of all such mappings that fix $m-k$ nodes and permute k nodes. Note that always $k \geq |V_2 \setminus V_0| = m'$. Then we can write

$$\begin{aligned}
\mathbb{P}(D) &\leq \mathbb{P}\left(D|m' \leq x\right) + \mathbb{P}\left(D|m' > x\right) \\
&\leq \mathbb{P}\left(D|m' \geq m(1-t)\right) + \mathbb{P}\left(m' < m(1-t)\right) \text{ as } k > m' > x \\
&\leq \sum_{k=x}^{m}\left(\sum_{\pi\in\Pi_k} \mathbb{P}\left(\Delta(G_{1,1}, G_{1,2}, \pi_0) > \Delta(G_{1,1}, G_{2,1}, \pi)\right)\right) + \mathbb{P}\left(m' < x\right) \qquad (3.1)
\end{aligned}$$

First we estimate $\mathbb{P}\left(\Delta(G_{1,1}, G_{1,2}, \pi_0) > \Delta(G_{1,1}, G_{2,1}, \pi)\right)$. We partition V_2 into two sets of nodes $C_\pi \subset V_2$ and $W_\pi \subset V_2$ such that $u \in C_\pi$ iff $\pi^{-1}(u) = u$ and $u \in W_\pi$ otherwise. Also denote by V_0 nodes that are sampled in G_1 and G_2. Note, that $|C_\pi| = m-k$, $|W_\pi| = k$ and $m' = |V_2 \setminus V_0| = \mathsf{Bi}(m, 1-t)$.

Define mapping $\pi' = \pi \circ g$ where g is a bijection $g : V_2 \to V_1$, which works as follows: If $u \in C_\pi$, then $g(u) = u$; the remaining nodes W_π we map as follows, we arbitrarily split W_π into two equal parts[6] to W_1 and W_2 and we map each $u \in W_1$ s.t. $g(u) = \pi_0(\pi^{-1}(u))$ the rest we map arbitrarily, but not in place. Note that $\pi'|_{W_1 \cup C_\pi} = \pi_0|_{W_1 \cup C_\pi}$

In the following, we show that w.h.p. $\Delta(G_{1,1}, G_{1,2}, \pi') < \Delta(G_{1,1}, G_{2,1}, \pi)$. This follows from Lemma A.2(see Appendix), that states

$$\mathbb{P}\left(\Delta(G_{1,1}, G_{1,2}, \pi') - \Delta(G_{1,1}, G_{2,1}, \pi) < 0\right) \leq \exp\frac{1}{8}\frac{(\lambda_1 - \lambda_2)^2}{\lambda_1 + \lambda_2}, \qquad (3.2)$$

where $\lambda_1 = \mathbb{E}\left[\Delta(G_{1,1}, G_{2,1}, \pi)\right]$ and $\lambda_2 = \mathbb{E}\left[\Delta(G_{1,1}, G_{1,2}, \pi')\right]$. It only remains to prove that $\frac{(\lambda_1-\lambda_2)^2}{\lambda_1+\lambda_2} = \omega(1)$

[6] Without loss of generality we can assume $|W_\pi|$ is even.

$$\lambda_1 = \binom{m-k}{2} 2ps(1-s) + \left((m-k)k + \binom{k}{2}\right) 2ps(1-ps)$$
$$\lambda_2 = \binom{m-k+\frac{k}{2}}{2} 2ps(1-s) + \left(\left(m-k+\frac{k}{2}\right)\frac{k}{2} + \binom{\frac{k}{2}}{2}\right) 2ps(1-ps)$$
$$\lambda_1 - \lambda_2 = k\left(m - \frac{3}{4}k\right) ps^2(1-p) \geq k\frac{m}{4} ps^2(1-p)$$
$$\lambda_1 + \lambda_2 = \left(2m^2 - 3mk + \frac{5}{4}k^2\right) ps(1-s) + \left(3mk - \frac{5}{4}k^2\right) ps(1-ps)$$
$$\leq 4m^2 ps(2-s-ps)$$

Thus, $\frac{(\lambda_1-\lambda_2)^2}{\lambda_1+\lambda_2} \geq \frac{k^2 s^3 p}{64(2-s-ps)}$ that is $\omega(1)$ for $k > x$. This enables us to bound the first term of 3.1:

$$\sum_{k=x+1}^{m} \sum_{\pi \in \Pi_k} \mathbb{P}\left(\Delta(G_{1,1}, G_{1,2}, \pi_0) > \Delta(G_{1,1}, G_{2,1}, \pi)\right) \overset{(a)}{\leq} \sum_{k=x+1}^{m} \sum_{\Pi_k} \mathbb{P}\left(\Delta(G_{1,1}, G_{1,2}, \pi_0) > \Delta(G_{1,1}, G_{1,2}, \pi')\right)$$
$$\overset{(b)}{\leq} \sum_{k=x+1}^{m} \exp k\left(\log m - \frac{mps}{16} \cdot s^2\right) \tag{3.3}$$

The (a) follows from 3.2 and the last inequality (b) follows from Equation 19 of [88] where conditions of the Theorem 4.1 from [88] are met (except the condition $p \to 0$ that the authors never use).

The second term of 3.1 follows from the fact that

$$\mathbb{P}\left(m' < x\right) \leq \mathbb{P}\left(m' < m(1-t)(1-\delta)\right) \leq \exp -\frac{m(1-t)\delta^2}{2} \tag{3.4}$$

due to Chernoff bound. Note that $m' = \mathsf{Bi}(m, 1-t)$.

Then putting together 3.1, 3.3 and 3.4 we obtain

$$\mathbb{P}(D) \leq \sum_{k=x+1}^{m} \exp\left(k\left(\log m - \frac{mps}{16} \cdot s^2\right)\right) + \exp\left(-\delta^2 \frac{m(1-t)}{2}\right).$$

□

3.2 Towards a General Assembly Algorithm for Arbitrary Patches

In Section 3.1, we considered a graph assembly problem in the setup where patches are possibly noisy egonets. The proposed fingerprint-assembly algorithms rely on the fact that (i) each patch consists of a central node and adjacent neighbors and (ii) we have all the egonets of the master graph as an input. Many questions remain open: (i) What if we have the general shaped patches (not egonets)? (ii) What if nodes have labels, how can we incorporate this information? (iii) What if the number of patches is smaller than the number of nodes? (Obviously there is some overhead of edges that are highly repetitive among the egonets, this overhead can be explored to reduce the number of patches.) (iv) What if noise in the patch creation process is not uniform? These questions reflect real network-assembly challenges such as neural-network assembly, where labels are types of neurons and the goal is to assemble a neural network from its small observations [108]; or, the assembly of a social network from its multiple observations, where labels are the first names. Consider the last example in detail: assume we are given multiple observations of a social network, where only the first names of users are known. The labels are highly ambiguous because many persons can have the same name. In general, observations are not ego-centered and some observations are more incomplete than others. For example, consider collaboration networks provided by the e-print arXiv [45, 75] constructed from citation graphs of papers from a particular a category (Astro Physics, Condensed Matter, General Relativity, etc.). The authors' names are provided, but they are ambiguous. The goal is to construct a general collaboration graph. In these examples, we cannot apply fingerprint assembly algorithms because of the assumptions described above. However, we have some labels as side information. We assume that the same users across two patches keep the same label.

In this section, we propose a general assembly algorithm that does not rely on the shape or the number of patches. We emphasize that the proposed algorithm is rather a direction towards a general class of a graph-assembly algorithms. Many questions about different regimes (small large patches) as well as about the theoretical guarantees and analysis of proposed heuristics, remain open.

In summary, we are interested in the general scenario where we have multiple patches of arbitrary shapes. We consider a patch collection $\mathscr{P}$ in its most general form, see Definition 1.4, originated from a master graph G. We assume that the master graph is labeled with a small label set $[l]$ and that this labeling is preserved through the patching process. In other words, each copy of a node in the patches has the same labels as in the master graph. Then, an assembly problem is to merge these patches to obtain the estimator $\hat{G}$ of the master graph G, as before.

The labels serve as side information. For the preprocessing step of the algorithm that we propose, we process this side information as follows: for each patch we identify small, rare, labeled subgraphs. By analogy with the network-alignment problem, we call these subgraphs *seed-subgraphs*. Using the frequencies of these subgraphs, we identify patches with large

intersection. After this, we proceed by aligning the chosen pairs of patches and merging them pairwise.

3.2.1 General Assembly Algorithm: High-Level Description

Our strategy consists of three major steps, repeated iteratively:

1. Select a pair of patches $G_1, G_2 \in \mathscr{P}$ that are "close" to each other. By close, we mean patches having large node-intersection or, in other words, having many nodes originated from the same node of the master graph.

2. Find an alignment π between these two patches (see Definition 1.1). We consider a regime where we have few relatively large patches: in this scenario it is rational to use PGM to align the patches.

3. We merge the two aligned patches into a new patch and replace the two patches in the collection $\mathscr{P}$ by the new one.

We repeat these steps until $|\mathscr{P}| = 1$. The remaining element is the estimator $\hat{G}$ of the master-graph G.

Below, we explain each step of the algorithm in detail and evaluate its performance.

3.2.2 Main Steps of the Algorithm

Selecting "Close" Patches: Graph Similarity Heuristic

To select a pair of patches that have a large node-intersection, we propose a heuristic that measures the similarity of two graphs, based on frequencies of common seed-subgraphs. This *seed-similarity* metric puts a larger weight on rare subgraphs. The intuition behind this is that the more rare a subgraph is overall, the more likely its occurrence in both graphs will signal an overlap. Finding these rare subgraphs of a graph is a problem of independent interest. It is studied in network security for finding threats [51], for identifying malicious software pieces, or for revealing attacks on social networks [15].

We say that a labeled graph H *occurs* in a graph G if there exists a subgraph H' of G that is isomorphic to H with consistent labels. Denote by $N_H(G_i)$ the number of occurrences of H in G_i. In this work, we restrict ourselves to the following types of subgraphs: singletons, single-edges, closed triangles and 4-cliques. Denote the set of all the labeled subgraphs of these four types by T. We call a labeled subgraph $H \in T$ a *seed-subgraph*. We also define $N_H(\mathscr{P}) = \sum_{G_i \in \mathscr{P}} N_H(G_i)$ a total number of occurrences of a subgraph H through all the patches of a collection. We formally define a graph similarity metric as follows.

Definition 3.12 (Seed-Similarity)**.**

$$sim(G_i, G_j) = \sum_{H \in T} \frac{\min(N_H(G_i), N_H(G_j))}{N_H(\mathscr{P})}$$

This metric is similar to tf – idf metric [7], where graphs can be viewed as documents and seed-subgraphs as terms. However, in experiments, we observe that the sim metric has a slight advantage as it is normalized by the total number of occurrences rather than by the number of patches where a subgraph occurs. See evaluation of the proposed heuristic in 3.2.3.

For the first step of each iteration of the assembly algorithm, we select two patches from $\mathscr{P}$ as follows, G_1 is picked at random, G_2 is the one with the highest seed-similarity $sim(G_1, G_2)$.

Seed Selection for Patch Alignment

We align two patches G_1 and G_2 selected at the previous step with the PGM algorithm described in Section 2.2. To find an initial seed-set, we use the fact that PGM is highly robust to the noise in the seed-set [60], thus we can tolerate some errors in the seed-set. Given the two selected patches G_1 and G_2, we find similar nodes based on the Jaccard similarity index between the following multisets, each consists of labels of neighbors of the node. The definition of the Jaccard-similarity index of the two sets, A and B, is as follows:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

More precisely, for the two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$, the seed selection step works as follows. Denote by $N_l(v)$ a multiset of labels of neighbors of v. For each node $v_1 \in V_1$, we select a node $v_2 \in V_2$ such that $J(N_l(v_1), N_l(v_2))$ is minimal over all $v_2 \in V_2$. We order an obtained list of pairs (v_1, v_2) by their corresponding Jaccard indices $J(N_l(v_1), N_l(v_2))$. We take the top a_0 pairs (with the highest Jaccard index) where a_0 is the required number of seeds that we want to select.

Percolation Graph Matching and Merge

By embedding labels into the PGM process, we adjust the PGM algorithm to the labeled case, thus considering only nodes with the same label as the possible matches.

[7] tf – idf heuristic is intended to reflect how important a word is to a document in a collection or corpus.
tf – idf = $TF(t)IDF(t)$, where
$TF(t)$ = (Number of times word t appears in a document) / (Total number of words in the document).
$IDF(t) = log_e$(Total number of documents / Number of documents with word t in it).

We run a PGM over the selected pair of patches $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ with the selected seeds (with high Jaccard similarity) and obtain the alignment π. We merge these two graphs into a new patch $G_3(V_3, E_3)$ as follows. The new vertex set $V_3 = V_1 \cup V_2 \setminus \pi(V_1)$ and the new edge set $E_3 = E_1 \cup E_2'$, where E_2' are the edges E_2 projected to the new vertex set V_3 as follows: for any $(i, j) \in E_2$ we include (i', j') to E_2' where if $i \in V_2 \setminus \pi(V_1)$ then $i' = i$ else $i' = \pi^{-1}(i)$ (same for j', if $j \in V_2 \setminus \pi(V_1)$ then $j' = j$ else $j' = \pi^{-1}(j)$). Consider the example of two patches in Figure 3.5 and assume we merge them via alignment $\pi = \{(\emptyset, 1), (2,2), (3,5), (4,4), (5,\emptyset)\}$. As a result $V_3 = [5]$, $E_2' = \{(1,3), (2,3), (3,4)\}$ and $E_3 = \{(1,3), (2,3), (2,5), (3,4), (4,5)\}$.

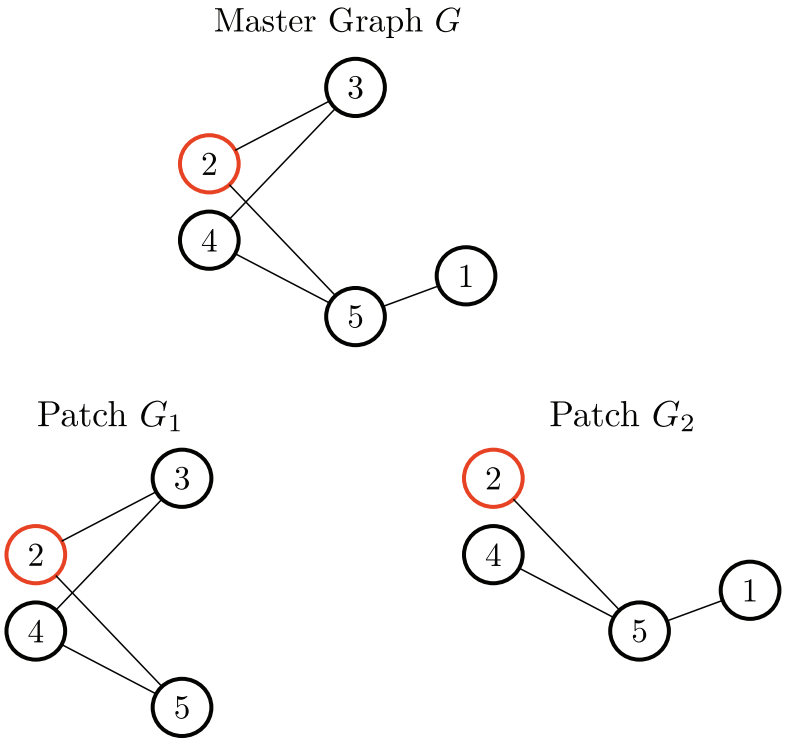

Figure 3.5 – Two patches G_1 and G_2 generated from a master graph G where nodes are labeled with red and black.

Note that the merge procedure is, in fact, symmetric because the new edge set is a union of the edge sets of the two merged graphs, even though the labels of the aligned vertices are taken from the left graph.

3.2.3 Evaluation

Experimental Setup

We experiment with an egonet patch collection $\mathscr{P} = \{G_i\}_{i \in [n]}$ similar to the one considered in the previous Section 3.1, only with larger ego-nets. We start by generating a master graph $G(V, E)$ from the $G(n, p; q)$ model (see Section 1.3.2). We assign labels uniformly from a label

set $[l]$ with size $l \ll n$. From G, we generate a patch collection $\mathscr{P}$ with m patches, as follows. To generate a patch G_i, we select a node $i \in V$ at random; then we explore G starting from i with a breadth-first search, until we visit D nodes. We denote the obtained set of visited nodes by N_i. We take a subgraph of G induced by N_i and anonymize it, as described in Definition 1.7; the resulting graph G_i is our patch. We keep original labels in the patches. We generate m patches in this way.

Note that, even though we use the $G(n,p;q)$ model and extend egonets as patches, the algorithm does not rely on any of these assumptions and is applicable to an arbitrary network and an arbitrary type of patches. We create a table of the main parameters of the algorithm and the ranges of considered values:

	Parameter	Range
Number of labels	l	$\{50, 100200, 500\}$
Number of patches	m	$\{50, 100, 200\}$
Size of a patch	D	$\{500, 1000, 2000, 4000\}$
PGM threshold	r	$\{3, 8, 14\}$
Average degree of the generator graph	np	12
Probability of triangle closure	q	0.95
Number of nodes in the graph	n	20000

Table 3.1 – Parameters of the graph assembly algorithm.

Graph-Similarity Heuristic

To evaluate the proposed seed-similarity metric, we set up the following experiment: From the aforementioned patch collection $\mathscr{P}$, we select a patch G_i at random; then, for each patch $G_j \neq G_i$, we plot the size of the intersection (taken from the ground truth) $G_i \cap G_j$ vs. the value of the proposed heuristic $sim(G_i, G_j)$; we repeat for 20 realizations of the master graph.

In the later experiments on the assembly algorithm, we consider only the simplest seed-subgraphs (triangles) for computational purposes. We suggest, however, that more complicated (hence more rare) subgraphs should provide higher confidence in the selected patches. For this we evaluate three variations of the seed-similarity metric: First, we plot $sim(G_i, G_j)$ where the set of seed-subgraphs contains only triangles ($T = T_3$); second, we plot $sim(G_i, G_j)$ where the set of seed-subgraphs contains all the triangles and subgraphs that are formed by 4-cliques ($T = T_{3,4}$); and third, we consider the tf$-$idf metric that is similar to $sim(G_i, G_j)$, but is normalized by a number of patches containing a seed subgraph instead of the total number of occurrences.

See Figures 3.6, 3.7 and 3.8 for comparisons of three proposed variations of the heuristic for the graph similarity, with respect to how well they reflect node overlap. For 50 labels we already observe high correlation between the proposed seed-similarity of the two patches and

their node overlap. Note that we are interested in the question about whether highly similar patches have high node overlap? Precisely, at each iteration, we need to select only one pair of patches for the algorithm to continue, hence we are not interested in dense clusters of patches with low overlaps. Figure 3.6 demonstrates a regime where we have an extremely small number of labels $l = 20$, this is insufficient to identify highly overlapping patches. In Figure 3.7, we see that even $l = 50$ labels are enough to efficiently find close patches. Figure 3.8 shows the result for larger patches $D = 1000$, where the seed-similarity with $T = T_3$ has a slight advantage over the tf – idf metric. We also note that the seed-similarity with $T = T_{3,4}$ is more accurate for the patches with low overlap. Overall, we see that selecting graphs with a large seed-similarity likely provides us patches with high intersection.

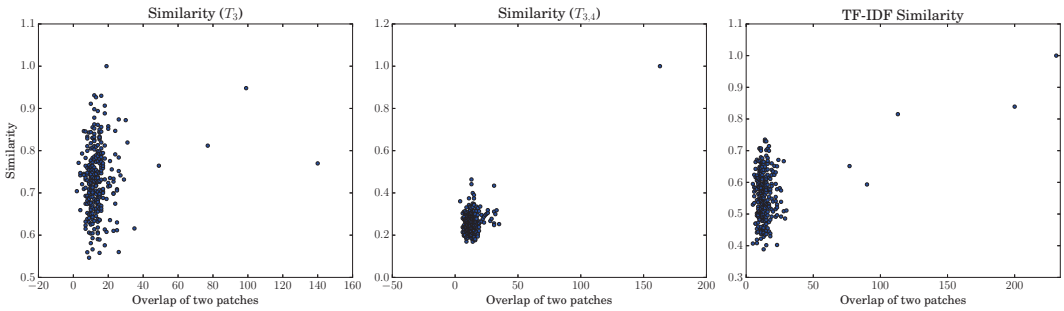

Figure 3.6 – The proposed heuristics for graph similarity for $l = 20, m = 100, D = 500$.

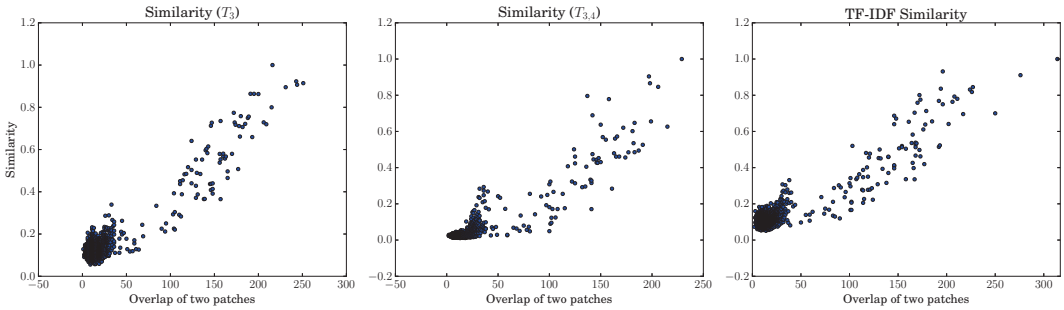

Figure 3.7 – The proposed heuristics for graph similarity for $l = 50, m = 100, D = 500$.

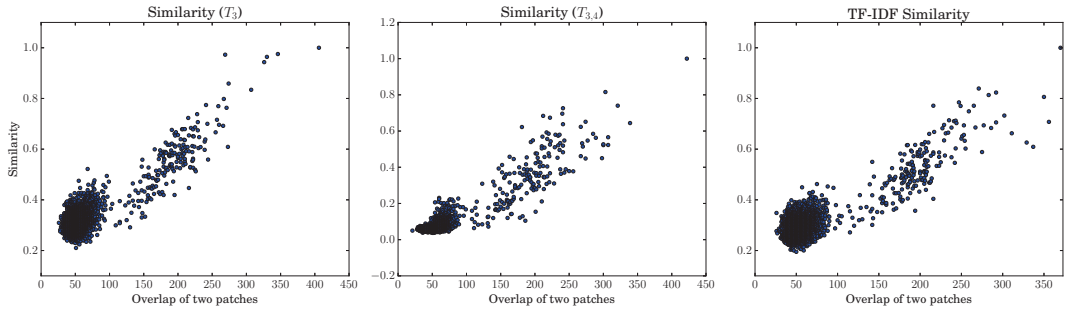

Figure 3.8 – The proposed heuristics for graph similarity for $l = 50, m = 100, D = 1000$.

Seed Selection for Patch Alignment

To evaluate our seed-selection strategy we set up the following experiment: for label-set sizes $l = \{100, \ldots, 1000\}$, we generate a labeled graph $G(n, p; q)$ with $n = 20000$ and generate a patch collection with 500 patches, each of size 2000. We select two patches G_1 and G_2 as follows: the first patch is picked at random and the second one is the one with maximal seed-similarity; then we select $a_0 \sim 200, 600, 1000$ seed-pairs with the highest Jaccard similarity index, as described above, and we compute a fraction of correct seeds among the selected ones (the correct ones are those originated from the same node of the master graph). We compare the performance of our algorithm with a simple baseline that selects a_0 pairs with the same label as follows. First, it takes a least frequent label γ of G_1 then it adds to a seed-set all the pairs (s_1, s_2) such that $s_1 \in V_1$ and $s_2 \in V_2$ and s_1 and s_2 have the label γ; it proceeds with the second least frequent label and so on, until it reaches a_0 seeds. The results are shown at the Figure 3.9. You can see that the number of labels has little effect on the fraction of correct seeds, whereas number a_0 is important. Hence it is easy to find a few correct seeds with highly similar neighborhoods. This observation can be helpful step towards seedless network alignment.

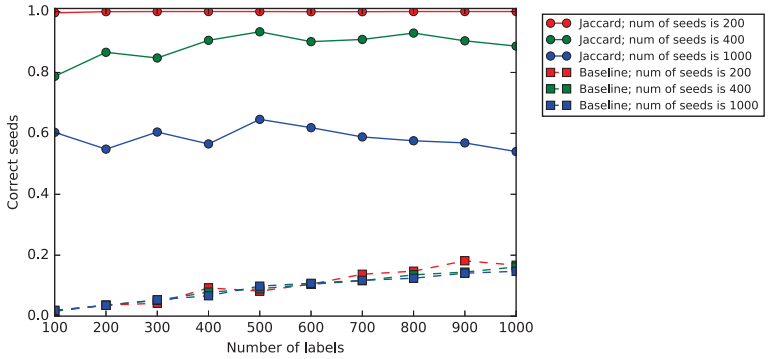

Figure 3.9 – Number of correct seeds vs. number of labels l ($m = 500, D = 2000$).

Experiments on the Assembly Algorithm

We run several experiments to demonstrate the trade-offs between different parameters of the algorithm and to evaluate its performance.

Performance Characteristics

To evaluate the performance of the algorithm we compute and plot the precision, recall and F1-score metrics, we compute these by comparing the obtained results with the ground truth defined as follows. First, for an initial patch collection $\mathscr{P} = \{G_i\}_{i\in[n]}$, for each patch $G_i(V_i, E_i)$, we define a ground truth function $g_i = f_i^{-1} : V_i \to V$ (recall from Definition 1.4 that f_i is a bijection). Each time we merge two patches $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ into a new patch $G_3(V_3, E_3)$, we define $g_3 : V_3 \to V$ as $g_3(v) = g_1(v)$ if $v \in V_1$ and $g_3(v) = g_2(v)$ if $v \in V_3 \setminus V_1$.

As we described earlier, the algorithm terminates when only one patch remains in the collection. We take it as an estimator $\hat{G}$ of the master graph G. We take its corresponding ground truth function and denote it $g : \hat{V} \to V$; we also define $g(\hat{E}) : \hat{E} \to \binom{V}{2}$ from g. We compute the following quantities from the output of the algorithm: the set of recovered edges $\hat{E}$ and the set of recovered edges in the ground truth coordinates $g(\hat{E})$. Denote $E_{corr} = g(\hat{E}) \cap E$, where E is a set of original edges. We define the precision, recall and F1-score:

$$\text{precision} = \frac{|E_{corr}|}{|\hat{E}|}$$

$$\text{recall} = \frac{|E_{corr}|}{|E|}$$

$$\text{F1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Note that the projected set $g(\hat{E})$ (hence the introduced metrics) can take different values in scenarios where the algorithm cannot make a deterministic choice. See the example in Figure 3.5 that contains two patches generated from a master graph. Although merging these two patches, the algorithm cannot possibly decide between the two following alignments: $\pi_1 = \{(\emptyset, 1), (2,2), (3,\emptyset), (4,4), (5,5)\}$ or $\pi_2 = \{(\emptyset, 1), (2,2), (3,5), (4,4), (5,\emptyset)\}$, however, these two alignments result in different estimated edge-sets $g(\hat{E}_1)$ and $g(\hat{E}_2)$. Hence, for patches merged via π_1, $|E_{corr}| = 5$ and resulting in recall = 1 and precision = 1, whereas for patches merged via π_2, $|E_{corr}| = 4$ resulting in recall = $\frac{4}{5}$ and precision = $\frac{4}{5}$.

For the numbers of labels in $\{50, 200, 500\}$, in Figures 3.10, 3.11 , 3.12 , 3.13 , 3.14 and 3.15, we plot the precision/recall/F1 curves vs. the size of the patch D and vs. the number of patches m.

We see that the performance of the algorithm depends strongly on several factors and has several sensitive parameters: large/small number of labels l, large/small patch size D, many/few patches m and large/small r.

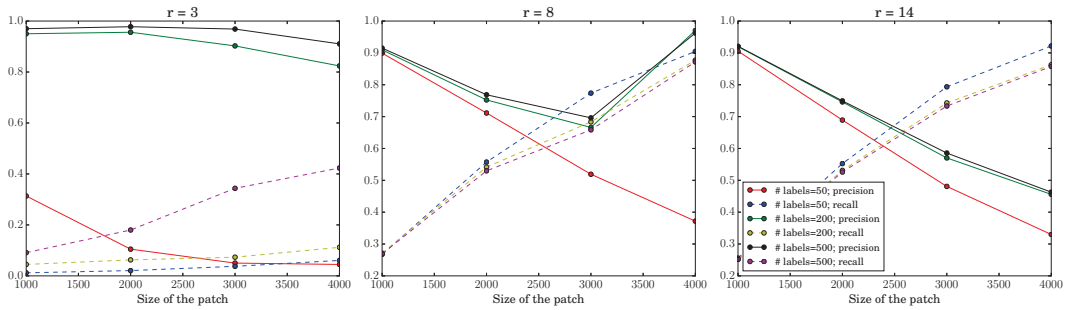

Figure 3.10 – Precision/Recall of the assembly vs. size of the patch D for a number of patches $m = 50$.

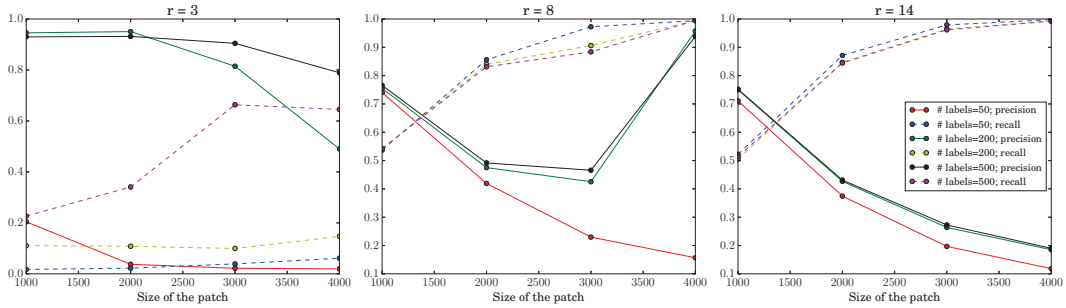

Figure 3.11 – Precision/Recall of the assembly vs. size of the patch D for a number of patches $m = 150$.

We show three subplots for three values of $r = \{3, 8, 14\}$. Note that executions with different values of r introduce fundamentally different types of errors: for a small r, the algorithm tends to percolate to a larger set of nodes introducing errors by merging erroneously and reducing recall; whereas for a larger r, the algorithm stops too early, hence introduces duplicates (not merging some nodes) thus reducing precision.

Several observations:

- Larger patches (and more patches), i.e., larger D and m, introduce more errors in some regimes (see Figure 3.10 with $r = 14$, for example). Explanation: suppose the entire network (or a large part) is covered by patches and the graph assembly algorithm succeeds in merging most of them. Beyond this, adding more patches introduces duplicates and errors, increasing the number of wrong edges; whereas the number of correct edges does not increase or increases very little.
- There is a tradeoff in the size of the patch D vs. the number of patches m: We need to assure that not only a large part of the network is covered by patches, but also that patches have a large enough intersection to find alignments. However, in the proposed approach, few larger patches are easier to align than multiple small patches.

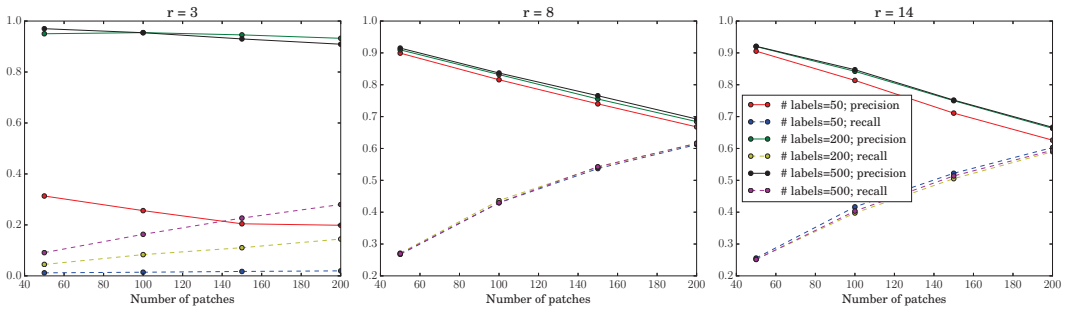

Figure 3.12 – Precision/Recall of the assembly vs. number of patches m for a size of a patch fixed to $D = 1000$.

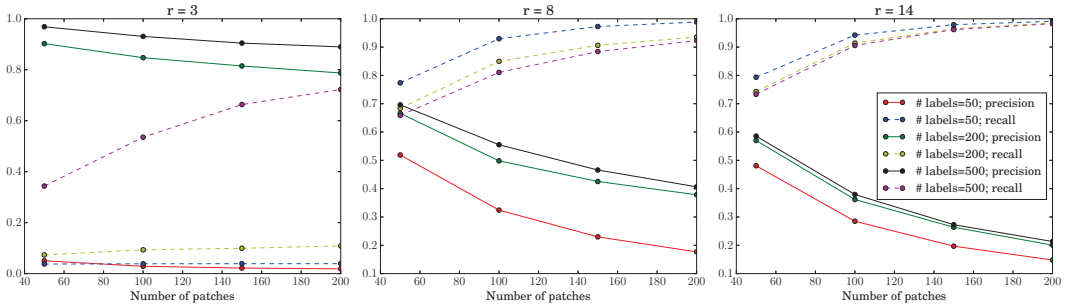

Figure 3.13 – Precision/Recall of the assembly vs. number of patches m for a size of a patch fixed to $D = 3000$.

- The algorithm that runs with a smaller r is more sensitive to the label-set size l, because labels prevent the algorithm from erroneously merging nodes (see Figures 3.14 and 3.15 for $r = 3$).
- The algorithm that runs with a larger r performs worse for larger patches (see Figures 3.14 and 3.15 for $r = 8$ and $r = 14$), because for a larger r, the PGM stops too early hence introduces more duplicates. And labels do not help in this case. Thus, executions with larger patches and large r introduce more duplicates than executions with smaller patches.

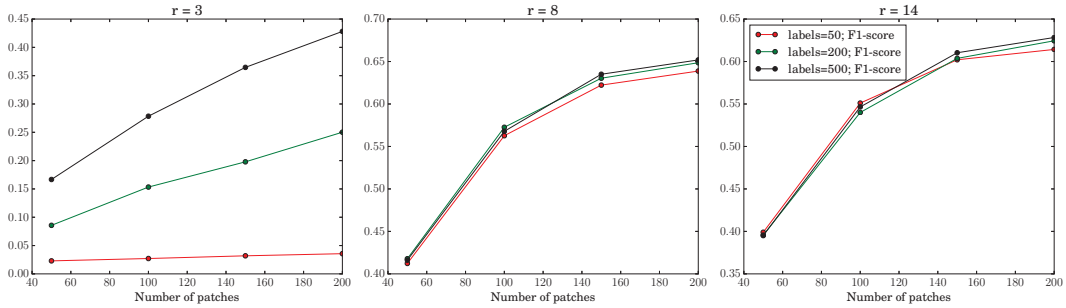

Figure 3.14 – F1-score of the assembly vs. number of patches m for a size of a patch fixed to $D = 1000$.

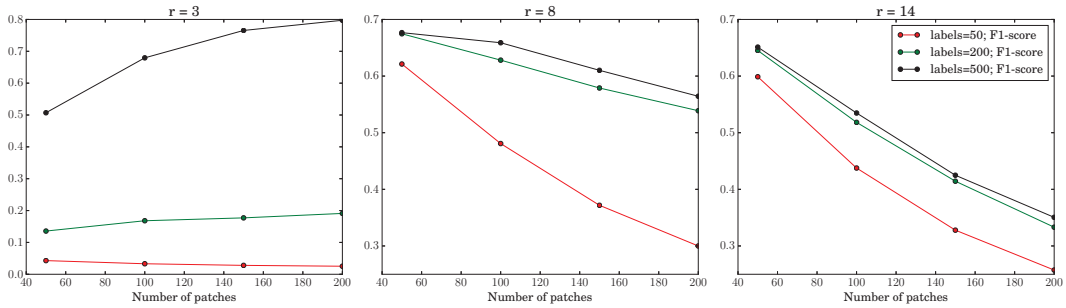

Figure 3.15 – F1-score of the assembly vs. number of patches m for a size of a patch fixed to $D = 3000$.

3.2.4 Discussion

Space/Time Complexity

First, note that our algorithm makes exactly $m-1$ iterations; at each iteration it selects, aligns and merges a pair of patches. The largest data-structure we have to maintain is the counts of the seed-subgraphs $\{N_H(G_i)_{H\in T}\}$ for each patch G_i (for that we also maintain the total frequencies for the collection $\{N_H(\mathscr{P})_{H\in T}\}$), we call these structures f_{seed}. If we consider only triangles as seed-subgraphs $T = T_3$, the total number of labeled copies of these is l^3, hence for m patches we have to maintain up to ml^3 frequencies in f_{seed}. Computing this map for each patch takes around Dd_{avg}^2 checks for node connectivity (to find triangles we check whether two neighbors of a node are connected), where d_{avg} is an average degree of the node in the patch. We have to recompute f_{seed} completely at each iteration of the algorithm, because if a structure of at least one patch is changed, the total number of occurrences of the seed-subgraphs $N_H(P)$ needs to be updated and the frequencies have to be renormalized. Thus we need around $m^2Dd_{avg}^2$ connectivity checks to maintain this structure f_{seed}. At each iteration, we need to search the frequencies f_{seed} m times to compute seed-similarity and to select a similar pair of patches, thus we perform around m^2 computations of seeds similarity,

that is up to $m^2 l^3$ operations over the maps. We implement f_{seed} as a hashmap for each patch, where the key is a triple of the labels of the seed-subgraph (we assume three labels are ordered, to account for permutations) and the value is a normalized frequency of the seed-subgraph. Hence the time complexity of the algorithm can be estimated as $\Theta(m^2 D d_{avg}^2 + m^2|x|)$, where x is at most l^3 a number of different seed-subgraphs in a patch. In summary, the performance of the algorithm is highly dependent on m and D, whereas the dependency on n is only imposed through the necessity of the network being well covered by patches.

The proposed algorithm is a direction towards a general powerful class of graph assembly algorithms. The first open question is about theoretical analysis of the proposed graph similarity metric and seed-selection criteria for different random graph models. Overall, measuring labeled-graph similarity based on common rare subgraphs is of independent research interest. Another question is about considering very small patches. The use of PGM is not justified in this scenario. Thus some other technique for alignment of pairs of patches can be used. We suggest to use techniques analogous to the seed selection method (similarity of labeled neighborhoods) to find an alignment of the nodes of the two patches.

Conclusion

In this thesis, we study network reconstruction from ambiguous data. We discussed many reasons for data to be ambiguous and incomplete, such as privacy concerns, specifics of data collection, or errors in the data processing. We proposed to explore structural information in the data to reconstruct an accurate version of the network. We looked at the problem from two different perspectives: First, we formulated the network alignment problem, where we infer a network from large, but noisy observations; second, we formulated the network assembly problem where we infer a network from multiple small observations, called patches. We considered two settings that are with and without side information.

In Chapter 2, we investigated the network alignment problem. In the first part, we demonstrated the theoretical feasibility of an alignment of two graphs with partial node overlap under a proposed random graph model. We formulated sufficient conditions on density and overlap where networks can be aligned with no side information. In the second part, we assumed that we are given a small amount of side information in the form of a seed set, and we proposed and analyzed a percolation graph matching algorithm for network alignment. We characterized a sharp threshold in the number of seeds for the algorithm to succeed. We also proposed an improvement of the algorithm to a scalable version that successfully aligns large graphs of 10 millions nodes.

In Chapter 3, we investigated the network assembly problem. At first we showed the theoretical feasibility of assembly of networks from multiple, possibly noisy observations under the proposed random graph model. We formulated a property of the graphs generated from this model. This property is the uniqueness of subgraphs of common neighbors of an edge. These subgraphs serve as an unique edge fingerprint and enables us to stitch patches together. In the second part of the chapter, we assume access to side information, in the form of node labels, where labels are assigned from a small label set. We proposed a graph assembly algorithm that works with patches of the most general form.

Future Directions and Perspectives

Network alignment has received much of attention recently and many open problems were solved, such as the analysis of PGM under more realistic network model or tight converse

bounds on the feasibility of network alignment. We list a few questions that remain open:

- The next step in the understanding of network alignment is to characterize conditions for the feasibility of network alignment with no side information for more realistic models. The graphs generated from, for example, Barabási–Albert [5] model possess such a property as a power-law degree distribution, thus suggesting more distinctive neighborhoods of nodes. Such properties might help in finding an alignment.

- Another open direction is finding algorithms for seedless graph-alignment. One idea is to use the seed-selection techniques analogous to those in 3.2.2, but instead of label-sets of the neighbors, we could look at the similarities of the degree sequence of the neighbors of a node. Another idea is to combine the PGM with the matching algorithm proposed in [87] to search for likely seeds. The fact that PGM is robust to the noise in the seed-set enables us to tolerate errors in the output of the matching algorithm.

- One unusual application of PGM is an efficient finding duplicates in the graph. For this we observe that PGM searches for a counterpart of the nodes in two aligned graphs. Thus, if we align the graph $G(V,E)$ with itself, with slight modifications of PGM, for each pair of duplicates $v, v' \in V$ the PGM should provide us a list of pairs (v,v), (v',v'), (v',v) and (v,v') all with the same number of marks. We suggest experimenting on this hypothesis in the noiseless and the noisy scenarios, and analyzing the performance under the Erdős-Rényi graph model. We also point out here that the complexity of PGM is $\Theta(n * d_{avg}^2)$, whereas naive search for duplicates might need n^2 comparisons; which is extremely important for large graphs.

As we mentioned, the network assembly problem is a new direction with many unexplored questions. We name only a few; this list is not exhaustive.

- For the feasibility of the graph-assembly problem, the first open question is about how many egonet patches are actually needed. Is it enough to have each edge occur exactly once in the patch collection and to have at least one patch to merge until we reach n nodes of the graph?

- A theoretical analysis of our proposed heuristics for graph assembly, such as seed-similarity metric and seed-selection criteria, is an open direction. First, an analysis of these, for basic random-graph models, might provide some theoretical guarantees and some ways for improvements. Second, it is interesting to compare the proposed seed-similarity with other graph-similarity metrics, that are of independent interest.

- Maintaining a history of merges in the graph-assembly process might improve the quality of the results, as well as provide a more precise evaluation technique that estimates each merge and not only the final estimator.

- One interesting application of the network assembly is to analyze some graph properties without complete graph reconstruction. For example, to obtain and analyze a path $v_1, v_2, \ldots, v_k$ we could stitch several patches (via some of the proposed assembly algorithms), thus obtaining a small subgraph of a whole graph that contains the path. This is relevant for extremely large graphs that are hard to analyze as a whole.
- The last observation is that graph-assembly algorithms are highly parallelizable because each merge is independent of the others, thus we might adapt the current implementation in order to scale the algorithms to even larger graphs.

A An appendix

A.1 Concentration Lemmas

Lemma A.1. [Chernoff-Hoeffding bound [39]]
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* $0 < \varepsilon < 1$,

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

Lemma A.2. [Difference of Binomials [88]] *Let* X_1 *and* X_2 *be two binomial random variables with means* λ_1 *and* λ_2, *where* $\lambda_2 > \lambda_1$. *Then,*

$$\mathbb{P}(X_2 - X_1 \leq 0) \leq 2\exp\left(-\frac{1}{8}\frac{(\lambda_2 - \lambda_1)^2}{\lambda_2 + \lambda_1}\right).$$

Bibliography

[1] I. BHATTACHARYA AND L. GETOOR, *Collective entity resolution in relational data*, ACM Transactions on Knowledge Discovery from Data, 1 (2007). [Cited on pages 2 and 9.]

[2] E. ADAR AND L. A. ADAMIC, *Tracking information epidemics in blogspace*, in Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, Washington, DC, USA, 2005. [Cited on page 1.]

[3] W. AIELLO, F. CHUNG, AND L. LU, *A random graph model for power law graphs*, Experimental Mathematics, (2001). [Cited on page 7.]

[4] M. AIZENMAN AND J. L. LEBOWITZ, *Metastability Effects in Bootstrap Percolation*, Journal of Physics A: Mathematical and General, 21 (1988). [Cited on page 13.]

[5] R. ALBERT AND A.-L. BARABÁSI, *Statistical mechanics of complex networks*, Reviews of Modern Physics, (2002). [Cited on page 94.]

[6] P. ANCHURI, M. J. ZAKI, O. BARKOL, S. GOLAN, AND M. SHAMY, *Approximate graph mining with label costs*, in Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, IL, USA, August 2013. [Cited on page 10.]

[7] A. ARASU, J. NOVAK, AND J. TOMLIN, *Pagerank computation and the structure of the web: Experiments and algorithms*, 2002. [Cited on page 1.]

[8] J. ARINO AND P. VAN DEN DRIESSCHE, *A Multi-City Epidemic Model*, Mathematical Population Studies, 10 (2003). [Cited on page 1.]

[9] V. ARVIND, B. DAS, AND J. KÖBLER, *The space complexity of* k *-tree isomorphism*, in Proceedings of the 18th International Symposium on Algorithms and Computation, Sendai, Japan, December 2007. [Cited on page 70.]

[10] V. ARVIND, J. KÖBLER, S. KUHNERT, AND Y. VASUDEV, *Approximate graph isomorphism*, in Mathematical Foundations of Computer Science : 37th International Symposium, Proceedings, 2012. [Cited on page 8.]

[11] N. ATIAS AND R. SHARAN, *Comparative analysis of protein networks: Hard problems, practical solutions*, Communications of the ACM, 55 (2012). [Cited on page 2.]

[12] A. AWAN, H. BARI, F. YAN, S. MOKSONG, S. YANG, S. CHOWDHURY, Q. CUI, Z. YU, E. PURISIMA, AND E. WANG, *Regulatory network motifs and hotspots of cancer genes in a mammalian cellular signalling network*, IET Systems Biology, 1 (2007). [Cited on page 2.]

[13] L. BABAI, *Graph isomorphism in quasipolynomial time*, CoRR, abs/1512.03547 (2015). [Cited on page 70.]

[14] L. BABAI, P. ERDÖS, AND S. M. SELKOW, *Random graph isomorphism*, SIAM Journal on Computing, 9 (1980). [Cited on page 8.]

[15] 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, 2007. [Cited on pages 6, 7, and 82.]

[16] S. BANDYOPADHYAY, U. MAULIK, L. B. HOLDER, AND D. J. COOK, *Advanced Methods for Knowledge Discovery from Complex Data (Advanced Information and Knowledge Processing)*, Springer-Verlag New York, Inc., 2005. [Cited on page 1.]

[17] J. B. L. BARD AND S. Y. RHEE, *Ontologies in biology: Design, applications and future challenges*, Nature Reviews Genetics, 5 (2004). [Cited on page 2.]

[18] M. BAYATI, D. F. GLEICH, A. SABERI, AND Y. WANG, *Message-passing algorithms for sparse network alignment*, ACM Transactions Knowledge Discovery Data, 7 (2013). [Cited on page 8.]

[19] I. BHATTACHARYA AND L. GETOOR, *Entity resolutions in graphs*, in Mining Graph Data, Wiley, 2006. [Cited on page 2.]

[20] M. BIRYUKOV, *Co-author Network Analysis in DBLP: Classifying Personal Names*, 2008. [Cited on pages 3 and 5.]

[21] V. D. BLONDEL, M. ESCH, C. CHAN, F. CLÉROT, 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). [Cited on pages 5, 10, and 59.]

[22] V. D. BLONDEL, J.-L. GUILLAUME, R. LAMBIOTTE, AND E. LEFEBVRE, *Fast unfolding of communities in large networks*, Journal of Statistical Mechanics: Theory and Experiment, 2008 (2008). [Cited on page 1.]

[23] B. BOLLOBÁS, *Almost every graph has reconstruction number three*, Journal of Graph Theory, 14 (1990). [Cited on page 9.]

[24] B. BOLLOBÁS, *Random Graphs (2nd edition)*, Cambridge University Press, 2001. [Cited on pages 8, 22, and 48.]

[25] P. BONACICH, *Power and centrality: A family of measures*, American journal of sociology, 92 (1987). [Cited on page 1.]

[26] D. BOYD AND N. B. ELLISON, *Social Network Sites: Definition, History, and Scholarship*, Journal of Computer-Mediated Communication, 13 (2007). [Cited on page 2.]

[27] O. BRDICZKA, J. LIU, B. PRICE, J. SHEN, A. PATIL, R. CHOW, E. BART, AND N. DUCHENEAUT, *Proactive insider threat detection through graph learning and psychological context*, in IEEE Symposium on Security and Privacy Workshops, San Francisco, CA, USA, May 2012. [Cited on page 5.]

[28] G. BRESLER, M. BRESLER, AND D. TSE, *Optimal assembly for high throughput shotgun sequencing*, BMC bioinformatics, 14 (2013). [Cited on page 9.]

[29] G. CASELLA AND R. BERGER, *Statistical Inference*, Thomson Learning, 2002. [Cited on page 62.]

[30] S. CATANESE, P. D. MEO, E. FERRARA, G. FIUMARA, AND A. PROVETTI, *Crawling facebook for social network analysis purposes*, in International Conference on Web Intelligence, Mining and Semantics, 2011. [Cited on page 2.]

[31] C. F. CHIASSERINI, M. GARETTO, AND E. LEONARDI, *De-anonymizing scale-free social networks by percolation graph matching*, in Proceedings of IEEE Information Communication, Hong Kong, 2015. [Cited on pages 4 and 7.]

[32] C. F. CHIASSERINI, M. GARETTO, AND E. LEONARDI, *Impact of clustering on the performance of network de-anonymization*, in Proceedings of ACM Conference on Online Social Networks, Palo Alto, CA, USA, November 2015. [Cited on pages 4 and 7.]

[33] P. CHRISTEN, *A survey of indexing techniques for scalable record linkage and deduplication*, IEEE Transactions on Knowledge and Data Engineering, (2011). [Cited on page 9.]

[34] 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 (2004). [Cited on page 8.]

[35] S. A. COOK, *The complexity of theorem-proving procedures*, in Proceedings of the Third Annual ACM Symposium on Theory of Computing, 1971. [Cited on page 8.]

[36] D. CULLINA AND N. KIYAVASH, *Improved achievability and converse bounds for erdős-rényi graph matching*, Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science, (2016). [Cited on pages 8 and 37.]

[37] J. DE LAS RIVAS AND C. FONTANILLO, *Protein–protein interactions essentials: Key concepts to building and analyzing interactome networks*, PLoS Comput Biol, (2010). [Cited on page 2.]

[38] P. DOSHI, R. KOLLI, AND C. THOMAS, *Inexact matching of ontology graphs using expectation-maximization*, Web Semantics: Science, Services and Agents on the World Wide Web, 7 (2009). [Cited on page 8.]

[39] D. DUBHASHI AND A. PANCONESI, *Concentration of Measure for the Analysis of Randomized Algorithms*, Cambridge University Press, 1st ed., 2009. [Cited on page 97.]

[40] M. EL-KEBIR, J. HERINGA, AND G. W. KLAU, *Natalie 2.0: Sparse global network alignment as a special case of quadratic assignment*, Algorithms, 8 (2015). [Cited on page 8.]

[41] D. ERDŐS, R. GEMULLA, AND E. TERZI, *Reconstructing graphs from neighborhood data*, ACM Transactions on Knowledge Discovery from Data, 8 (2014). [Cited on page 10.]

[42] P. ERDÖS AND A. RÉNYI, *On random graphs isomorphism*, Publicationes Mathematicae Debrecen, 6 (1959). [Cited on pages 8, 11, and 14.]

[43] M. FALOUTSOS, P. FALOUTSOS, AND C. FALOUTSOS, *On power-law relationships of the internet topology*, in Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, 1999. [Cited on page 1.]

[44] A. GANDOMI AND M. HAIDER, *Beyond the hype: Big data concepts, methods, and analytics*, International Journal of Information Management, 35 (2015). [Cited on page 1.]

[45] J. GEHRKE, P. GINSPARG, AND J. KLEINBERG, *Overview of the 2003 kdd cup*, ACM SIGKDD Explorations Newsletter, (2003). [Cited on page 81.]

[46] L. GETOOR AND A. MACHANAVAJJHALA, *Entity resolution: Theory, practice & open challenges*, in International Conference on Very Large Data Bases, 2012. [Cited on page 2.]

[47] M. GJOKA, M. KURANT, C. T. BUTTS, AND A. MARKOPOULOU, *Walking in facebook: A case study of unbiased sampling of osns*, in Proceedings of the 29th Conference on Information Communications, INFOCOM, 2010. [Cited on page 2.]

[48] R. GROSS AND A. ACQUISTI, *Information revelation and privacy in online social networks*, in Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, 2005. [Cited on page 6.]

[49] S. GUHA, K. TANG, AND P. FRANCIS, *NOYB: Privacy in Online Social Networks*, in Workshop on Online Social Networks, 2008. [Cited on page 6.]

[50] 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 Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2011. [Cited on page 8.]

[51] P. HOLME, B. J. KIM, C. N. YOON, AND S. K. HAN, *Attack vulnerability of complex networks*, Physical Review, 65 (2002). [Cited on page 82.]

[52] M. HUMBERT, T. STUDER, M. GROSSGLAUSER, AND J.-P. HUBAUX, *Nowhere to Hide: Navigating around Privacy in Online Social Networks*, Springer Berlin Heidelberg, 2013. [Cited on page 2.]

[53] J. J. P. III, T. L. FOND, S. MORENO, AND J. NEVILLE, *Fast generation of large scale social networks with clustering*, CoRR, (2012). [Cited on page 7.]

[54] S. JANSON, T. ŁUCZAK, AND A. RUCIŃSKI, *Random Graphs*, Wiley, 2000. [Cited on page 11.]

[55] S. JANSON, T. ŁUCZAK, T. TUROVA, AND T. VALLIER, *Bootstrap percolation on the random graph $g_{n,p}$*, The Annals of Applied Probability, 22 (2012). [Cited on pages 38, 39, 40, 44, 45, 47, and 58.]

[56] T. A. JUNTTILA AND P. KASKI, *Engineering an efficient canonical labeling tool for large and sparse graphs*, in Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments and the Fourth Workshop on Analytic Algorithms and Combinatorics, New Orleans, LA, USA, January 2007. [Cited on page 71.]

[57] M. KAFSI, E. KAZEMI, L. MAYSTRE, L. YARTSEVA, M. GROSSGLAUSER, AND P. THIRAN, *Mitigating epidemics through mobile micro-measures*, NetMob, (2013). [Cited on pages 1 and 2.]

[58] F. KAPLAN, *The venice time machine*, in Proceedings of the ACM Symposium on Document Engineering, 2015. [Cited on pages 1 and 3.]

[59] H. KARDES, D. KONIDENA, S. AGRAWAL, M. HUFF, AND A. SUN, *Graph-based approaches for organization entity resolution in mapreduce*, 2013. [Cited on pages 9 and 10.]

[60] E. KAZEMI, S. H. HASSANI, AND M. GROSSGLAUSER, *Growing a graph matching from a handful of seeds*, International Conference on Very Large Data Bases, 8 (2015). [Cited on pages 4, 7, 53, 58, and 83.]

[61] E. KAZEMI, L. YARTSEVA, AND M. GROSSGLAUSER, *When can two unlabeled networks be aligned under partial overlap?*, in 53rd Annual Allerton Conference on Communication, Control, and Computing, Monticello, IL, USA, September 2015. [Cited on pages 3 and 11.]

[62] 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*, Proceedings of the National Academy of Sciences, 100 (2003). [Cited on page 7.]

[63] 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 (2004). [Cited on page 7.]

[64] P. J. KELLY, *A congruence theorem for trees.*, Pacific Journal of Mathematics, 7 (1957), pp. 961–968. [Cited on page 9.]

[65] W. O. KERMACK AND A. G. MCKENDRICK, *A contribution to the mathematical theory of epidemics*, Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 115 (1927). [Cited on page 1.]

[66] G. W. KLAU, *A new graph-based method for pairwise global network alignment*, MC Bioinformatics, 10 (2009). [Cited on page 8.]

[67] A. KOROLOVA, R. MOTWANI, S. U. NABAR, AND Y. XU, *Link Privacy in Social Networks*, in Proceeding of the 17th ACM Conference on Information and Knowledge Management, 2008. [Cited on page 6.]

[68] N. KORULA AND S. LATTANZI, *An efficient reconciliation algorithm for social networks*, in Proceedings of the Very Large Data Bases Endowment, 2014. [Cited on pages 4 and 7.]

[69] B. KRISHNAMURTHY AND C. E. WILLS, *Characterizing privacy in online social networks*, in Proceedings of the First Workshop on Online Social Networks, 2008. [Cited on page 6.]

[70] O. KUCHAIEV AND N. PRŽULJ, *Integrative network alignment reveals large regions of global network similarity in yeast and human*, Bioinformatics, 27 (2011). [Cited on page 6.]

[71] M. KURANT, A. MARKOPOULOU, AND P. THIRAN, *Towards unbiased bfs sampling*, IEEE Journal on Selected Areas in Communications, 29 (2011). [Cited on page 2.]

[72] J. F. KUROSE AND K. ROSS, *Computer Networking: A Top-Down Approach Featuring the Internet*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002. [Cited on page 1.]

[73] A. LAKHINA, J. W. BYERS, M. CROVELLA, AND P. XIE, *Sampling biases in IP topology measurements*, in Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 1, 2003. [Cited on page 1.]

[74] K.-K. LAM, A. KHALAK, AND D. TSE, *Near-optimal assembly for shotgun sequencing with noisy reads*, BMC bioinformatics, 15 (2014). [Cited on page 9.]

[75] J. LESKOVEC, *Stanford network analysis project*. http://snap.stanford.edu/index.html. [Cited on pages 51 and 81.]

[76] J. LESKOVEC AND C. FALOUTSOS, *Sampling from large graphs*, in Proceedings of the International Conference on Knowledge Discovery and Data Mining, 2006. [Cited on page 2.]

[77] J. LESKOVEC, J. KLEINBERG, AND C. FALOUTSOS, *Graphs over time: Densification laws, shrinking diameters and possible explanations*, in Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, 2005. [Cited on page 1.]

[78] J. LESKOVEC, K. J. LANG, A. DASGUPTA, AND M. W. MAHONEY, *Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters*, CoRR, (2008). [Cited on page 1.]

[79] C.-S. LIAO, K. LU, M. BAYM, R. SINGH, AND B. BERGER, *Isorankn: Spectral methods for global alignment of multiple protein networks*, Bioinformatics, 25 (2009). [Cited on page 7.]

[80] J. LIU, K. H. LEI, J. Y. LIU, C. WANG, AND J. HAN, *Ranking-based name matching for author disambiguation in bibliographic data*, in Proceedings of the Knowledge Discovery from Data Cup Workshop, 2013. [Cited on page 10.]

[81] W. LUO AND W. TAY, *Estimating infection sources in a network with incomplete observations*, in Global Conference on Signal and Information Processing, Austin, TX, USA, December 2013. [Cited on page 1.]

[82] N. MASUDA AND P. HOLME, *Predicting and controlling infectious disease epidemics using temporal networks*, tech. rep., 2013. [Cited on page 2.]

[83] E. MOSSEL AND N. ROSS, *Shotgun assembly of labeled graphs*, ArXiv e-prints, (2015). [Cited on pages 9 and 60.]

[84] E. MOSSEL AND N. SUN, *Shotgun assembly of random regular graphs*, ArXiv e-prints, (2015). [Cited on pages 9 and 60.]

[85] A. NARAYANAN AND V. SHMATIKOV, *De-anonymizing social networks*, in Proceedings of the 30th IEEE Symposium on Security and Privacy, Washington, DC, USA, 2009. [Cited on pages 4, 6, 7, and 19.]

[86] M. E. J. NEWMAN, D. J. WATTS, AND S. H. STROGATZ, *Random graph models of social networks*, Proceedings of the National Academy of Sciences of the United States of America, 99 (2002). [Cited on page 11.]

[87] P. PEDARSANI, D. R. FIGUEIREDO, AND M. GROSSGLAUSER, *A bayesian method for matching two similar graphs without seeds*, in Communication, Control, and Computing (Allerton), 2013 51st Annual Allerton Conference on, 2013. [Cited on pages 7 and 94.]

[88] P. PEDARSANI AND M. GROSSGLAUSER, *On the privacy of anonymized networks*, in Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2011. [Cited on pages 4, 6, 8, 11, 19, 22, 80, and 97.]

[89] W. PENG, F. LI, X. ZOU, AND J. WU, *A two-stage deanonymization attack against anonymized social networks*, Computers, IEEE Transactions on, 63 (2014). [Cited on page 4.]

[90] M. PENROSE, *Random Geometric Graphs*, Oxford University Press, USA, 2003. [Cited on pages 7 and 54.]

[91] P. C. PINTO, P. THIRAN, AND M. VETTERLI, *Locating the source of diffusion in large-scale networks*, Physical review letters, 109 (2012). [Cited on page 1.]

[92] J. PUJARA AND L. GETOOR, *Building dynamic knowledge graphs*, in NIPS Workshop on Automated Knowledge Base Construction, 2014. [Cited on page 9.]

[93] J. PUJARA, H. MIAO, L. GETOOR, AND W. COHEN, *Knowledge graph identification*, in International Semantic Web Conference (ISWC), 2013. [Cited on page 9.]

[94] A. QUAN-HAASE AND K. MARTIN, *Digital humanities: The continuing role of serendipity in historical research*, in Proceedings of the 2012 iConference, 2012. [Cited on page 1.]

[95] M. ROSEN-ZVI, T. GRIFFITHS, M. STEYVERS, AND P. SMYTH, *The author-topic model for authors and documents*, in Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence, 2004. [Cited on page 10.]

[96] M.-E. ROSOIU, C. T. DOS SANTOS, AND J. EUZENAT, *Ontology matching benchmarks: Generation and evaluation*, in Proceedings 6th ISWC Workshop on Ontology Matching (OM), Bonn, Germany, 2011. [Cited on page 8.]

[97] M. H. SCHAEFER, L. SERRANO, AND M. A. ANDRADE-NAVARRO, *Correcting for the study bias associated with protein-protein interaction measurements reveals differences between protein degree distributions from different cancer types*, Frontiers in Genetics, 6 (2015). [Cited on page 2.]

[98] G. SCHOENEBECK, *Potential networks, contagious communities, and understanding social network structure*, in Proceedings of the 22Nd International Conference on World Wide Web, 2013. [Cited on page 3.]

[99] B. SCHWIKOWSKI, P. UETZ, AND S. FIELDS, *A network of protein-protein interactions in yeast*, Nature Biotechnology, 18 (2000). [Cited on page 1.]

[100] C. SESHADHRI, A. PINAR, N. DURAK, AND T. G. KOLDA, *Directed Closure Measures for Networks with Reciprocity*, ArXiv e-prints, (2013). [Cited on pages 14 and 60.]

[101] K. SHARAD AND G. DANEZIS, *De-anonymizing d4d datasets*, in Workshop on Hot Topics in Privacy Enhancing Technologies, Bloomington, Indiana, USA, 2013. [Cited on page 10.]

[102] K. SHARAD AND G. DANEZIS, *An automated social graph de-anonymization technique*, in Proceedings of the 13th Workshop on Privacy in the Electronic Society, Scottsdale, AZ, USA, November 2014. [Cited on pages 5 and 10.]

[103] Y.-K. SHIH AND S. PARTHASARATHY, *Scalable multiple global network alignment for biological data*, in ACM Conference on Bioinformatics, Computational Biology and Biomedicine, 2011. [Cited on page 7.]

[104] P. SHVAIKO AND J. EUZENAT, *Ontology matching: State of the art and future challenges*, IEEE Transactions on Knowledge and Data Engineering, 25 (2013). [Cited on page 8.]

[105] R. SINGH, J. XU, AND B. BERGER, *Pairwise global alignment of protein interaction networks by matching neighborhood topology*, in Annual International Conference on Research in Computational Molecular Biology, 2007. [Cited on page 7.]

[106] 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 (2008). [Cited on pages 1 and 7.]

[107] A. SINGHAL, *Introducing the knowledge graph: Things, not strings*, Official Google Blog, May, (2012). [Cited on page 9.]

[108] D. SOUDRY, S. KESHRI, P. STINSON, M.-H. OH, G. IYENGAR, AND L. PANINSKI, *A shotgun sampling solution for the common input problem in neural connectivity inference*, ArXiv e-prints, (2013). [Cited on pages 9, 17, and 81.]

[109] D. A. SPIELMAN, *Faster isomorphism testing of strongly regular graphs*, in Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, Philadephia, USA, 1996. [Cited on page 8.]

[110] M. STARNINI, A. BARONCHELLI, A. BARRAT, AND R. PASTOR-SATORRAS, *Random walks on temporal networks*, Physical Review, 85 (2012). [Cited on page 2.]

[111] S. SUDHAHAR, *Automated Analysis of Narrative Text using Network Analysis in Large Corpora*, PhD thesis, University of Bristol, 2015. [Cited on page 1.]

[112] C. VON MERING, R. KRAUSE, B. SNEL, M. CORNELL, S. G. OLIVER, S. FIELDS, AND P. BORK, *Comparative assessment of large-scale data sets of protein–protein interactions*, Nature, 417 (2002). [Cited on page 2.]

[113] D. J. WATTS AND S. H. STROGATZ, *Collective dynamics of 'small-world' networks.*, Nature, (1998). [Cited on page 14.]

[114] G. WONDRACEK, T. HOLZ, E. KIRDA, AND C. KRUEGEL, *A practical attack to de-anonymize social network users*, in Proceedings of the IEEE Symposium on Security and Privacy, 2010. [Cited on page 6.]

[115] J. YANG AND J. LESKOVEC, *Defining and evaluating network communities based on ground-truth*, CoRR, abs/1205.6233 (2012). [Cited on page 5.]

[116] L. YARTSEVA, J. S. ELBERT, AND M. GROSSGLAUSER, *Assembling a network out of ambiguous patches*, in Proceedings of the 54rd Annual Allerton Conference on Communication, Control, and Computing, no. EPFL-CONF-212900, 2016. [Cited on page 11.]

[117] L. YARTSEVA AND M. GROSSGLAUSER, *On the performance of percolation graph matching*, in Proceedings of the First ACM Conference on Online Social Networks, Boston, MA, USA, October 2013. [Cited on pages 4, 7, and 11.]

[118] B. ZHOU AND J. PEI, *Preserving privacy in social networks against neighborhood attacks*, in Proceedings of the IEEE 24th International Conference on Data Engineering, 2008. [Cited on page 6.]

Lyudmila Yartseva

Present Address
Lausanne, Switzerland
Contacts:
E-mail: lyudmila.v.yartseva@gmail.com
Office Phone: +41 21 693 6468
Mobile Phone: +41 76 226 2901

Interests:

Networks Mining, Reconstruction and Analysis; Large Network Algorithms; Random Graphs.

EPFL and IIS projects

- *Current research: Graph assembly and graph alignment. Reconstructing graphs from multiple different observations*
- *Current research: Graph matchability, graph-matching algorithm*
  Supervision: Professor Grossglauser (EPFL LCA4). We investigate the problem of graph matching which refers to aligning the vertex sets of two networks by using structural cues and it can be viewed as the generalization of the classic graph–isomorphism problem. We identify a condition for perfect matchability of two partially overlapping networks. We also propose matching algorithm and prove its efficiency theoretically and show experimentally that it performs well over real and random network data with partial node overlap. The algorithm is well parallelizable and implemented with Map-Reduce (Java/Hadoop).
- *Adaptive and incremental computation. Dynamic complexity of some problems.*
  Supervision: Professor Koch (EPFL DATA Lab). We investigated dynamic complexity classes and found a dyn-complexity of following relations: Reverse-Same-Generation relation, Transitive Closure and Natural Join.
- *Schema Covering with Concept Lattices.*
  Supervision: Dr. Miklos (EPFL LSIR). We build a polynomial algorithm for a version of a Schema Covering Problem.
- *Definability issues on structures over words and trees.*
  Supervision: Professor Selivanov. We investigated problems of definability of structures over words with a subword order and over repetition free words with the same order. We established a result about biinterpretability of these structures with a standard model of arithmetic.

Education

PhD	École Polytechnique Fédérale de Lausanne	2011 - 2017
Unaffiliated Doctoral Student	École Polytechnique Fédérale de Lausanne	2010 - 2011
Post Graduate Student	A.P. Ershov Institute of Informatics Systems	2008 - 2011
Master Student	Novosibirsk State University Department: Mechanics and Mathematics Programming Section	2005 - 2007
Undergraduate Student	Novosibirsk State University Department: Mechanics and Mathematics	2001 - 2005

4.85/5 GPA, first class degree

Professional Experience

Software engineering intern Google Zurich 2015

Analysis and reducing effect of correlations in score computation for entities reconciliation (C++)

Software developer/ Database Developer MS Team 2007 - 2010

Software products development: Windows applications, Web, data conversions, import-export, database design and logical data model. (C ♯, VFoxPro, Transact-SQL)

Passed Microsoft Exam: "Designing and Implementing Databases with MS SQL Server 2000 Enterprise Edition." (2008)

Software developer Eltex 2005 - 2006

Development various small applications: embedded http server and billing system for a phone station, a client for maintaining a calls database (FireBird, Borland C++, SQL), software for remote operations with a phone and BAS (protocols UDP, telnet, using WinAPI)

Research practice United Institute of Geology, Geophysics, and Mineralogy 2003 - 2006

Investigation of db models (Hierarchical, Network, Relational). Development of technology for cascade conversions of hierarchical data schemes (C/C++, XML, Z39.50)

Awards

Diploma (second degree) from the International Scientific Students Conference

Extracurricular

Sport: Badminton, Climbing
Permit type: B
Languages: Russian, English, French (B2), German (A1)
Marital status: married, one child