Hypergraph Analytics: Modeling Higher-Order Structures and Probabilities

A Dissertation

Submitted to the Faculty of Graduate School

at

University of Minnesota

by

Ankit Sharma

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

Professor Jaideep Srivastava

May, 2020

Acknowledgements

> In this world, there is nothing so sublime and pure as knowledge. Such knowledge is the mature fruit of all mysticism. And one who has achieved this enjoys the self within himself in due course of time.
>
> *jnan-yog* – Bhagavad Gita 4.38

In one's journey of realizing one's nature (*prakriti*) or art, one is aided by one's environment (*parivesh*). It is, therefore, a great pleasure to acknowledge the debt I owe to many people who constituted my dwelling and for their help.

Foremost are my ancestors and then my parents, who brought me into this world and provided a beautiful abode for me to think freely without expectations and minimal social encumbrances. It was my mother and her professional life within academia, which introduced me to the world of research and knowledge. I am deeply indebted to my parents, my sister, and my wife, for their patience and sacrifice for allowing me to pursue research while being so far away for all these years. I am highly beholden to my parents for upbringing me at the prestigious high school of *Maharaja Sawai Man Singh Vidyalaya* in Jaipur, which provided me with a peaceful environment and resources, enabling myself to dream in the world of learning.

I have considered Mr. Keshav Sharma as my life long mentor and role model in every sense. He is the one who has inspired me to think independently and be quintessential without being bothered by society. He is also responsible for both motivation as well as corroboration of my interest in the beauty of abstraction. And finally, it was he who introduced me to Prof. Jaideep Srivastava, my Ph.D. advisor. I, therefore, am wholeheartedly indebted to him.

Next, I would like to thank those who aroused my interest in research to the level where I could start considering it as a career choice. Firstly, I would like to thanks Prof. Amitabh Roy, who was my undergrad thesis advisor. His exemplary disciplined life dedicated to learning and research had a deep impression on me. He is responsible for introducing me to an actual in-depth inquiry and at a reference level that was unknown to me. I am also thankful to Prof. Dheeraj Sanghi for his excellent course on advanced

computer networks. I am indebted to him for exposing me to the beauty of research methodology, which he remarkably illustrated while teaching this course as a process of questioning and answering, where we naturally arrived at questions that were answered by complex network protocols. I am especially thankful to Prof. Vishwanath Sinha, as it was his reference that provided me with the opportunity of internship at *IIT Kanpur* and embrace the incredible research environment there. Lastly, I am deeply indebted to Prof. Raghubir Sharan for organizing the one of a kind course on *Ethics in Information Age*, which was my first solemn exposition to general philosophy and which I found infectious; I am most grateful to him for firing my enthusiasm for the subject whose beauty has given so much pleasure ever since. Further, I thank all of them for their confidence in me and strong references to graduate school admissions, without which this journey would not have been possible.

I am beholden to my closest friend Siddhartha Sharma, for it was only because of his persistent push I applied for graduate schools abroad and his incessant encouragement to enter the Ph.D. program. Without this, none of this could have been accomplished.

I arrived at *University of Minnesota* in *August, 2011* as a masters student, with an inclination towards research but still hesitant to delve into Ph.D. At this juncture, when Prof. Jaideep's incredible optimism and invigoration gave me the final push, I decided to embark on the road to Ph.D. Since the day we met, I have been inspired by his brave and open-minded way of building ideas as well as expressing them. He has taught me the difference between a thought and an idea, where the only difference between the former and the latter is one's belief in his thoughts and, more importantly, confidence in himself. If one has the clarity of thought, that thought becomes an idea. The more clarity, the more beautiful, the idea is. His ability to work in parallel on many ideas and multiple people with various personalities, which in tandem with his powerful administrative skills, magically gets things done like a brave-heart, is an inspirational wonder. His ability to empathize with students and their personality and provide a personalized environment and guidance to him/her is a learning in itself. I am deeply indebted to him for providing an environment that allowed me to work freely, which he very well empathized is something I craved for. And it is only because of this, I have been able to become this independent researcher, what I am today.

Most importantly, I find myself lucky that he introduced me to this intriguing mathematical object of *Hypergraph*. This object has been a ship that has allowed me to sail through the powerful connections this object makes across disciplines. It was this craving – to find abstractions and connections across different models which are on the core merely the same story told in multiple ways – that has been the primary enjoyment to me. Therefore, I am sincerely thankful to him for introducing me to Hypergraphs, and it was simply not possible for me to complete my dissertation without his valuable remarks and exemplary guidance.

I had a more substantial background in computer engineering from my undergrad and an incessant interest in computer networks. During my first couple of years at *UMN* I learned that the networks are ubiquitous. Again I am thankful to Prof. Jaideep, and his labs focus on social networks research. I was lucky enough that during the same

time, *Institute of Mathematics and its Applications (IMA)* conducted several seminars on Complex Network Analysis and Algebraic Topology applications. These seminars gave me a profound exposition to the world of complex networks and their mathematical modeling. This expo had a radical transformation of my mind, realizing that it was not just the narrow interest in computer networks, but I had a more general infatuation towards networks and their mathematical modeling. I, therefore, am deeply indebted to IMA.

During my Ph.D. I was lucky to be guided, inspired as well as to also collaborate with various important people whom I would like to convey my utmost gratitude.

I was fortunate enough to be a part of *MESH* project under the supervision of Prof. Abhishek Chandra and Prof. Jaideep. This project provided a unique opportunity to analyze hypergraph from not just algorithmic, but also systems and scalability perspective. I am especially thankful to Prof. Chandra, for his guidance, patience, and valuable criticism from the computer systems point of view, which had a significant impact on the work presented in this thesis.

During my second year, I attended the colloquium, where Prof. Ray Kuang presented his *bi-random walk* algorithm, which gave me my first research idea of networks' applications to the problem of item recommendation. I am indebted to him for this first inspiration. I applied this algorithm within my thesis as well, under his guidance.

It was an honor to have both Prof. Chandra and Prof. Kuang, on my Ph.D. committee and providing various valuable remarks over the years. Also, it was a great pleasure to have Prof. Ravi Bapna on my Ph.D. committee and for his precious remarks and intriguing discussions from the standpoint of management sciences. It was a great learning experience, and I enjoyed our interaction.

I was fortunate enough to be mentored by Dr. Mohammad Aurangzeb Ahmad, who at that time was finishing his PhD. while I just joined *UMN*. Through him, I learned some invaluable research skills and participated in the first in a complete research process. I was subtly also influenced by Prof. Nisheeth Srivastava, who was finishing his Ph.D. at UMN, for his research was strikingly original, independent, and intriguing. I am deeply indebted to him being a great inspiration to me.

I had an incredible learning and enjoyable experience while interning at *Army Research Lab, Maryland*, during the summer of 2015. I had the pleasure of working under the supervision of Dr. Ananthram Swami and the excellent mentorship of Dr. Terrence J. Moore. It was under their guidance that I worked on the formulation of hypergraphs in terms of *Hasse diagram*, which I have exploited in several parts of this thesis. I am indebted to both of them for providing me with precious remarks and criticism, which helped me achieve a solid research outcome.

Prof. Jaideep's sabbatical offered a unique opportunity in the middle of the Arabian desert at *Qatar Computing Research Institute (QCRI)*, where I spent sixteen months during 2016-17. It was an extraordinary and a lifetime experience both culturally as well as research environment wise, with a fabulous diversity of researchers. It was in Doha, where I gave a fresh look at my thesis and realized hypergraph data in various other fields, especially NLP. This realization opened up avenues for connections between

higher-order language models, including deep-learning, to that of algebraic or topological hypergraph models. At this juncture, I met Prof. Shafiq R. Joty, a research scientist at QCRI that time, working on deep learning based NLP models. It was through several fruitful discussions with him, the idea of hyperedge embeddings aroused. I am, therefore, thankful to Prof. Shafiq, and all the colleagues and friends from QCRI for such a productive and enjoyable experience.

I am, in general, thankful to all the present and past members of DMR Lab at UMN for the lovely memories and support. I would especially like to thank Himanshu Karkwal for all the help he has provided during my research and for good and tough times. I would also like to thank Prof. Xiaodong Feng, who was a visiting scholar at DMR Lab, and the excellent research collaborations we built during his visit. I would also like to thank Dr. Ayush Singhal for his help during research as well as moral support. Also, thanks to all the beautiful friends and colleagues at UMN. Specially Dhruv Sharma, Dr. Jehwan Oh and Dr. Zoheb H. Borbora, for the beautiful memories and support.

I am mysteriously indebted to the following philosophers: *Schopenhauer*; philosopher of science: *Schrödinger*, philosopher of mathematics: *Brouwer*, and philosopher of Language: *Wittgenstein*. I am greatly influenced by sociologists: *Durkheim* and *Max Weber*. My interest in improvised and drone music is partly guided by *Pran Nath, La Monte Young* and *Josh Cage*. My fascination for *Indian Classical Music* and *minimal electronic music* like deep house, techno or progressive, developed after arriving at UMN. If it were not the support of *Pandit Bhimsen Joshi* and *Kesarbai Kerkar*, as well as the wonderful minimal electro music platforms of *rts.fm* and *BoilerRoom*, working on my thesis would not have been fun and enjoyable.

Minneapolis, Minnesota ANKIT SHARMA
May 2020

O Saraswati, You are the purifier (of our Intellect), and Your Strength (of Wisdom) grows within us with Sacrificial Offerings (inner and outer), May Your presence within me make me rich in Wisdom.

Rig Ved ‖1.3.10‖

To my ancestors, my parents, my sister and my wife ....

Abstract

Data structured in the form of overlapping or non-overlapping sets are found in a variety of domains, sometimes explicitly but often subtly. For example, teams, which are of prime importance in industry and social science studies are sets of individuals; item sets in pattern mining of customer transactions are sets, and for various types of analysis in language studies a sentence can be considered as a set or bag of words. Although building models and inference algorithms for structured data has been an essential task in the fields of machine learning and statistics, research on set-like data remains less explored. Relationships between pairs of elements can be modeled as edges in a graph. However, for modeling relationships that involve all members of a set, hyperedges in a Hypergraph are more natural representations. Hypergraphs are less known graph-theoretic structure as compared to graphs. Because of this popularity graphs have been employed prolifically to model data of all kinds. Little attention is given to the fact that whether the data is naturally being generated as dyadic interactions or not. We think that much data is even deliberately converted to a graph for the sake of fitting it into a graph-based model and destroying the precious information present when it was originally generated.

This thesis describes analyzing complex group structured data from domains like social networks, customer transaction data, and general categorical data, via the lens of Hypergraphs. To do so, we propose the Hypergraph Analytics Framework, under which we shall be interested in three higher-level questions pertaining to the hypergraph modeling. Firstly, how to model higher-order hypergraph information and what kind of lower-order approximations are available or sufficient depending upon the problem at hand. This question is addressed across the thesis as we employ different hypergraph models contingent upon the problem at hand.

Secondly, we shall be interested in understanding what kind of inferences are possible over the hypergraph structure and what kind of probabilities can be learned. For this, we shall be dissecting the problem of hypergraph inference into various hyperedge prediction sub-problems and developing inference methods for each of them. We develop inference methods for both cross-sectional analysis: when we ignore the time information about group interactions into account, as well as longitudinal analysis: where we leverage temporal data. We also develop separate methods for conducting inference over observed and unobserved regions of the hypergraph structure. This variety of inference mechanisms on hypergraph structure together constitute the first part of the thesis,

which we refer to as the *Spatial Analysis* within our Hypergraph Analytics framework.

Lastly, we are interested in learning what kinds of compression algorithms are possible for hypergraphs and how effective these techniques are. Here we develop techniques to compress the hypergraph topology to lower-dimensional latent space. We shall be chiefly considering hyperedge compression or hyperedge embeddings. We examine two different embedding approaches. First, is an algebraic approach which leverages leverage the relationship between hypergraphs and symmetric higher-order tensors. Symmetric tensor decomposition techniques are then developed to learn embeddings. Second, is a neural networks based solution which employs auto-encoders regularized by hypergraph structure. Together, both these approaches constitute the second part of the thesis, which we refer to as *Spectral Analysis* within the proposed Hypergraph Analytics framework.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In group-structured data, we have multiple entities related by some form of group relation, where these groups or set relations can be overlapping or non-overlapping. Such relationships occur far more frequently in the real world than has usually been studied (Estrada & Rodriguez-Velazquez, 2005). A variety of domains involve this kind of data, sometimes explicitly but often subtly. Most noticeable among them are social group relationships. Rapid growth in the amount and richness of online interactions through social networking sites such as Facebook and Twitter, from online multiplayer games like World of Warcraft (Ahmed et al., 2011); and group communication tools such as Skype and Google Docs, are producing group or team interaction data at an unprecedented scale. Not to mention, collaboration data of scientific or software teams (Contractor, 2013) and e-mail data containing group communication logs (Klimt & Yang, 2004). These examples include data on individual characteristics (profile), their connections (social graph), and behavior (individual and interactions). Analysis of this data using the latest computational techniques is the rapidly growing area of Computational Social Science (Lazer et al., 2009).

There are other fields in which modeling relationships between a collection of entities is crucial as well, and large group-structured datasets are available. Examples include Natural Language Processing (Bengio & Bengio, 2000), Biology (Klamt et al., 2009a)

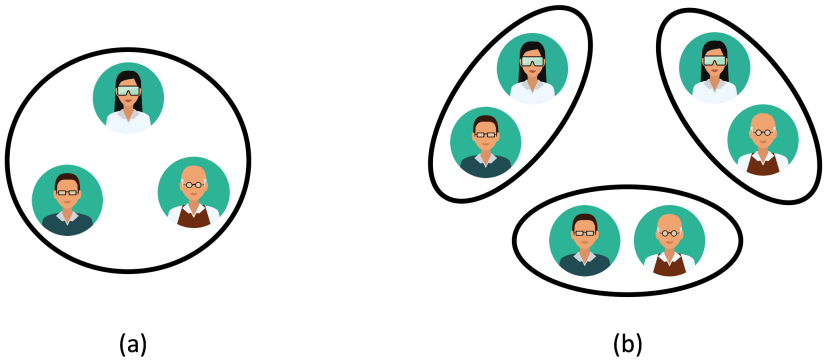

Figure 1.1: Left (a) is a hyperedges representing a collaboration between three authors and Right (b) we have three dyadic edges representing three different two person collaborations. Note the same three individuals are involved in both cases (a) and (b) but these are two different scenarios.

and E-commerce (Deshpande & Karypis, 2004).

Although building models and inference algorithms for structured data has been an important task in the fields of machine learning and statistics, research on group or "set-like" data still remains less explored. Graph theory has been a powerful tool to model relationships between entities and their properties, and its use for modeling networks is well established across multiple disciplines. The standard approach has been to consider dyadic interaction between pairs of vertices, leading to a 'simple graph' (henceforth called a *graph*) model of a network of entities and their interactions. While suitable for many purposes, we believe that a group of entities has an identity of its own, and should be modeled as such, rather than as a collection of dyadic components. We shall dwell into this argument in more detail in Section 1.2, but here we describe the intuition using a simple example. For example, a group of academics A, B, and C co-authoring a paper is not the same as three papers co-authored by A and B, B and C, and A and C, respectively. Specifically, the latter fails to model the *meeting of minds of the three academics* around an idea (see Figure 1.1). This example illustrates that the right mathematical object for modeling group interactions is not a *graph* but a *hypergraph* (Berge, 1976; 1984). A hypergraph can be defined as a tuple $H = (V, E)$,

where V is the set of entities called nodes or vertices in the network and E is the set of subsets of V, called hyperedges, representing relations between one or more entities (see Definition 12).

Hypergraphs are less known graph-theoretic structure as compared to graphs. Because of this popularity graphs have been employed prolifically to model data of all kinds. Little attention is given to the fact that whether the data is being naturally generated as dyadic interactions or not. We think that much data is even deliberately converted to a graph for the sake of fitting it into a graph-based model and destroying the precious higher-order interaction information present when it was initially generated. Recently, with the rise of the deep learning era, algorithms catering to higher-order data, specifically in computer vision and NLP, are now being introduced to research communities that focus on graph-theoretic or algebraic algorithms, with Deep Learning being a universal language. With this confluence, the need to model higher-order is emerging in the graph as well as algebraic communities in the form of hypergraphs and their algebraic representation as tensors. Recent advances in machine learning have opened up avenues to build models that can capture higher-order information in the structure and train them using large amounts of data. Therefore, the realization of the presence of these higher-order structures and the growing need to incorporate this structural information into the analysis is increasingly important.

In this thesis, we put forth hypergraph based machine learning models while aiming to capture the higher-order information contained within groups via the hypergraph model. Together, with the growing need to predict future behavior based on an analysis of the past, in several group-structured data oriented applications (as described in Section 1.1), this creates an exciting opportunity for developing computational techniques for hypergraph based predictive modeling.

In this chapter, we present the overall vision for the proposed thesis. Specifically, in Section 1.1, we first describe a number of motivating applications where entities interact in groups with each other in real-world settings. Next in Section 1.2 we provide an overview of the major graph based models for groups, eventually leading to the concept

of hypergraphs and why they are an attractive model for studying the properties of the motivating applications. Finally, in Section 1.3, we describe the proposed framework for analysis and prediction over large-scale evolving hypergraphs as well as the thesis overview.

1.1 Motivating Applications for Hypergraph

Our research on hypergraph predictive modeling is motivated by several real-world applications that have complex multi-entity interactions. As compared to dyadic (pairwise) interactions or relationships, groups of entities have more complex interactions, and they often evolve in complex ways, as illustrated by the following application classes and their representative examples.

Social networks and media:

The growing popularity of social networks, such as Twitter, is generating tremendous amounts of data. There are 330 million monthly active users and the average exchange of tweets per day is noted to be 340 million. Each user account is followed by a group of several users, who receive any message posted by the user being followed. This following relationship is dynamic and evolves over time, as the follower-followed relationships change. Twitter is often used for viral marketing through the social network, where predicting the information flow through these groups can help optimize the process. In addition, identifying rumors and/or rumor spreaders can help control an undesirable phenomenon. We show how Twitter information spread can be modeled as directed hypergraphs.

Real-world team collaborations and interactions:

The digital era is opening up unprecedented opportunities for collaboration both in academic network as well as in organizations. An increasing variety of social media-social networking sites, group communication tools like Google Hangout, Skype, etc., is

enabling individuals to team up in ways that would not otherwise be possible like joining more teams, working in multiple teams in parallel (Mortensen et al., 2007), working in distributed (Hinds & Mortensen, 2005) and more diversified (Daspit et al., 2013) teams. How such self assembling teams and their network evolve over time affects their productivity (Ancona & Caldwell, 1992) and innovation (Taylor & Greve, 2006) both. Therefore, predicting the evolution of network of teams (hypergraph) and predicting performance of these teams (hyperedges), is critical for both organizations as well as funding agencies for academic research. There are several datasets that are commonly used for studying such explicit collaborations: U.S. Patent Office (USPTO) collaboration network (Subbian et al., 2013), DBLP (Tang et al., 2007a) and PubMed. At the same time, there are implicit team interactions, such as referral networks. Consider a physician referral network, where the patient is referred by a physician to another for a specific medical reason. Often, these doctors collaborate in teams and understanding their team interactions could significantly minimize the overall referral costs to the insurance company. Consider a scenario, where a team of doctors who form an implicit team to resolve a particular patient's medical situation, and where a subset of them may be out-of-network resulting in significant costs to the insurance companies. Understanding such subset behaviors in teams is an important problem in healthcare, especially in the context of accountable care organizations (ACOs) and managed care.

Goal-oriented teamwork in a rapid-response setting:

Rapid-response teams, e.g. firefighters and police units, are long term stable groups often lasting many years, and people develop deep social bonds. A common practice activity is to self-organize into a team, e.g. 'action team', and enter into a competition, either against other teams, or in a computer simulation. Such activities provide the opportunity to achieve success, socialize, and improve skills - both individual and collaborative. Thus, data collected from online team games serves as a large-scale experimental platform to understand both individual and team interactions.

Massive multi-player online games and virtual worlds:

Massively multi-player online (MMO) games have shown signicant promise as a crucible for studying behavioral sciences at a ne granularity. We describe one such game, EVE Online, as an example. EVE has space-based combat involving ships con-gured with equipment allowing them to damage, tank, or heal. EVE has an open, sandbox style of play emphasizing player-versus-player (PvP) combat rather than end-game player-versus-environment (PvE). The social organization of players in EVE is complex. At a functional level, EVE contains eets consist-ing of two to several dozen players engaged in combat or mining tasks. In EVE, players self-assemble into player corporations for more complex purposes such as PvP combat, claiming sovereignty and con-trol over remote star systems, and developing supply chains to manufacture complex and highly protable products. However, player corporations can also ally themselves with other player corporations and create supra-corporation organizations called alliances. Alliances may exist to police shared space, protect man-ufacturing centers, or perform other collective actions. In addition to alliances, corporations can also set standings for other corporations to indicate the extent to which their members can be trusted or should be avoided. Thus, EVE provides a very rich domain in which to analyze and model corporations and alliances as an ecosystem of teams.

Capturing higher-order relations in Knowledge Graphs:

Knowledge graphs (KGs) are multi-relational graphs such as Freebase (Bollacker et al., 2008), Google Knowledge Graph (goo), ConceptNet (Speer & Havasi, 2012) or Yago (Suchanek et al., 2007), where each entity of the KG represents an abstract concept or concrete entity of the world and relationships are predicates that represent facts involving two of them. These are directed graphs whose nodes correspond to entities and edges are in the form of triples (*head*, *label*, *tail*), each of which indicates that there exists a relationship of name label between the entities head and tail. Relationships, however, can be higher-order or more complex like nested, not necessarily binary. For example,

"Jacob flew from London to Seattle." can be modeled in a KG as two independent binary triples (*Jacob*, *flew from*, *London*) and (*Jacob*, *flew to*, *Seattle*). Instead, a more informative way would be to model it as a single 4-ary tuple (*Jacob*, *flew*, *from London*, *to Seattle*). Need for retention of this higher order information has been realized both in knowledge extraction (Christensen et al., 2011; Mausam, 2016) and NLP (Kalchbrenner et al., 2014) communities. Further, methods to extract such n-ary tuples already exist (Christensen et al., 2011). We believe hyperedges are an excellent mechanism to model such higher order relations. Consider the sentence *"Early scientists believed that earth is the center of the universe"*, from which we extract the nested relation $<$*Early scientists*, *believed*,$<$*earth*, *is the center of*, *the universe*$>>$. This can be modeled as two nested directed hyperedges: one with *earth* as tail node and *universe* as head node with *is the center of* as the type or attribute of the hyperedge; and the second being a *believed*-type hyperedge with *Early scientists* as tail node and *earth* and *universe* in the head set. Such Knowledge Hypergraphs (KHG) can undergo the purification and completion processes via embedding or link prediction techniques similar to those of KG (Bordes et al., 2013; Trouillon et al., 2017; Yang et al., 2014), except that in case of KHG we make use of hyperedge embeddings for predictions. As described in Section 3, these techniques form the core of this proposal.

Customer analysis and recommender systems:

We live in the era of online shopping, which generates massive data about millions of customers and thousands of products from a large number of categories. For example, (www.alibaba.com), the world's largest retailer orchestrated the world's biggest online shopping day, with 500 million shoppers within 24 hours, where its own sales reaching over US$ 38 billion on 11th November, 2019 (ali). Customer behavior is captured by the baskets (sets) of items bought together. Analyzing customer behavior was one of the first motivating applications for the field of frequent pattern mining (Agrawal et al., 1993). Sets of frequently bought items have a higher-order correlation which can be

leveraged for recommending items to customers while shopping online (Christakopoulou & Karypis, 2014; Deshpande & Karypis, 2004). From a hypergraph modeling standpoint – these highly-correlated itemsets can be modeled as hyperedges and the hypergraph topology can be used as a regularization for the recommendation objective function. Not only can these higher-order correlations help better recommend individual items for a given user, but can also be employed for recommending a whole group of items (slate) – also called as slate recommendation (Ie et al., 2019a;b) within recommendation literature. Slate recommendation can be used for example, for recommendation of what to wear for a party, the system can recommend entire sets of items, like a shirt-pant pair with a jacket along with matching shoes and belt; and there can be multiple such styling options offered to the customer given his budget. From a hypergraph perspective these sets of items or slate is nothing but a hyperedge within the hypergraph of item sets. Further, by assigning probabilities to hyperedges we are essentially solving the slate recommendation problem. Another related problem is to assign probabilities to association rules between item sets, as each association rule is a directed hyperedge. Lastly, predicting how purchase patterns will evolve over time can be naturally modeled as one of predicting hypergraph evolution.

Text Processing:

Increasing use of Internet has resulted in massive amount of online *textual* data. Analyzing this data is of prime importance to a variety of fields like Marketing: market research, social media analysis, voice of customer; Business: competitive intelligence, voice of employees, ; and others. Depending upon the task at hand, different text processing techniques may incorporate the structural information in text. For certain tasks we can consider a sentence or a paragraph as a set of words (hyperedge), rather than a sequence of words. This is specially true for IR tasks that either require position invariance with respect to the words or which do not depend on the specific ordering of words. Such tasks include, but not limited to, Sentiment and Relational Classification

of sentences, Topic Modeling and Topic Segmentation of a groups of sentences, and Automatic Text Summarization. All of these are examples of predictive tasks that can be undertaken using hypergraph as a model of text. Note that treating sentences as sets of words loses the sequential information within a sentence, but the hypergraph (overlapping sets of words) topology as well as the information about sequence of sentences (sequence of hyperedges) persist.

Biology:

Various kind of biological networks arise across various sub-disciplines as well as applied field like medicine, bio-informatics, computational biology, etc. Understanding topological properties of these networks is a key problem within these disciplines. Graphs remain the most popular model choice for these networks. However, more often than not, these networks are more complex than can be captured using graphs. A lot of these networks have group structured relationships which can be more naturally modeled as hypergraphs. For example the discipline of phylogenetics deals with studying phlogenetic trees or networks (Avise, 2006). These networks, also called *tree of life*, define relationships between various biological species depending on their common ancestry. In such networks an ancestor specie relates a set of child species; this group of species can be considered as a hyperedge and the entire phylogenetic network can be naturally represented as a hypergraph (Baar et al., 2019). In these trees the parent nodes that are mode near the leaf species, relate the leaves more strongly as compared to the parents higher up in the tree. Such information can be encoded as hyperedge weights. See section 1.2 for a mode detailed example of phylogenetic group data.

Another example are the Protein-protein Interaction (PPI) networks which capture the physical interaction (transient or stable) between proteins. Proteins play a principle roles in biological function, their interactions determine molecular and cellular mechanisms, which control healthy and diseased states in organisms (Safari-Alighiarloo et al., 2014). Knowledge gained from studying these networks can be translated into effective

diagnostic and therapeutic strategies. However, the methodologies that are adopted, like TAP (Gagneur et al., 2004; Gavin et al., 2002), to obtain these interactions often output "protein complexes" which can have more than two interacting proteins. These protein complexes – which are a set of proteins – can be model more effectively as hyperedges and the PPI network as hypergraph (Fionda, 2019; Klamt et al., 2009b).

Further, cellular functions within organisms are a synergy of interactions between various modules within cellular networks (Pereira-Leal et al., 2007). Like PPIs are an example of cellular networks and protein complexes are modules within PPI. These modules are dynamic self-assembling functional units that are constantly evolving and interacting in both intra-modular as well as inter-modular manner (Levy et al., 2008; Marsh & Teichmann, 2015). Therefore, understanding the evolution of these networks is key in understanding the cellular functions. Also it is important for understanding progression of diseases caused by evolution of incorrectly folded proteins for example. Modeling these modules (like protein complex) as hyperedges and the cellular network (like PPI network) as hypergraph, we can employ predictive hypergraph models to infer evolution of these networks.

1.2 Modeling Network of Groups: Graph versus Hypergraph

These examples illustrate a broader picture of groups that takes their dynamic nature into account and views them as part of a complex system of sets and entities operating as an ecosystem. Groups in such systems have one or more of the following properties:

- *The groups and their ecosystems are often large and complex*, consisting of a large number of entities, groups, and interactions. The groups often address highly complex problems involving a large amount of diverse information.

- *The groups interact with each other in a variety of ways*, requiring interactions beyond their core entities and necessitating external linkages to other groups and

units. Thus groups often combine or link with other groups in coordinated behavior. Effects on groups in one part of the ecosystem bleed over onto other groups.

- *The groups evolve over time.* Group membership shifts over time, and their boundaries may be ill-defined. Some groups may form for limited time and go out of existence when their collaboration purpose is complete. Entities can move from one group to another relatively fluidly.

In summary, group interactions involve subsets of entities which are usually interconnected and overlapping with one another, and *a network perspective is particularly useful for analyzing them.* This leads to our **central question**: *What is a good model for this network of groups?*

But, before we dive into elucidating this question, let us first attempt to abstract out the different forms of group-structured data that are gathered from the various real-world applications like the ones discussed in Section 1.1. Records of group interactions can be available both sometimes implicitly otherwise explicitly, and at a various temporal granularity. We divide the various forms of group data into three different kinds:

- First, is the **group event data** and is in the form of logs of group interaction events. More precisely, consider the set $V = \{1, 2, ..., n\}$ of n entities and a set of m groups $G = \{g_i | i \in \{1, .., m\}, g_i \subseteq V\}$. We define group interaction logs as a set of tuples $L = \{(g_i, t_j) : i \in \{1, 2, ...m\}, j \in \{1, 2, ..., T\}\}$ where the tuple (g_i, t_j) implies the group event when the entities of the i^{th} group g_i interacted at time t_j. Note that the same group g_i can occur at several time instances, and also multiple different groups can be active at the same time instance t_j. Such group event logs are found, for example, in research publication records containing various groups of authors publishing different research articles (see Figure 1.2), or different groups of items purchased as recorded in an item basket transaction database of

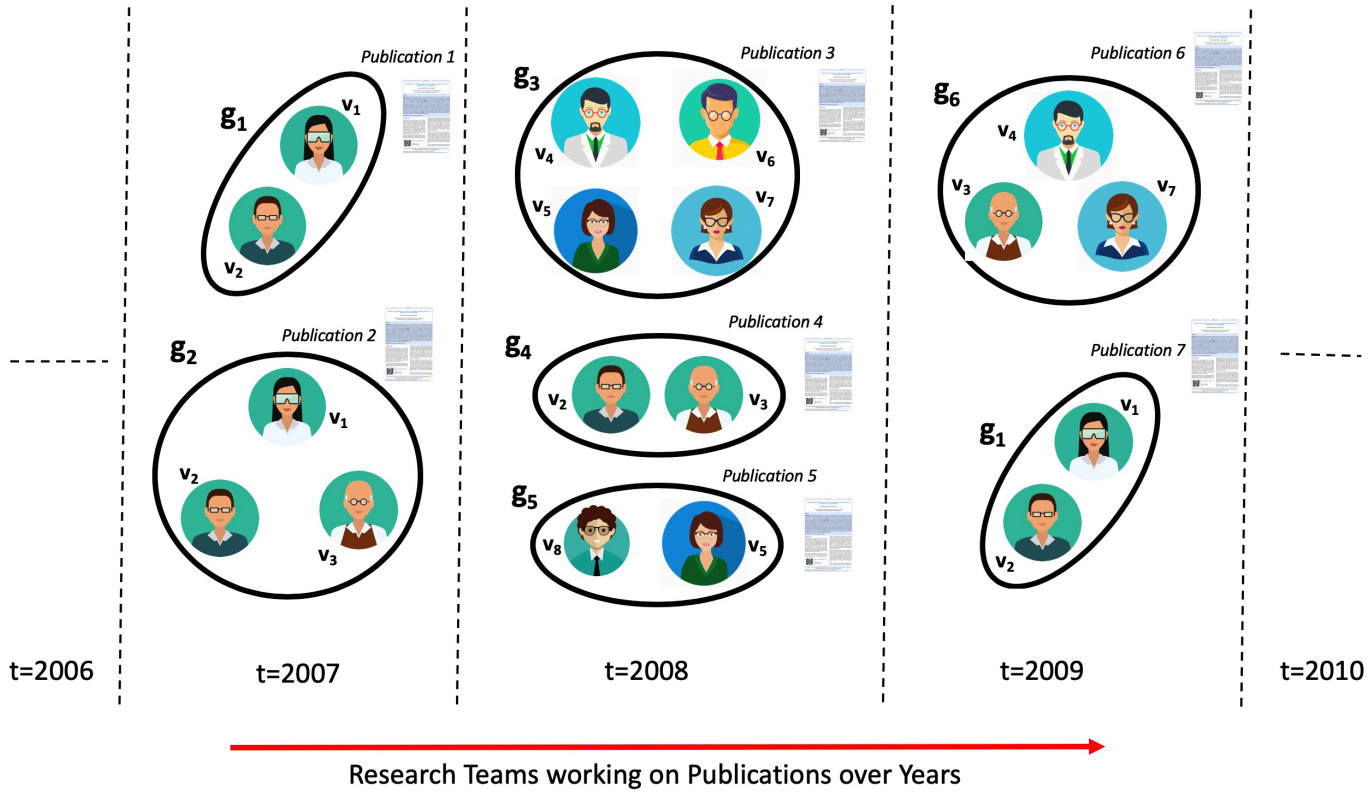

Figure 1.2: Example illustrating a subset of research publication records containing various groups ($G = \{g_1, ..., g_6\}$) of authors ($V = \{v_1, ..., v_7\}$) publishing different research articles (Publication $1 - 7$) across three years. The group interaction logs for this subset of publication records is the set of tuples $L = \{(g_1, 2007), (g_2, 2007), (g_3, 2008), (g_4, 2008), (g_5, 2008), (g_6, 2009), (g_1, 2009)$.

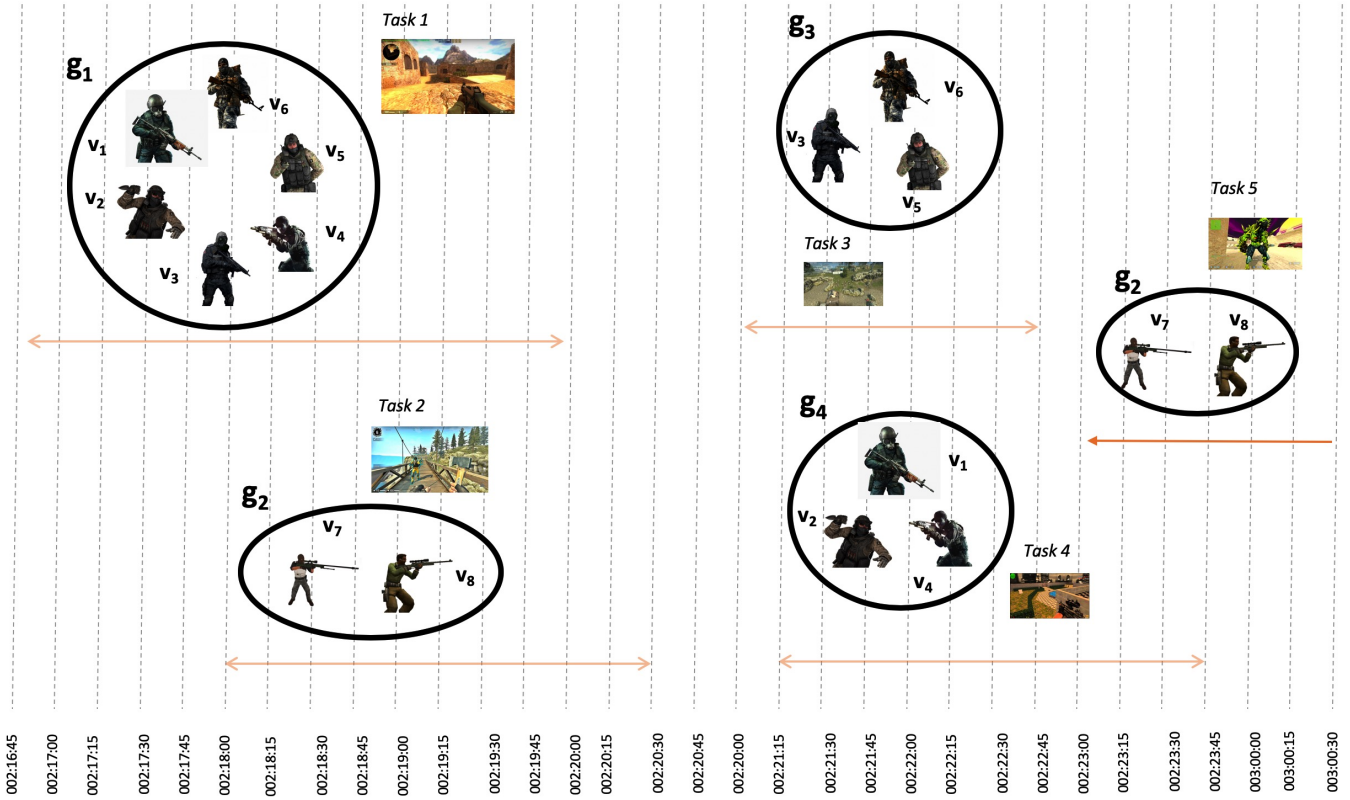

Figure 1.3: Example illustrating a part of online gaming data logs. Showing various groups ($G = \{g_1, ..., g_4\}$) of gamers ($V = \{v_1, ..., v_8\}$) performing different tasks (Tasks $1 - 5$) within game. The group data could have been directly logged as group level logs or aggregated from individual action logs. The data is curated at 15 minute granularity and interaction logs is the set of tuples $L = \{(g_1, 002:16:45 - 002:20:00), (g_2, 002:18:00 - 002:30:00), (g_3, 002:21:00 - 002:22:45), (g_4, 002:21:15 - 002:23:45), (g_2, 002:23:00-)\}$.

some online e-commerce website. Note that such group event logs are not always explicitly available; for example, in online gaming data (see Figure 1.3), often the event logs maintained are those of individual player actions. But group events can be extracted by, for example, looking at all the players that simultaneously entered the same map at the same level within the game, and further, there might be chat logs where this group was found communicating (Ahmed et al., 2011). Another subtlety to mention is that often, the existence of a group is *inferred*. For example, in the case of research publication records, what we observe is the record of a publication tagged with a set of author names. From this record, we infer or assume that this set of authors must have worked together in the past. And these past interactions, in the end, culminated in this research publication. This scenario is in contrast to online gaming data, where we are recording the minute by minute details of the actions and interactions happening within a team. This conclusion leads us to another remark that we can observe group events at varying *temporal granularity*. In the case of online gaming teams, the granularity is of a minute, whereas in the instance of research publication, we only observe the final research outcome: the publication. Such granularity can, therefore, determine our confidence in how closely is the data portraying the underlying group phenomena.

- Second, is the **group relationship data** which contains relationship information between multiple entities. These relationships are often either derived from other sources like from the group event data itself, which we just described. For example, if we observe a specific subset of items that are found together in baskets of a large number of users in an e-commerce website database, we can treat this subset as a group relationship (see Figure 1.5). We can also associate a weight with this relationship like the number or percentage of baskets that contain this subset. These weights reflect the frequency of re-occurrence of a subset and, therefore, can be used as a degree of belief or trust as to whether the given subset has a group relationship in reality. Let us take another example: assume we have

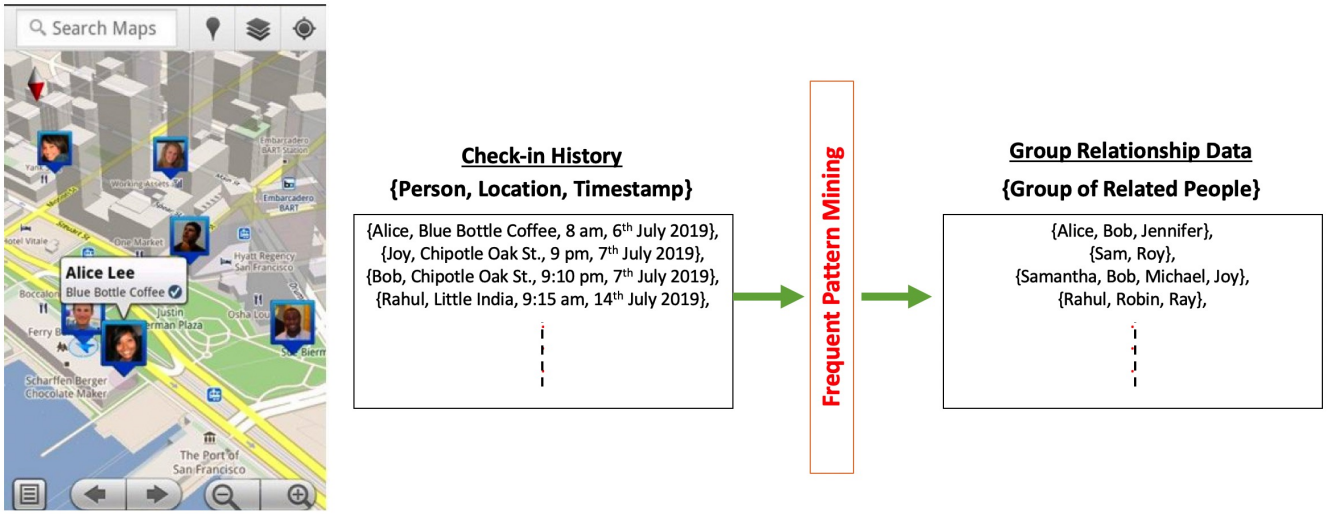

Figure 1.4: Example illustrating extraction of social group relationships from user's restaurant check-in data. Left is the user's check-in history maintained by some mobile application. History includes for example the user info, restaurant, timestamp. Frequent pattern mining techniques are then used to obtain subset of people who happen to be seen together within a short window of time at various restaurants over a while. These co-located set of individuals is then treated as a group.

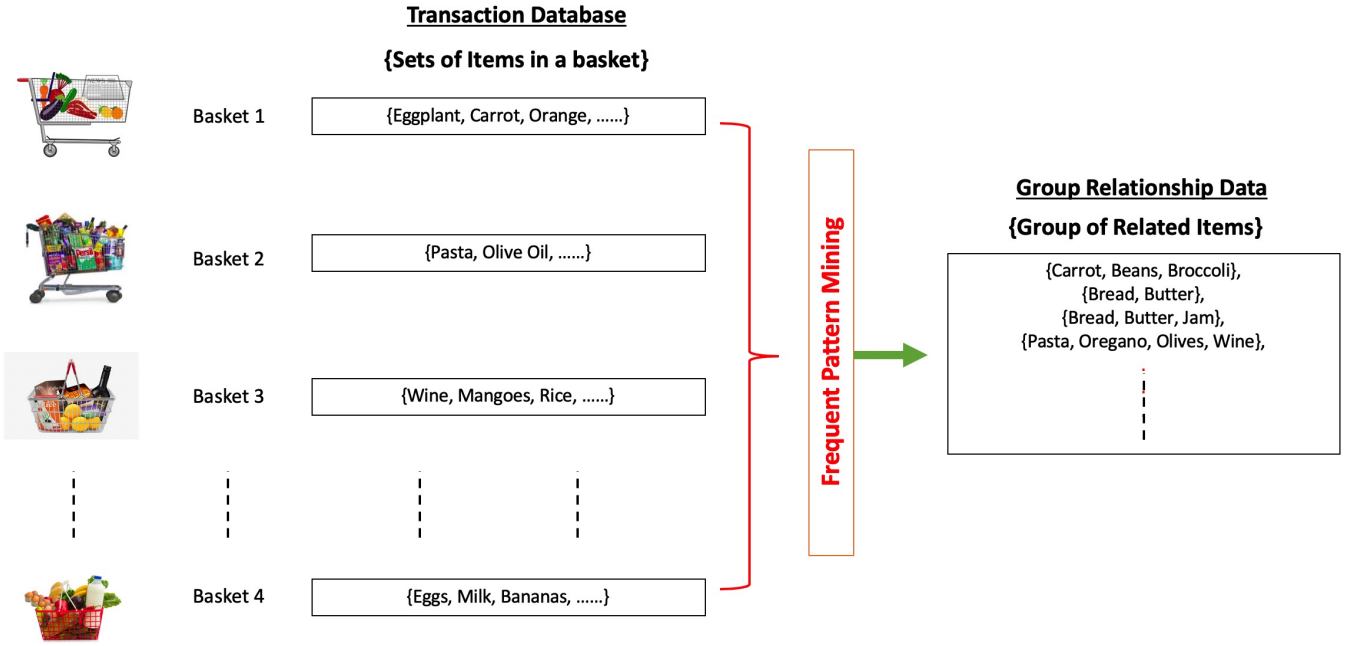

Figure 1.5: Example illustrating grocery baskets of various customers (transaction database) which is then mined to obtain frequently bought together item sets or groups of related items (via frequent pattern mining).

the records of individuals who checked-in at various restaurants in a city over a few months or a year (see Figure 1.4). We can then apply ***frequent pattern mining*** [1] to find out subsets of people who checked-in within a window of, let's say one hour, at any particular restaurant. If this set of people happens to be seen together within a short window of an hour and at various restaurants over a while, we can say with high certainty that this set of individuals is part of the same social group in reality as well. The co-location frequency or weight, thus, acts as a measure of belief.

Frequent patterns within a group event data, however, is not the only way group relationships can be extracted. In several cases, we can derive both the relationships as well as their weights via ***domain knowledge*** itself (see Figure 1.6). For example, within the home decor domain, the set S_1 ={bed, mattress, mattress-protector}, from a utility perspective, represents a meaningful relationship as the items in this set are complimentary and are likely to be purchased together. Also, one can model the weight associated with this set using domain information. For example, the weight in the case of the set S_1 can be a function of the brands of the three entities in it. Different consumer psychology theories might dictate if people tend to buy the same brand bed as well as a mattress, or they might incline for a variety of brands, with the most popular brand for each kind of item in the set. The more a group like S_1 tends to satisfy a particular cognitive theory, the more weight the weight function should output. Thus, a domain knowledge derived weight function can model the importance of a relationship. We just gave an example of an item basket, but such domain-based predetermined relations and their importance exist in several other domains. Like related set of biological species based on common ancestry is a key topic in phylogeny (Avise, 2006; Baar et al., 2019), a sub-discipline within biology (see Figure 1.6); or a related set of proteins in bio-informatics called protein complexes (Klamt et al., 2009b); or in-

[1] Remark, extracting frequent patterns (or frequently reoccurring subsets) from a transaction database, is the central problem studied in a subarea of computer science called frequent pattern mining.

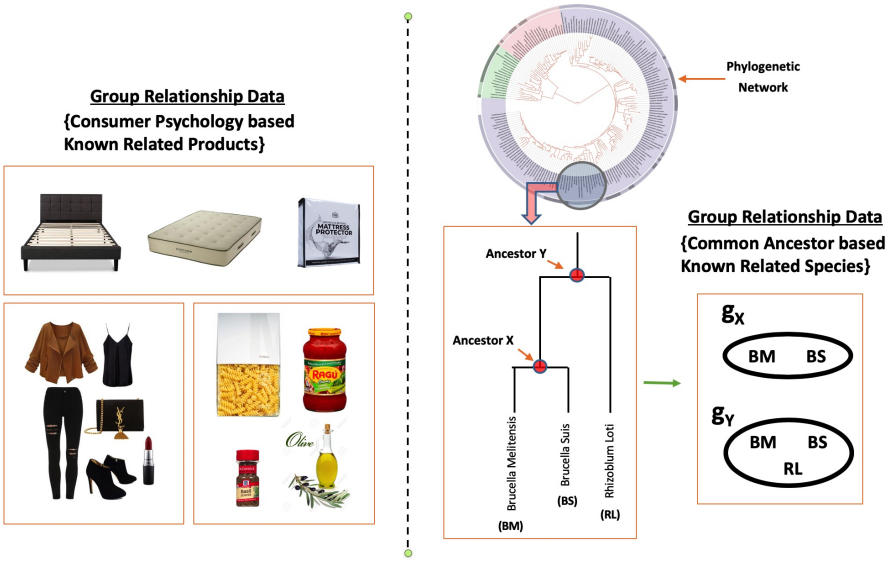

Figure 1.6: Left are a few example set of products which are related utility or consumer psychology wise: bed set, an entire women attire including accessories and a set of pantry items shopped along with pasta. Right is an example from phylogeny in biology. A phylogenetic network or tree shows relationship between various species based on genetic or physical characteristics. These trees help us understand how new species form from common ancestral species. At top right is the entire tree or network. We zoom in on a small sub-tree shown in the rectangle box. What this particular tree tells us is that specie BM and specie BS are more closely related to each other than either specie is to specie RL. The reason is that specie BM and specia BS share a more recent common ancestor than BM and BS do with specie RL. An ancestor node, therefore, corresponds to a group relationship, which we extract as a group, as shown in the bottom right.

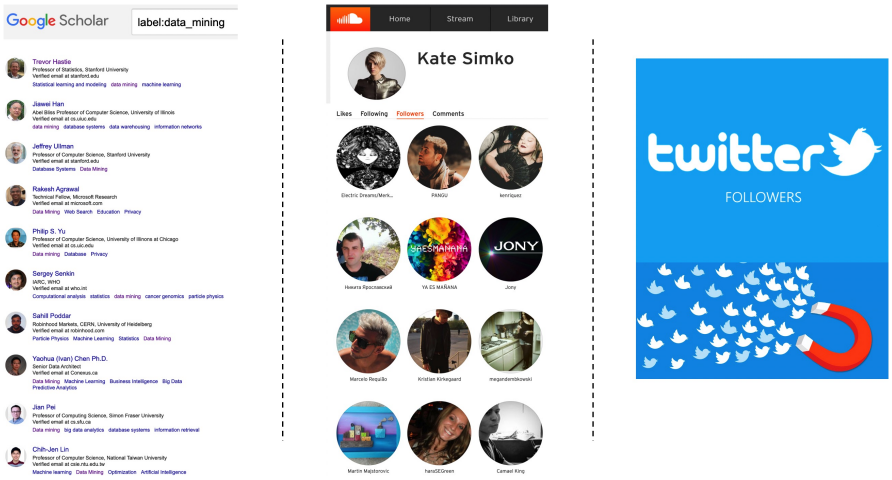

Figure 1.7: This figure illustrates a few example of social category data. Left: Shows a list of researchers with "data mining" as their research interest. Image displays the top-10 results from Google Scholar [2] which maintains indexes of researchers with respect to research categories. Unlike research teams which are highly interactive groups, these set of researchers are simply interest based groups which might be interacting at all. Middle: Shows a list of followers of a famous American DJ artist "Kate Simko" on the music platform called Sound Cloud [3]. Again these set of followers might be unaware of each other but still they are tagged within Kate Simko's category. Right: Shows a how a popular or magnetic profile on the famous follow-followee platform called Twitter [4], attracts thousands or millions of followers. All such set of followers of a popular profile is an example of category based group.

teracting set of molecules in a macromolecular assembly which is a key component of supramolecular chemistry (Ariga et al., 2008; Lehn, 1990).

- The third is **category data**, where we aggregate entities into sets if they have the same attribute values for one or more essential attributes. A set, in this case, represents a category and entities present in it are by virtue of their attributes alone. However, these entities might not be interacting with each other as such. Notice, this is in contrast with the group relationship data as well as group event data where a group is derived or observed, primarily, as a meaningful interaction between a set of entities. For example, a community of all the researchers who publish at a particular publication venue or belong to a particular sub-discipline

Figure 1.8: This figure exhibits some examples of category data from the retail products domain. Left: Depicts a few bag images associated with the category "women's handbags". These bags form a group simply because of their association with product category: women's handbag. This group of bags is hence, a category group. Middle: A category group of the brands who produce leather products. Right: Another instance of category group depicting images of a set of shoes belonging to category: Men's Shoes.

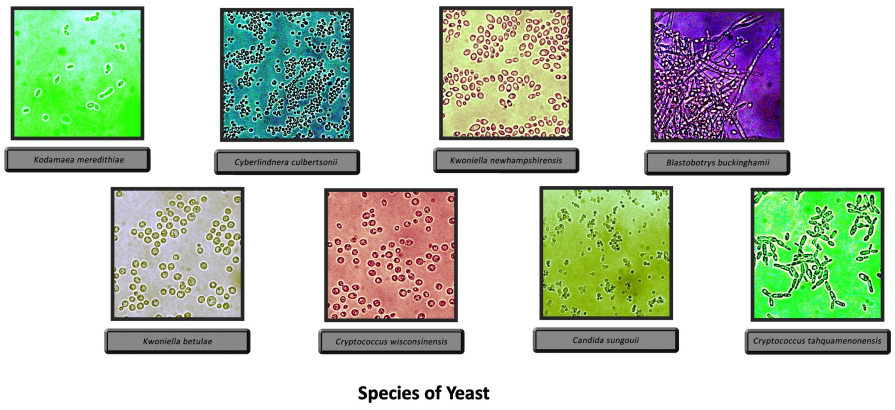

Figure 1.9: This figure displays a few example species of yeast out of the 1500 or more known species. All these hundreds of species together form a category group as they are simply belonging to category: yeast. Apart from the knowledge of this category we do not know about any meaningful relationship between these yeast species. However, if we try to bring in information from for example phylogenetic tree between these species, in that case we have more actual relationship based on ancestry between these species and we rather treat it as a group relationship data and not category data.

represents a category (see Figure 1.7). Researchers in such a community might not be interacting with each other together for a joint group task such as publication. Such a community is different from research teams working on a publication while interacting together over some time. Other social community examples are a set of users who have subscribed to a particular online podcast channel or who follow an accessible Twitter [5] account. The group of all the metals in the periodic table, or the various species of yeast (see Figure 1.9) and the product category of women's handbags on an e-commerce website (like Amazon [6]), can be other examples (see Figure 1.8).

Remark, although we have provided the above categorization of group data, we believe that a complete analysis and taxonomy of such data is a significant research direction that requires an independent research effort. We, therefore, consider building a general taxonomy for group-structured data, as out of the scope of this thesis. However, for the specific domain of sociology, we do have developed a taxonomy for social groups and their hypergraphs. For interested readers we point to Section 2.5 of Chapter 2. For this thesis and the discussions within it, however, we find the above categorization suffice enough.

Within the three kinds of group data: group event data, group relationship data and category data, which we described above, we observe the following different abstractions for groups.

Definition 1 *(**Static Groups**) Consider a finite set $V = \{1, 2, ..., n\}$ of n entities. We call a subset $g_i \subseteq V$ of this set a group. We consider and a finite set of m groups $G = \{g_i | i \in \{1, .., m\}, g_i \subseteq V\}$. Then the tuple (G, V) together define a collection of **static groups**.*

Definition 2 *(**Weighted Groups**) Consider a finite set $V = \{1, 2, ..., n\}$ of n entities. We call a subset $g_i \subseteq V$ of this set a group. We consider and a finite set of m groups*

[5] www.twitter.com
[6] www.amazon.com

$G = \{g_i | i \in \{1, .., m\}, g_i \subseteq V\}$. *We also associate a weight* w_i *for each group* g_i *and a weight function* $w(x), \forall x \in G$, *such that* $w_i = w(g_i)$. *Then the tuple* (G, V, w), *define a collection of* ***weighted groups***.

Definition 3 *(**Temporal Groups**) Consider a finite set* $V = \{1, 2, ..., n\}$ *of* n *entities. We call a subset* $g_i \subseteq V$, *a group. We consider a finite set of* m *groups* $G = \{g_i | i \in \{1, .., m\}, g_i \subseteq V\}$. *We also consider the temporal activity pattern these groups. We define a group interaction event when the entities of the* i^{th} *group* g_i *interacted at time* t_j. *In terse, we also refer to it as group* g_i *was* ***active*** *at time* t_j. *We then define group interaction logs as a set of tuples* $L = \{(g_i, t_j) : i \in \{1, 2, ...m\}, j \in \{1, 2, ..., T\}\}$ *where the tuple* (g_i, t_j) *implies the group activity or interaction event. Then the tuple* (G, V, L) *together define a collection of* ***temporal groups***.

We use the adjective "static" to differentiate static groups from temporal groups, which are "dynamic" in the sense that their specification includes time (or temporal) information. Also, notice that weighted groups generalize static groups by adding weights to each group. Similarly, temporal groups also generalize static groups by adding temporal information about group activity. We can, therefore, also have ***weighted temporal groups***, which generalize all the three above, and given by the specification: (G, V, w, L).

We think that these four group abstractions are quite appropriately suited for modeling the three types of group data we had described before. For example, the static groups can easily model the category data, the weighted groups are sufficiently appropriate for capturing the weighted group relationship data, and the temporal or weighted temporal groups are reasonably general and suitable for modeling the dynamic group event data.

These group abstractions, therefore, clearly define the various group relationship objects we aim to study and model. We now jump back to answering our central question: What is a good model for this network of group relationships? As noted before, Network-based models that employ Graph-theoretic representations have been

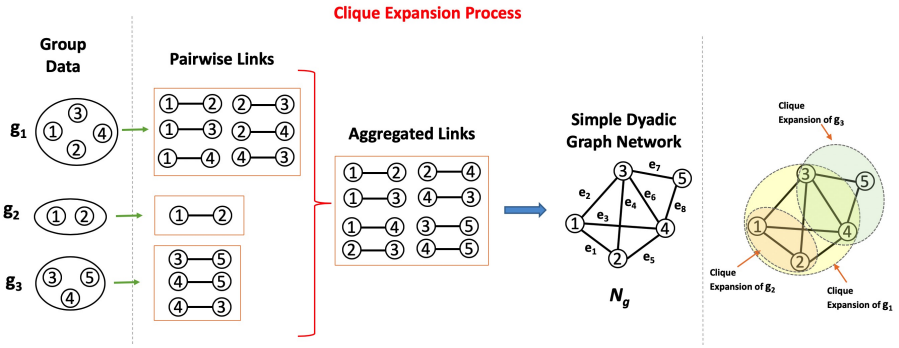

Figure 1.10: Leftmost is a collection of static groups (G, V) where we have a set of five entities $V = \{1, 2, 3, 4, 5\}$ and a collection of three groups: $G = \{g_1, g_2, g_3\}$ where $g_1 = (1, 2, 3, 4), g_2 = (1, 2), g_3 = (3, 4, 5)$. We then clique expand each group by making pairwise link between each pair of entities in that group. All the sets of pairwise links obtained from each group are then aggregated (i.e. taking union). Each pairwise link in the aggregate set is treated as a dyadic edge and a simple dyadic graph network, N_g, is obtained. The entire process of converting static group data, (G, V), to the clique-expanded graph, N_g, is called the process of *clique expansion*. The rightmost figure highlights this process by indicating which group resulted into which clique within the final graph.

used, prolifically across disciplines, to capture relationships between entities. We will, therefore, now consider adopting various objects from graph theory to model the group abstractions.

Let us consider a collection of static groups (G, V) where we have a set of five entities $V = \{1, 2, 3, 4, 5\}$ and a collection of three groups: $G = \{g_1, g_2, g_3\}$ where $g_1 = (1, 2, 3, 4), g_2 = (1, 2), g_3 = (3, 4, 5)$, between them (shown at the leftmost in Figure 1.10). One chain of thought could be to link each pair of entities if they are part of one or more groups. Accordingly, in the case of group g_1, we connect each of the six pairs of entities with a link. If we apply the same process to the rest of the groups and aggregate all the pairwise links, we end up in the network structure as shown at the rightmost in Figure 1.10. This resulting network is nothing but a simple undirected graph consisting of the vertex set V and a set of dyadic edges $E_g = \{(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4), (3, 5), (4, 5)\}$. Each $v_i \in V$ represents an entity in the network and each edge $e_j \in E$ represents a relation between two entities (vertices).

More formally we define a graph as follows.

Definition 4 ***Dyadic Graph (or Simple Graph):*** *Formally, an undirected simple graph can be defined as a tuple* $N_{sg} = (V, E_g)$*, where* $V = \{v_i\}_{i=1}^{n}$ *is the set of vertices* (v_i) *and* $E_g = \{e_j\}_{j=1}^{m}$ *is the set of dyadic edges* $e_j \subseteq V$ *such that the their cardinality is two:* $|e_j| = 2$ *(Berge, 1976).*

The entire transition from the set of overlapping groups: (G, V) (shown leftmost in Figure 1.1), to the dyadic graph network: $N_{sg} = (V, E_g)$ (rightmost in Figure 1.1), is referred to as the process of *clique expansion* (Agarwal et al., 2006a; Zien et al., 1999). We formally define, both a clique as well as clique expansion as follows:

Definition 5 ***Clique:*** *A clique,* C*, in an undirected graph* $N_{sg} = (V, E_g)$ *is a subset of the vertices,* $C \subseteq V$*, such that every two distinct vertices are adjacent. This is equivalent to the condition that the induced subgraph of* N_{sg} *induced by* C *is a complete graph.*

Definition 6 ***Clique Expansion:*** *Consider a collection of static groups,* (G, V)*. The clique expansion algorithm constructs a simple graph* $N_c = (V, E_c)$ *from this collection of static groups G(V,E) by replacing each set* $g_i \in G$ *with an edge for each pair of vertices in the set:* $E_c = \{(p, q) : p, q \in g_i, g_i \in G\}$*.*

This concept of a complete subgraph or "clique" was first used in mathematical psychology to graph theoretically model a group or clique of people (Luce & Perry, 1949). In our example we model each set g_1, g_2 and g_3, as highlighted in the central figure in figure 1.10, as a clique and perform a clique expansion of each set. Now let's carefully analyze the clique-expanded graph, N_c, in figure 1.1, that we arrive after clique expansion. We ask ourselves the question: "what all can this graph tell us about the original collection of static groups from which it is derived"? It is easy to observe that each edge in the derived graph indicates if the connecting vertices (entities) were present in at least one group in G. Thus, the only information retained in the graph is the co-occurrence information viz. if a pair of entities occurred together in any group

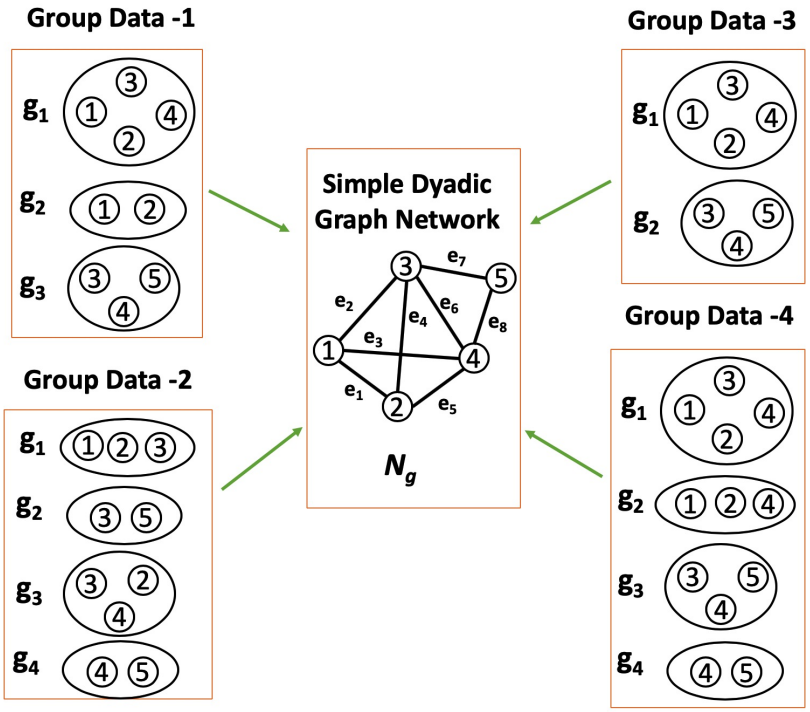

Figure 1.11: This figure gives example of a few different static group datasets that can result in the same graph after clique expansion. Thus, highlighting the fact that there is loss of information in the clique expansion process. A lot of group-level information is lost in transition to a simple dyadic graph.

or not. Any information beyond pairwise interaction is not preserved. Alternatively, for example, by observing the set of edges $(3, 4)$, $(4, 5)$ and $(3, 5)$ it isn't straightforward to tell if these three pairs were three different dyadic groups in the G or were they just a single group $(3, 4, 5)$. Therefore, there can be several different collections of static groups that could result in the same graph after clique expansion. This is demonstrated in Figure 1.11.

But can graphs do better than this? Note, in our example, group entities 1 and 2 have co-occurred twice: once as a part of the group $(1, 2, 3, 4)$ and also independently as the group $(1, 2)$. We can, therefore, think to incorporate such co-occurrence frequency information while building the graph. For this purpose, we can use weighted graphs instead of a simple graph where we can associate a weight with each edge. Weighted

graphs are defined as follows.

Definition 7 ***Weighted Dyadic Graph (or Weighted Graph):*** *For a given simple graph $N_g = (V, E_g)$, we can define an undirected weighted graph as $N_{wg} = (V, E_g, w^g)$, where w if a weight function that assigns a scalar weight w_i with each edge $e_i \in E_g$ viz. $w_i = w^g(e_i)$.*

A simple way to incorporate frequency is to assign weight $w^g(e)$ of an edge $e = (p, q)$ as the number of groups in G containing both vertices p and q. This amounts to the weight function: $w^g(e) = |\{g \in G : e \subseteq g\}|$ (see Rodriguez (2002; 2003)). By capturing this extra information about frequency, the weighted graphs turn out to be more informative than the simple graphs. Note that this extra information concerning an edge is not limited to just frequency. In general, the weight can be any function. Usually, it is some function of the weights associated with groups, like those available in the weighted group data (see definition 2). Formally, the dyadic edge weights are: $w^g(e) = f(\{w(g) : g \in G, e \subseteq g\})$ where (G, V, w) is the collection of weighted groups. Various functions f have been employed in the past to convert group data to weighted graphs. We refer readers to Agarwal et al. (2006a) and the references therein, for detailed examples of such functions.

We just accomplished two different networks by considering the "entities in group data as vertices" of simple as well as weighted graphs. We can also consider treating, not just entities, but also the **groups as vertices**. Let us again consider our example collection of static groups (G, V) where we have a set of five entities $V = \{1, 2, 3, 4, 5\}$ and a collection of three groups: $G = \{g_1, g_2, g_3\}$ where $g_1 = (1, 2, 3, 4), g_2 = (1, 2), g_3 = (3, 4, 5)$, between them (see Figure 1.12). We treat the three groups as three vertices and denote this second group-vertex set as $V_g = \{v_{g_1}, v_{g_2}, v_{g_3}\}$. We then link the entity vertices in the entity-vertex set V to the group vertices in group-vertex set V_g. We simply connect an entity vertex to a group vertex if that entity is a part of that group. This resulting network is nothing but a bipartite graph consisting of a joint vertex set $V \bigcup V_g$ and the dyadic edges $E_b = \{(1, v_{g_1}), (2, v_{g_1}), (3, v_{g_1}), (4, v_{g_1}), (1, v_{g_2}), (2, v_{g_2}), (3, v_{g_3}),$

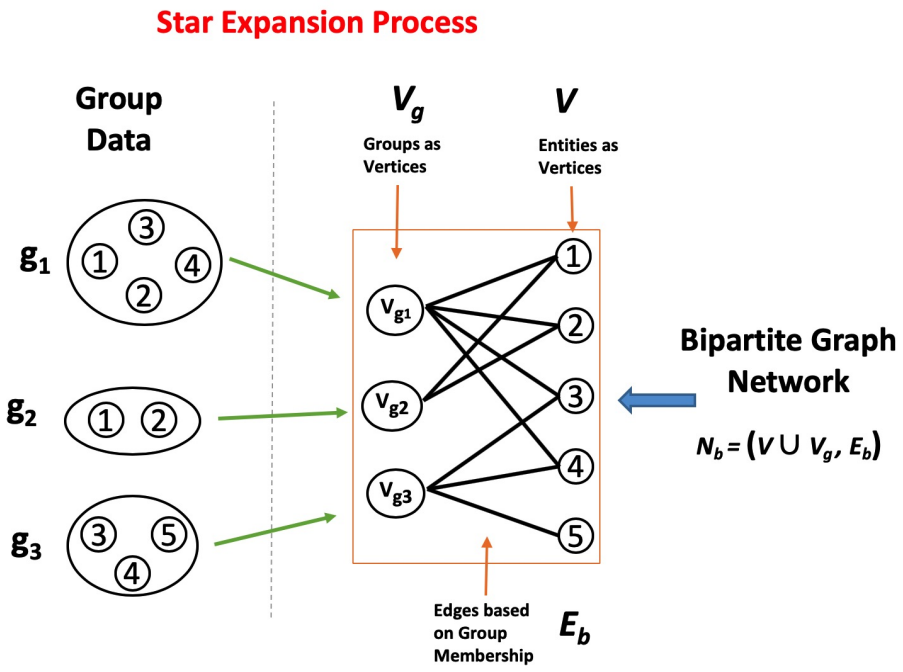

Figure 1.12: Leftmost is a collection of static groups (G, V) where we have a set of five entities $V = \{1, 2, 3, 4, 5\}$ and a collection of three groups: $G = \{g_1, g_2, g_3\}$ where $g_1 = (1, 2, 3, 4), g_2 = (1, 2), g_3 = (3, 4, 5)$. Each entity is then treated as a vertex, so V becomes the first vertex set. Also, the three groups as three vertices and we denote this second group-vertex set as $V_g = \{v_{g_1}, v_{g_2}, v_{g_3}\}$. We then link the entity vertices in the entity-vertex set V to the group vertices in group-vertex set V_g. We simply connect an entity vertex to a group vertex if that entity is a part of that group. This resulting network is nothing but a bipartite graph consisting of a joint vertex set $V \bigcup V_g$ and the dyadic edges: $E_b = \{(1, v_{g_1}), (2, v_{g_1}), (3, v_{g_1}), (4, v_{g_1}), (1, v_{g_2}), (2, v_{g_2}), (3, v_{g_3}), (4, v_{g_3}), (5, v_{g_3})\}$. The entire process of converting static group data, (G, V), to the star-expanded bipartite graph, N_b, is called the process of *star expansion*.

$(4, v_{g_3}), (5, v_{g_3})\}$. More formally we define a bipartite graph as follows:

Definition 8 ***Bipartite Graph:*** *Formally, an undirected bipartite graph can be defined as a tuple* $N_b = (V, E_b)$. *Here,* $V = (V_1 \cup V_2)$ *is the union of two vertex partitions* $V_1 \subset V$ *and* $V_2 = (V \setminus V_1)$, *and* $E_b = \{e_j = (u_j, v_j)\}_{j=1}^{m_b}$ *is a set of* m_b *dyadic edges such that* $u_j \in V_1$ *and* $v_j \in V_2$.

This concept of conversion of group data to a bipartite graph is referred as the *star expansion* (Zien et al., 1999). For purpose of clarity we formally define this expansion as follows.

Definition 9 ***Star Expansion:*** *The star expansion algorithm constructs a bipartite graph* $N_b = (V_b, E_b)$ *from a collection of static groups* (G, V). *Each entity in set* V *is treated as a vertex and therefore,* V *becomes the first vertex set. Secondly, this algorithm considers each group* $g_i \in G$ *as a vertex* v_{g_i} *as well, resulting into a second vertex set:* $V_g = \{v_{g_j} : g_j \in G\}$, *such that the complete vertex set is* $V_b = (V \cup V_g)$. E_b *is the set of* m_b *dyadic edges containing an edge from each entity vertex* $v_i \in V$ *to a group vertex* $v_{g_j} \in V_g$, *if* v_j *is an element of group* $g_j \in G$: $E_b = \{(v_i, v_{g_j}) : v_i \in V, v_{g_j} \in V_g, v_i \in g_j\}$.

Traditionally, such bipartite network models, for example, have been used under the name affiliation networks, to model relationship between actors or individuals to events/collectives (like clubs or committees) they have membership in (Wasserman & Faust, 1994). Bipartite network are also referred to as *2-mode networks*, to highlight that they have two kind of vertices as compared to simple graphs which are, in contrast, called the *1-mode networks*. We now ask ourselves the question: what does a 2-mode or bipartite graph tells us about the collection static groups from which it is derived, and how does it compare with the 1-mode simple graph? As we can observe that we have one group-type vertex dedicated to represent a group and we connect this to each entity-type vertex representing the entities within this group. This link therefore, models the relationship of a group to an entity which can have a domain dependent interpretation. For example in social groups, this link reflects the membership of a member or individual

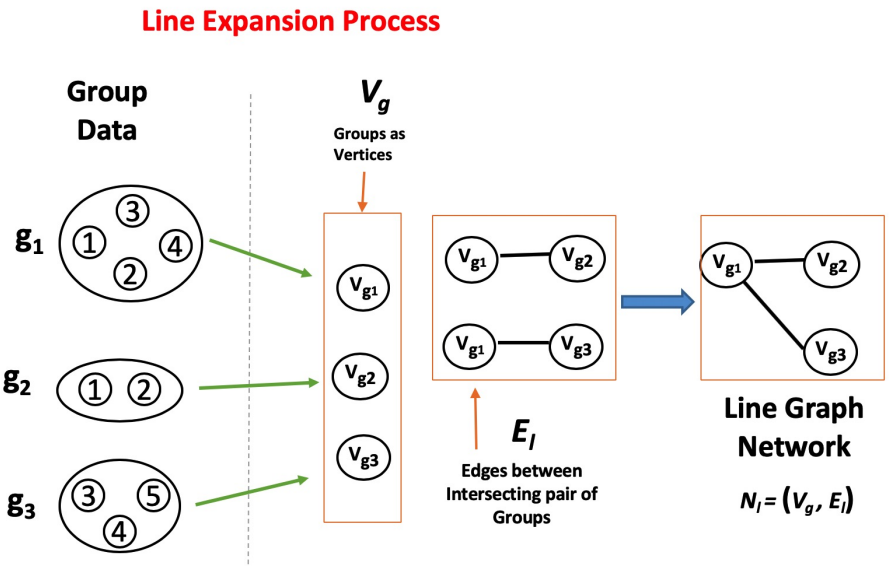

Figure 1.13: Leftmost is a collection of static groups (G, V) where we have a set of five entities $V = \{1, 2, 3, 4, 5\}$ and a collection of three groups: $G = \{g_1, g_2, g_3\}$ where $g_1 = (1, 2, 3, 4), g_2 = (1, 2), g_3 = (3, 4, 5)$. Each entity is then treated as a vertex, so V becomes the first vertex set. Also, the three groups as three vertices and we denote this second group-vertex set as $V_g = \{v_{g_1}, v_{g_2}, v_{g_3}\}$. We then link the entity vertices in the entity-vertex set V to the group vertices in group-vertex set V_g. We simply connect an entity vertex to a group vertex if that entity is a part of that group. This resulting network is nothing but a bipartite graph consisting of a joint vertex set $V \bigcup V_g$ and the dyadic edges $E_b = \{(1, v_{g_1}), (2, v_{g_1}), (3, v_{g_1}), (4, v_{g_1}), (1, v_{g_2}), (2, v_{g_2}), (3, v_{g_3}), (4, v_{g_3}), (5, v_{g_3})\}$. The entire process of converting static group data, (G, V), to the star-expanded bipartite graph, N_b, is called the process of *star expansion*.

in a particular group or community. In contrast to simple graphs which model *how two entities are related to each other across various groups*, bipartite model captures the information about *how a given entity is related to a particular group*, of which it is a member of, by treating that group as an entity. For example they can be employed to model the affinity of a member towards its team. Also, note that a simple graph edge can encode information about "multiple" groups but a bipartite edge has information pertaining to only a "single" group.

In a similar fashion we can model interaction between groups by treating each group as a vertex. Let us again consider a collection of static groups (G, V) where we have a

set of five entities $V = \{1, 2, 3, 4, 5\}$ and a collection of three groups: $G = \{g_1, g_2, g_3\}$ where $g_1 = (1, 2, 3, 4), g_2 = (1, 2), g_3 = (3, 4, 5)$, between them (shown at the leftmost in Figure 1.13). We treat the three groups as three vertices and denote this as $V_g = \{v_{g_1}, v_{g_2}, v_{g_3}\}$. We then simply construct an edge any two group vertices in group-vertex set V_g if the corresponding groups have at least on entity in common. This resulting network is nothing but an intersection (or line) graph of the family of sets defined by the collection of static groups. This network is the line graph defined by the vertex set V_g and the dyadic edges $E_l = \{(v_{g_1}, v_{g_2}), (v_{g_1}, v_{g_3})\}$. More formally we define an intersection or line graph as follows:

Definition 10 ***Intersection or Line Graph:*** *Let $\mathcal{F} = \{S_1, ..., S_n\}$ be a family of finite sets. The intersection or line graph, denoted $L(\mathcal{F})$, is the graph having $\mathcal{F}$ as the vertex set with S_i adjacent to S_j if and only if $i \neq j$ and $S_i \cap S_j \neq \phi$. Here, $L(\cdot)$ is the line graph operator. (McKee & McMorris, 1999)*

We name this process of conversion from group data to line or intersection graph as *line expansion*; which we formally define as follows:

Definition 11 ***Line Expansion:*** *The line expansion algorithm constructs a line graph $N_l = (V_g, E_l)$ from a collection of static groups (G, V). This algorithm considers each group $g_i \in G$ as a vertex v_{g_i} resulting into a vertex set: $V_g = \{v_{g_j} : g_j \in G\}$. E_l is the set of m_l dyadic edges containing an edge between any two group vertex v_{g_i} and v_{g_j}, if the groups are intersecting i.e. $g_i \cap g_j \neq \phi$.*

The line graph is also referred to as the *representative graph*, and its role in capturing relationships among groups seems pretty intuitive. Unlike the bipartite graph, which is useful in modeling the relationship between an entity and a group, an edge in line-graphs can be used to model the interaction between two groups corresponding to the group vertices of the edge. For example, in social networks, line graphs can be employed to model the affinity between two social groups, which can be used to, for example, predict the likelihood of two groups merging in the future. Also notice that in contrast

to bipartite networks which contain two kinds of vertices, the line-graph is a dyadic simple graph.

However, neither the *2-mode* or bipartite network nor the line graph, explicitly capture relations between the vertices themselves, which have to be derived. On the other hand the derived *1-mode* or dyadic graph of entities, as argued before, cannot capture the coexistence of more than two entities in single relation. Therefore, in a nutshell all the graph based models which we have describe above have some limitations while trying to capture the group interactions. This demands for a mathematical structure that can capture node-to-node, node-to-group, and group-to-group interactions simultaneously.

We turn our attention to the mathematical structure called Hypergraphs, which can easily capture higher-order relationships while incorporating both group and node-level relations. In a hypergraph, in addition to having edges between pairs of entities, one can define a hyperedge that links entities that are part of a single group. Figure 1 shows a hypergraph with five nodes and three hyperedges. For example, three nodes 3, 4, and 5 are linked by a single hyperedge. Let us define hypergraphs more formally.

Definition 12 ***Hypergraph:*** *Formally, a hypergraph can be defined as a tuple* $H = (V, E)$*, where* $V = \{v_i\}_{i=1}^{n}$ *is the set of vertices* (v_i) *in the network and* $E = \{e_j\}_{j=1}^{m}$ *is the set of hyperedges:* $e_j \subseteq V$*. Each* e_j *is called a hyperedge, and represents a relation between one or more entities (Berge, 1976).*

Note: *We shall also use* N_g *interchangeably with* H*, specially in cases where we are referring to hypergraph as network of groups.*

Hypergraphs, therefore, are a generalization of graphs as well as bipartite graphs (Berge, 1984), which can have more than two nodes in an edge (rather than simple graphs where only 2 nodes are part of an edge). Therefore, hypergraphs can easily capture the coexistence of more than two entities in single relation. With this observation, we end this subsection and, in the next subsection, detail out the road map of modeling hypergraph, which is the focus of this thesis.

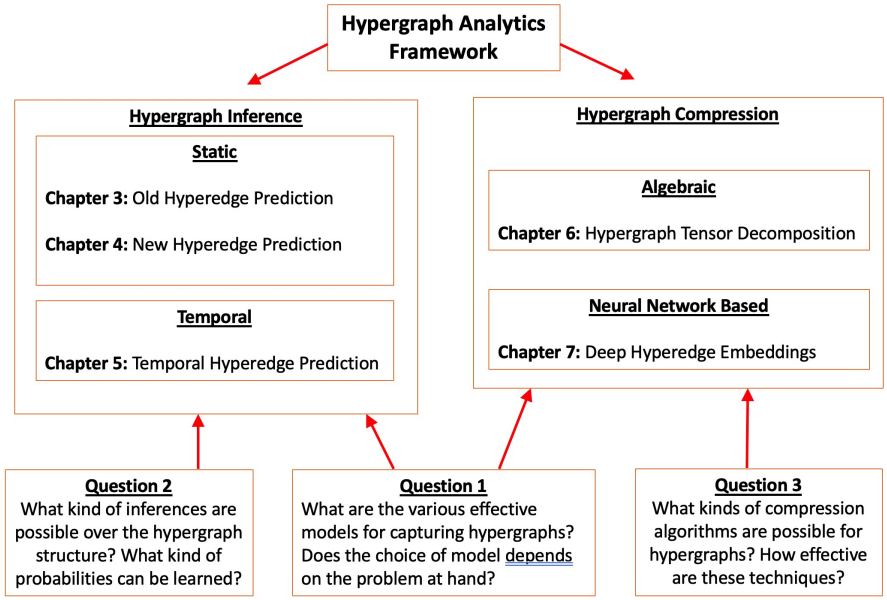

Figure 1.14: Thesis diagram describing the Hypergraph Analytics Framework and its various components; indicating which chapters map to which component of the framework.

1.3 Hypergraph Analytics

In the previous section, we observed that modeling group phenomena entail capturing the higher-order information encompassed in a group interaction between multiple entities. We then detailed the various possible graph-based networks that we can employ to model group data – starting from simple graphs to other kinds of graphs. We ended up with the realization that we have to go beyond simple graphs, which are limited to pair-wise relations, to capture group phenomena that require modeling higher-order relations. The **higher-order structure** that we are most interested in modeling is the **hypergraph**, which seems to be naturally suited for modeling group interactions. ***This thesis is dedicated to model and study group phenomena from the lens of Hypergraphs. In order to do so, we propose the Hypergraph Analytics Framework*** (see Figure 1.14).

Under this framework, we shall be primarily interested in the following higher-level

questions pertaining to the hypergraph modeling.

- *What are the various effective models for capturing hypergraphs? Does the choice of model depends on the problem at hand?*
- *What kind of inferences are possible over the hypergraph structure? What kind of probabilities can be learned?*
- *What kinds of compression algorithms are possible for hypergraphs? How effective are these techniques?*

Although there are several models, but the effective models for capturing hypergraphs are still not precise. Different models primarily vary by the amount of higher-order information they can retain. We split these models into four categories: bipartite graphs, hasse diagrams, clique expanded graphs, and hypergraphs in their original form (without any transformation to the previous three categories). A detailed overview of these categories, along with their data structures, shall be provided in Chapter 2. Various problems studied in this thesis shall employ all of these models. Choice of model will be dictated by the problem at hand. For example, in several cases a lower-order model approximation would be suffice and in other cases precise higher-order information retaining model is recommended. In this manner, across all chapters, we address the first central question of this thesis: how to model higher-order hypergraph information and what kind of lower-order approximations are available or sufficient depending upon the problem at hand.

As listed before, there are two other central questions about hypergraphs, which this thesis aims to find answers for. First, is that of conducting inference or learning probabilities over hypergraph structures. Second is that of compressing these higher-order hypergraph structures to low dimension embedding spaces. We, therefore, have two parts of this thesis. First part consists of Chapters 3, 4 and 5, which are dedicated to various kind of **inference mechanisms on hypergraph** structures. Second part

consists of Chapters 6 and 7, which are concerned with the **hypergraph compression techniques**. We now provide a short overview of these chapters below.

First and foremost, we would like to bring attention to the fact that by *hypergraph structure* we are referring to the entire set of possible hyperedges between a given set of entities. This set encompasses both the observed portion as well as that is not yet observed portion. Given a past observed data, one can can infer likelihood of both: the already observed group interactions to again happen in future as well as the probabilities of the unobserved portion of the hypergraph structure. Ascertaining the probabilities of observed and unobserved regions of a hypergraph structure, given the past observations, is what we refer to as the *inference on hypergraph structure*.

General inference over hypergraph structure is a complex problem. In this thesis we dissect it into smaller sub-problems which we have tried to answer separately. We perform the division on two dimensions. First, if the inference is being performed with or without taking the time information within group interactions into account. In other words if the inference is performed on temporal groups or static groups. Secondly, we distinguish on the basis of whether the inference deals with learning probabilities for observed regions of the hypergraph structure or that of the unobserved regions.

Let us give a more systematic treatment to these divisions, starting by providing some definitions.

Definition 13 ***Hypergraph Structure:*** *Given a set of n vertices $V = \{v_i\}_{i=1}^{n}$, the hypergraph structure is the tuple (V, S) with set $S = \{s | s \subseteq V \wedge s \neq \phi\}$, which is nothing but the set of all the not empty subsets of the vertex set V. Notice that $|S| = 2^n - 1$.*

Definition 14 ***Observed Hypergraph Structure:*** *Given a collection of static groups (G, V) we treat each entity in set V as a vertex and therefore, V becomes the vertex set. Let (V, S) be the hypergraph structure associated with this vertex set. We denote by $S_i = \{s_k^i, \forall k \in \{1, 2, ..., 2^{|g_i|} - 2\}\}$ the set of all proper subsets of each group $g_i \in G$. If we consider the union of all subsets of the sets in G along with G itself, i.e., $E^o = \{G \cup (\bigcup_{i=1}^{m} S_i)\}$, then we have the observed hypergraph structure, (V, E^o).*

Definition 15 ***Unobserved Hypergraph Structure:*** *Given a collection of static groups* (G, V) *and both the hypergraph* $((V, S))$ *as well as the observed hypergraph* $((V, E^o))$ *structures associated with it, we define the unobserved hypergraph structure simply as the* $E^u = \{S - E^o\}$.

Therefore, the hypergraph structure is a union of the two mutually exclusive sets: the observed and unobserved hypergraph structure; i.e. $S = E^o \cup E^u$ and $(E^o \cap E^u) \neq \phi$. To give a complete picture we also define specifically the *observed hypergraph*.

Definition 16 ***Observed Hypergraph:*** *Given a collection of static groups* (G, V) *we treat each entity in set* V *as a vertex and therefore,* V *becomes the vertex set. For each set* $g_i \in G$ *we associate a hyperedge* e_i*:* $E = \{e_i | e_i = g_i, \forall g_i \in G\}$. *This results in the hypergraph* $H = (V, E)$*, which we refer to as the observed hypergraph.*

Notice that *observed hypergraph* (H) is different from the *hypergraph structures*, in fact, $E \subseteq E^o$. It the most common hypergraph object associated with a group structured data and often in literature, it is the observed hypergraph that is being modeled. For simplicity we will use the word *hypergraph* to refer to this *observed hypergraph*.

Given this setup i.e. hypergraph and the various hypergraph structures, we divide the problem of inference on hypergraph structure into sub-problems. As there are two different kind of hypergraph structures: observed and the unobserved, its natural to have two different sub-problems. First is that of predicting likelihood of hyperedges in the observed hypergraph structures – ***old hyperedge prediction*** – given the observed hypergraph. Second is that of assigning probabilities to hyperedges in the unobserved hypergraph structure – ***new hyperedge prediction*** – again given the observed hypergraph. Chapter 3 and 4 respectively answer these two questions (see Figure 1.14).

Chapter 3 answers the first question. In this chapter we take the problem of predicting future groups or subgroups given past group interactions. The observed hypergraph structures is modelled using a *simplicial complex* and corresponding *hasse lattice* structure. A general overview of these models shall be provided in Chapter 2 and details

in the Chapter 3 itself. Model is evaluated for predicting both previously observed hyperedges as well as unobserved hyperedges. The evaluation is conducted for various cardinalities of the hyperedges.

Chapter 4 addressed the second problem of modeling the probabilities of hyperedges within the unobserved hypergraph structure. The biggest impediment in giving probabilities to the hyperedges in the unobserved hypergraph structure is the size of the later. Most of the real-world hypergraphs are sparse i.e. $m = |E| = O(n)$ and the maximum cardinality of observed hyperedges are well bounded i.e. $c_{max} << n$. Both these observations also amount to a sparse observed hypergraph structure i.e. $m' = |E^o| = O(n)$. However, the size of unobserved hypergraph structure explodes exponentially with increase in n. To contain this number we require a sampling mechanism that samples meaningful possible future hyperedges and assigns probability to them. In this chapter we propose a simple and elegant approach of incremental sampling. Here we sample a new hyperedge by adding a single vertex to a past observed hyperedge or its sub-hyperedge. Surprisingly, this sampling statistically seems to be covering a significant portion of future hyperedges in the datasets we study. We then propose various graph as well as hypergraph methods which output probability of the sampled hyperedges. We refer to these algorithms as accretion prediction methods, as they deal with an increase or accretion of hyperedges by addition of vertices. Given the elementary nature of the single vertex addition scheme, one can apply it recursively to ascertain probabilities of even farther (in cardinality sense) hyperedges in the unobserved region. Further, the algorithm proposed can also be easily generalized to non-incremental addition schemes i.e. adding more than one vertex at a time.

In summary, Chapter 3 deals with probabilities of past hyperedges repeating or hyperedge attrition prediction in form of sub-hyperedge prediction. Chapter 4 on the other hand address the hyperedge accretion prediction. Overall these two chapters address the hypergraph evolution problem by addressing sub-problems of hyperedge stability, as well as increase or decrease in cardinality prediction. However, we should notice that both these chapters are limited to static group data only. Therefore, they

are performing hypergraph evolution prediction by cross-sectional or static analysis(see Figure 1.14).

Chapter 5, therefore, extends our study of hypergraph evolution to temporal group data and performing longitudinal analysis. Time is a crucial information in modeling evolution of various interactions. Unlike static groups where the focus is limited to modeling the group structure, modeling temporal groups would require the model to also capture the time of group interaction apart from the group structure. In this chapter we model the time as another dimension of a higher-order matrices, also called *tensors*. Dimensions other than the time dimension are used to capture the group structure. This temporal tensor model can then be used for inference over hypergraph structure. This chapter limits itself to the problem of group or hyperedge repetition prediction. The inference is performed using algebraic techniques of tensor decomposition. A key focus of this chapter is to evaluate the performance of hypergraph bases tensors to that of graph based tensors and thus, performing graph versus hypergraph comparison, for the task of ***temporal hyperedge prediction***. The tensor decomposition techniques proposed have the ability to perform not just next time step predictions, but also longer range forecasting. It is easy to generalise these methods for the case of other hypergraph structure inferences for both observed and unobserved regions.

Interlacing the Chapters 3, 4 and 5, we reach a complete recipe of modeling and inference over the hypergraph structure for both cross-sectional and longitudinal analysis. These chapters together constitute the first part of the thesis. We would like to take the freedom of calling this part the ***Spatial Analysis*** within our Hypergraph Analytics framework. Here the "space" constitutes the hypergraph structure and the time dimension.

Second part of thesis, while complimenting the first half, deals with the ***Spectral Analysis*** part of the Hypergraph Analytics framework. In this we deal with techniques to compress the hypergraph topology to lower dimensional latent space. Separately, each dimension of this latent space provide a notion of frequency and together these dimensions provide the entire spectrum. Considering hyperedges as first class citizen,

we shall be chiefly considering hyperedge compression or *hyperedge embeddings*. We will examine two different embedding approaches in Chapters 6 and 7, which together will comprise the latter half of the thesis (see Figure 1.14).

Chapter 6 develops an algebraic model for hyperedge emebeddings for general (non-uniform) hypergraphs. We leverage the relationship between uniform hypergraphs and symmetric higher-order tensors. Given our main focus is embedding hyperedges, then it becomes crucial to retain the higher-order information contained in them. Making use of higher-order tensors, therefore, helps us retain the hyperedge-level information directly. However, uniform hypergraph tensors only consider hyperedges of fixed cardinality. We generalise this to non-uniform hypergraphs by considering array of tensors of various cardinalities. We then use tensor decomposition techniques which decompose tensors to latent factor matrices. These latent factors are the embeddings. As in our case we have an array of tensors, therefore, we propose a joint decomposition of these tensors along with hypergraph topology based regularization. ***Hypergraph tensor decomposition***, although preserves the hyperedge-level information, but, only outputs vertex embeddings. To address this issue we propose the notion of dual higher-order tensors which work on the hypergraph dual. The proposed joint tensor decomposition on array of dual tensors then outputs hyperedge embeddings directly. These embeddings are evaluates using various attribute based hypergraphs as well as simulated hypergraph data. Hyperedge embeddings are observed to be performing better than hyperedge embeddings built by combining vertex embeddings achieved from various graph based baselines.

Chapter 7 revisits the problem of hyperedge embedding, but unlike the batch-learning based model in the previous chapter, here we consider an online setting. Neural network-based models are inherently online, and in the past few years, they have shown tremendous performance on a wide variety of tasks. Hence, we choose neural networks for our task of learning ***deep hyperedge embeddings***. In general, the literature on graph embeddings is significantly more than on hypergraph embeddings. Also, until recently, the models for embedding graph structures have been focused more on techniques other than neural networks. Here we attempt to permeate this gap by de-

veloping a neural network supported hypergraph embedding methods. Inspired by the image auto-encoders from computer vision, we consider the hyperedge as the primary entity to be encoded and develop hyperedge auto-encoders. But unlike the images, where we use Gaussian noise per pixel to generate noisy image samples, hyperedges are discrete structures and require a discrete noise. Random discrete noise like salt-pepper noise might result in entirely unrelated noisy hyperedge samples, which may result in a non-meaningful training. Instead, we harness the hypergraph structure and use it to sample noisy hyperedges for a given original hyperedge. We devise a variety of random walk schemes on the hypergraph structure and generating meaningful noisy hyperedges. During the evaluation of these embeddings, our focus is three folds. We aim to examine how the embeddings that use graph topology perform in comparison to those leveraging the hypergraph structure. Secondly, we aim to study how shallow architectures perform in the relation of deep auto-encoders. Lastly, we intend to contrast the different noise generation schemes that we have designed. Hence, we design several experiments to carry out these evaluations using both real as well as synthetic datasets.

Chapter 2

Modeling Applications as Hypergraphs

In the previous chapter we introduced the basic problems this thesis studies and provided an overview of the over all analysis framework. We also enumerated the various classes of group structured datasets and defined a set of group data abstractions. Further, we also introduced some graph models for group data and provided some idea of the conversion process from group data to graph models. But this introduction was not mathematically rigorous and lacked use of any mathematical data structures. Therefore, as promised in Section 1.3, in this chapter we introduce the various models for hypergraphs as well as the supporting algebraic structures. The variety of models described in here have variable higher-order information retention capacity, but our discussion in this chapter would be limited to defining these models algebraically. All these models will be capitalized in the following chapters of this thesis.

Section 2.1 introduces the concept of hypergraph data and establishes some basic assumptions as well as hypergraph preliminaries which are followed in rest of this chapter. In Section 2.2 we introduce the various graph-based "proxy" models for hypergraph data. In the following Section 2.3, we introduce the hypergraph models for hypergraph data. In both Section 2.2 and 2.3 we provide definition as well as construction of var-

ious algebraic data structures associated with these models. Next, in Section 2.4 we describe temporal hypergraphs, their construction from temporal group data as well the associated algebraic infrastructure. Lastly, Section 2.5 describes the large variety of real world as well as synthetic hypergraph datasets that has been curated and employed during the research work within this thesis.

2.1 Preliminaries

2.1.1 Notations

We use non-bold face lower letters – $x, y, m, n, ...$ – to denote scalars. Lower case bold face letters – $\mathbf{u}, \mathbf{v}, \mathbf{w}, ...$ – to denote vectors; upper case bold face letters – $\mathbf{A}, \mathbf{H}, \mathbf{U}, ...$ – to denote matrices; upper case bold face *calligraphic* letters – $\mathcal{A}, \mathcal{Z}, \mathcal{P}, ...$ – to denote tensors. $\mathbb{R}$ and $\mathbb{R}^+$ denotes real numbers and positive real numbers respectively. $\mathbb{Z}$ and $\mathbb{Z}^+$ denote integers and positive integers, respectively. $[n]$ is the shorthand notation for the list of integers $(1, 2, ..., n)$.

2.1.2 Group Data as Hypergraph

The main focus of this thesis is to study *group structured data*. In Section 1.2 we defined various categories of group data that shall be of interest for us and then defined four different abstractions. We defined *Static Groups* and *Weighted Groups* abstractions (see Definitions 1 and 2), which do not take into consideration the temporal information. On the other hand the abstraction of *Temporal Groups* (see Definition 3) and *Weighted Temporal Groups*, which allows for modeling time. In this and the Sections 2.2 and 2.3, we shall be working with weighted groups. Section 2.4 will focus on the weighted temporal groups separately.

As a stepping stone for analyzing group structured data, we treat the abstraction of group data as a hypergraph. First we deal with static groups and more specifically we treat a weighted groups as a weighted hypergraph. Note that static groups are a

special case of weighted groups and therefore, models for weighted groups automatically models static groups as well. We had earlier defined an unweighted hypergraph (see Definition 12), let us now define a weighted hypergraph.

Definition 17 ***Weighted Hypergraph:*** *Formally, a weighted hypergraph can be defined as a tuple $H = (V, E, w)$, where $V = \{v_i\}_{i=1}^{n}$ is the set of vertices (v_i) in the network and $E = \{e_j\}_{j=1}^{m}$ is the set of hyperedges: $e_j \subseteq V$. Each e_j is called a hyperedge, and represents a relation between one or more entities. We also associate with each hyperedge e_i, a scalar weight w_i, which can be considered as a property of the relation which that hyperedge represents.*

We name this process of conversion from weighted groups to weighted hypergraph as *weighted hypergraph conversion* or simply hypergraph conversion; which we formally define as follows:

Definition 18 ***Weighted Hypergraph Conversion:*** *The weighted hypergraph conversion algorithm constructs a hypergraph H from a collection of weighted groups (G, V, w). This algorithm considers each entity v_i as a vertex, resulting into a vertex set V. Then we consider each group $g_i \in G$ of weight w_i as a weighted hyperedge, resulting into a set of hyperedges G and w as the weight function for these hyperedges. Together the tuple (V, G, w) constitute the weighted hypergraph H i.e. $H = (V, G, w)$ where $w_i = w(g_i)$.*

Note: *We shall also use N_g interchangeably with H, specially in cases where we are referring to hypergraph as network of groups.*

Once we have performed the conversion of group data to a hypergraph, we shall no longer be addressing group data in any further discussions. We will be concerned only to this hypergraph obtained from the conversion.

2.1.3 Hypergraph Preliminaries

In the interest of a more rigorous mathematical treatment while developing the various models, we shall be gradually introducing a variety of algebraic structures. The first and

the most basic one is the hypergraph incidence matrix, $\mathbf{H}$; which is defined as follows:

Definition 19 ***Hypergraph Incidence Matrix:*** *Given an unweighted hypergraph $H = (V, G)$ or a weighted hypergraph $H = (V, G, w)$, we can associate to it an incidence matrix $\mathbf{H} \in \{0, 1\}^{|G| \times |V|}$, where $m = |G|$ is the number of hyperedges and $n = |V|$ is the number of vertices of the hypergraph. The (g_i, v_j)-th entry of this matrix is 1 if j-th vertex is a part of i-th hyperedge i.e. $\mathbf{H}(g_i, v_j) = 1$ if $v_j \in g_i$ else 0. Each row of the incidence matrix, therefore, represents a particular hyperedge and its membership.*

Definition 20 ***Hypergraph Weight Matrix:*** *Given a weighted hypergraph $H = (V, G, w)$, we can associate to it a diagonal matrix, $\mathbf{W}$, of hyperedge weights, such that:*

$$\mathbf{W}(g, g) = w(g) \ , \ \forall g \in G \ , \tag{2.1}$$

which we refer to as the hypergraph weight matrix.

To provide a complete picture, we would also describe the *Hypergraph Dual* and its incidence matrix. The hypergraph dual is dual of a hypergraph in the sense that if we reverse the role of vertex and hyperedges in a hypergraph, we achieve its hypergraph dual. Every hypergraph has a hypergraph dual associated with it. Below we provide definitions for the both the hypergraph dual as well as the incidence matrix associated with it.

Definition 21 ***Hypergraph Dual:*** *The dual of a hypergraph $H = (V, G)$ is a hypergraph $H^* = (V^*, G^*)$ whose vertices $V^* = \{v_1^*, ..., v_m^*\}$ correspond to the hyperedges of H i.e. $v_i^* = g_i$ and with hyperedges $G^* = \{g_1^*, ..., g_n^*\}$ such that $g_i^* = \{g_j | v_i \in g_j, g_j \in G\}$ (Berge, 1984).*

Definition 22 ***Weighted Hypergraph Dual:*** *Let $H^* = (V^*, G^*)$ be the dual of hypergraph $H = (V, G)$. Then the dual of weighted hypergraph $H = (V, G, w)$ is $H^* = (V^*, G^*, w^*)$, where w^* is a weight function that assigns a scalar weight w_i^* to*

each hyperedge $g_i^* \in G*$. This weight function w_i^* can be a function of the weights of the dual vertices which are hyperedges in the original hypergraph viz. $w_i^* = w^*(s)$ where $s = \{w(g_{q_1}), ..., w(g_{q_k})\}$ and $v_i \in \{g_{q_1}, g_{q_2}, ..., g_{q_k}\}$.

Definition 23 ***Hypergraph Dual Incidence Matrix:*** *Let $H^* = (V^*, G^*)$ be the dual of the unweighted hypergraph $H = (V, G)$ or a weighted hypergraph $H = (V, G, w)$. We can associate to it an incidence matrix $\mathbf{H_{dual}} \in \{0,1\}^{|V|\times|G|}$ such that $\mathbf{H_{dual}} = \mathbf{H}^\intercal$ (Berge, 1984). Each row of the incidence matrix, therefore, represents a particular vertex and the hyperedges incident on it.*

Degrees of a Hypergraph

We can associate different kinds of degrees to a hypergraph. There are three basic degree: vertex degree, hyperedge degree and hyperedge cardinality; which we define as follows.

Definition 24 ***Vertex Degree:*** *Given an unweighted hypergraph $H = (V, G)$, we define degree $d(v)$ of a vertex v as the number of hyperedges incident on this vertex:*

$$d(v) = \sum_{g_i \in G} \mathbf{H}(g_i, v) \tag{2.2}$$

$\mathbf{D_v}$ *is the diagonal matrix consisting of vertex degrees viz. $\mathbf{D_v}(i,i) = d(v_i), \forall i \in [n]$.*

Definition 25 ***Weighted Vertex Degree:*** *Given an weighted hypergraph $H = (V, G, w)$, we define degree $d(v)$ of a vertex v as sum of the weights of the hyperedges incident on this vertex:*

$$d(v) = \sum_{g_i \in G} (\mathbf{H}(g_i, v) \cdot w(g_i)) \tag{2.3}$$

$\mathbf{D_v}$ *is the diagonal matrix consisting of vertex degrees viz. $\mathbf{D_v}(i,i) = d(v_i), \forall i \in [n]$.*

Note that we are using the same notations $d(\cdot)$ and $\mathbf{D_v}$ for both weighted as well as unweighted vertex degree; depending on the context it will be clear.

Definition 26 ***Hyperedge Hyperdegree:*** *Assume we are given an unweighted hypergraph $H = (V, G)$. Let us define the set G_g associated with a hyperedge g such that:*

$$G_g = \{e | e \cap g \neq \emptyset, e \subset G, e \neq g\} \tag{2.4}$$

set of all the incident hyperedges. Hyperedge degree $d(g)$ of a hyperedge g is then defined as the cardinality of this set:

$$\psi(g) = |G_g| \tag{2.5}$$

$\mathbf{\Psi_e}$ *is the diagonal matrix consisting of vertex degrees viz.* $\mathbf{\Psi_e}(j, j) = d(e_j), \forall j \in [m]$.

Definition 27 ***Weighted Hyperedge Hyperdegree:*** *Assume we are given an weighted hypergraph $H = (V, G, w)$. Let us define the set G_g associated with a hyperedge g such that:*

$$G_g = \{e | e \cap g \neq \emptyset, e \subset G, e \neq g\} \tag{2.6}$$

set of all the incident hyperedges. Hyperedge degree $d(g)$ of a hyperedge g is then defined as the sum of the weights of the hyperedges in the set G_g:

$$\psi(g) = \sum_{e \in G_g} w(e) \tag{2.7}$$

$\mathbf{\Psi_e}$ *is the diagonal matrix consisting of vertex degrees viz.* $\mathbf{\Psi_e}(j, j) = d(e_j), \forall j \in [m]$.

Note that we are using the same notations $\psi(\cdot)$ and $\mathbf{\Psi_e}$ for both weighted as well as unweighted hyperdegree degree; depending on the context it will be clear. Also to be succinct we shall be using "hyperdegree" to refer to hyperedge hyperdegree.

Definition 28 ***Hyperedge Degree or Cardinality:*** *Given an unweighted hypergraph $H = (V, G)$, we define the cardinality $c(g)$ of a hyperedge g as the count of the number of vertices inside the hyperedge:*

$$c(g) = |g| = \sum_{v \in V} \mathbf{H}(g, v) \tag{2.8}$$

$\mathbf{D_e}$ *is the diagonal matrix consisting of vertex degrees viz.* $\mathbf{D_e}(j,j) = c(e_j), \forall j \in [m]$.

2.2 Graph based Hypergraph Models

The first category of models we would consider are the graph-based models. Note that we had already introduced these models in Section 1.2, where we considered the conversion of group data into these graph models. However, there the treatment was less rigorous and more focused on discussing the degree of higher-order information retention across these variety of graph based models. Here we address it by providing a proper algebraic treatment. As agreed in Section 2.1.2, for all the following discussion, we shall assume that the original group data is already translated to a hypergraph. Also all the models considered in this section are static and not incorporate any temporal information.

2.2.1 Clique Expanded Graph

We first consider the graph model based on the process of *clique expansion* (see Def. 6) which we had earlier described in Section 1.2. This process converts a group data into simple dyadic graph (see Def. 4) by connecting each pair of elements for each of the groups and then aggregating them. We revisit this process and analyse it in a more structured manner and generalise it for both weighted and unweighted hypergraphs. We therefore, redefine the clique expansion process by defining the adjacency matrix of associated with the output simple graph.

Definition 29 ***Clique Expanded Graph (CE Graph):*** *Consider an unweighted hypergraph* $H = (V, G)$ *or a weighted hypergraph* $H = (V, G, w)$. *Let us define a matrix* $\mathbf{J} = \mathbf{H}^\intercal\mathbf{H} - \mathbf{D_v}$. *Then we define the adjacency matrix of the clique expanded simple graph,* $\mathbf{A_c} \in \{0,1\}^{|V|\times|V|}$, *as:*

$$\mathbf{A_c}(v_i, v_j) = \begin{cases} 1 & \text{if } \mathbf{J}(v_i, v_j) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

This is the adjacency matrix associated with the clique expanded simple graph $N_c = (V, E_c)$ *(see Def. 6).*

This vanilla clique expanded graph very well defines the basic clique-expanded topology but it discreetly disregards the an information pertaining to hyperedge weights. In order to incorporate this we turn our attention back to weighted graphs which we had defined in last chapter (see Def. 7) corresponding to simple graphs obtained via clique expansion. There we had briefly also described some weighting schemes. Here we reexamine those as well as more schemes to assign weights to the CE Graph. Interestingly, several studies on hypergraphs are essentially work using the same underlying CE graph and only differ in how they assign weights to its edges. So these methods, which we describe below, claim to study hypergraphs but actually they work on a graph based approximation or proxy.

Each method provides a scheme to give weights to edges of the CE graph and associate an adjacency matrix to the resulting Weighted CE graph (WCE graph). Note, unlike the CE graph these adjacency matrices are weighted as the graph is weighted. For the purpose of clarity we again define a weighted graph from clique expansion.

Definition 30 ***Weighted Clique Expanded Graph (WCE Graph):*** *For a given CE graph* $N_c = (V, E_c)$*, we can define an undirected weighted CE graph as* $N_{wc} = (V, E_c, w^c)$*, where* w^c *is a weight function that assigns a scalar weight* w^c_{ij} *to each edge* $(v_i, v_j) \in E_c$ *viz.* $w^c_{ij} = w^c(v_i, v_j)$*. The adjacency matrix of this graph is,* $\mathbf{A_{wc}} \in \mathbb{R}^{|V|\times|V|}$*, as:*

$$\mathbf{A_{wc}}(v_i, v_j) = w^c_{ij} \ . \tag{2.10}$$

Further, we will also associate a graph Laplacian matrix for a given CE or WCE graph. The graph Laplacian is the discrete analog of the Laplace-Beltrami operator on compact Riemannian manifolds (Belkin & Niyogi, 2001; Chung et al., 1986; Rosenberg, 1997). We will be making use these Laplacians in various machine learning models in the following chapters. There are different kinds of graph Lapacians (Chung & Graham,

1997) but we shall restrict ourselves to normalized Laplacian matrix. It is defined based on the graph adjacency matrix as follows:

$$\mathbf{L_{wc}} = (\mathbf{I} - \mathbf{D_v^c}^{-1/2}\mathbf{A_{wc}}\mathbf{D_v^c}^{-1/2}) \ , \tag{2.11}$$

where $\mathbf{D_v^c}$ is the diagonal matrix containing the vertex degrees which are defined as:

$$d^c(v_i) = \sum_{v_j \in V} \mathbf{A_{wc}}(v_i, v_j) \ , \tag{2.12}$$

which is simply the i-th row sum of the adjacency matrix $\mathbf{A_{wc}}$. Equations 2.10 and 2.15 define the template for the weighted CE graph but does not provided any definition for the edge weights. Now we will describe various kind of WCE graphs which vary by the different intuitive manners they assign weights to CE edges. For each graph we will provide the adjacency matrix and also the associated normalized Laplacian. Notice that a WCE graph builds upon a CE graph which can be obtained from either a weighted or an unweighted original hypergraph. So in a nutshell we can still achieve a weighted CE graph irrespective of whether the orginal hypergraph was weighted or not.

Definition 31 ***Rodríguez WCE graph (Rodriguez, 2002; 2003):*** *Consider a weighted CE graph , $N_r = (V, E_r, w^r)$ for given unweighted hypergraph $H = (V, G)$. A simple scheme would be to assign weight w_{ij}^c as the number of hyperedges common between the vertices v_i and v_j viz.:*

$$w_{ij}^c = \sum_{g \in G} \mathbf{H}(g, v_i)\mathbf{H}(g, v_j) \ , \tag{2.13}$$

which amounts to the following adjacency matrix:

$$\mathbf{A_r} = \mathbf{H^\intercal I H} = \mathbf{H^\intercal H} \ , \tag{2.14}$$

where $\mathbf{I}$ *highlights that we take each hyperedge's weight as one viz. unweighted hypergraph. The normalized Rodríguez Laplacian is:*

$$\mathbf{L_r} = (\mathbf{I} - \mathbf{D_v^r}^{-1/2}\mathbf{A_r}\mathbf{D_v^r}^{-1/2}) \ , \tag{2.15}$$

where $\mathbf{D_v^r}$ *is the diagonal matrix containing the vertex degrees which are defined (analogous to Eq. 2.12) as:*

$$\begin{aligned} d^r(v_i) &= \sum_{v_j \in V} \mathbf{A_r}(v_i, v_j) \\ &= \sum_{v_j \in V} \sum_{g \in G} \mathbf{H}(g, v_i)\mathbf{H}(g, v_j) \\ &= \sum_{g \in G} \mathbf{H}(g, v_i) \sum_{v_j \in V} \mathbf{H}(g, v_j) = \sum_{g \in G} \mathbf{H}(g, v_i)\mathbf{D_e}(g, g) \ , \end{aligned}$$

which amounts to:

$$\mathbf{D_v^r} = \mathbf{H}^\intercal \mathbf{D_e} \ . \tag{2.16}$$

Then, N_r *is the Rodríguez WCE graph with the adjacency matrix* $\mathbf{A_r}$ *and the normalized Lapacian* $\mathbf{L_r}$*.*

Another intuitive scheme could be to assign WCE edge weights with fractional contribution from each common hyperedge being inversely proportional to the hyperedge weight. This would result to what is referred in literature as Bolla WCE graph, which is described next.

Definition 32 ***Bolla WCE graph (Bolla, 1993):*** *Consider a weighted CE graph ,* $N_o = (V, E_o, w^o)$ *for given unweighted hypergraph* $H = (V, G)$*. A simple scheme would be to assign weight* w_{ij}^c *as the sum of the weights of the hyperedges common between the*

vertices v_i *and* v_j *viz.:*

$$w_{ij}^{c} = \sum_{g \in G} \mathbf{H}(g, v_i) \cdot (1/c(g)) \cdot \mathbf{H}(g, v_j) \ , \tag{2.17}$$

which amounts to the following adjacency matrix:

$$\mathbf{A_o} = \mathbf{H^{\intercal}} \mathbf{D_e}^{-1} \mathbf{H} \ , \tag{2.18}$$

where $\mathbf{D_e}^{-1}$ *highlights that we may assume each hyperedge's weight as inverse of its cardinality even though the hypergraph we are considering is unweighted hypergraph. The normalized Bolla's Laplacian is:*

$$\begin{aligned} \mathbf{L_r} &= (\mathbf{I} - \mathbf{D_v^o}^{-1/2} \mathbf{A_o} \mathbf{D_v^o}^{-1/2}) \\ &= (\mathbf{I} - \mathbf{D_v}^{-1/2} \mathbf{A_o} \mathbf{D_v}^{-1/2}) \ , \end{aligned}$$

where $\mathbf{D_v^o}$ *is the diagonal matrix containing the vertex degrees which are defined (analogous to Eq. 2.12) as:*

$$\begin{aligned} d^o(v_i) &= \sum_{v_j \in V} \mathbf{A_r}(v_i, v_j) \\ &= \sum_{v_j \in V} \sum_{g \in G} \mathbf{H}(g, v_i)(1/c(g))\mathbf{H}(g, v_j) \\ &= \sum_{g \in G} \mathbf{H}(g, v_i)(1/c(g)) \sum_{v_j \in V} \mathbf{H}(g, v_j) \\ &= \sum_{g \in G} \mathbf{H}(g, v_i)(1/c(g))c(g) \\ &= \sum_{g \in G} \mathbf{H}(g, v_i) = \mathbf{D_v}(v_i, v_i) \ , \end{aligned}$$

which amounts to this nice property:

$$\mathbf{D_v^o} = \mathbf{D_v} \ . \tag{2.19}$$

Notice that $\mathbf{D_v}$ *is the hypergraph vertex degree matrix and that this hypergraph is unweighted. Given the above setting, the graph* N_o *is the Bolla WCE graph with the adjacency matrix* $\mathbf{A_o}$ *and the normalized Lapacian* $\mathbf{L_o}$.

Both the Rodríguez and Bolla WCE graphs start with an unweighted hypergraph. However, one can consider a setting where the original hypergraph is weighted. In this case, one simple scheme to design weight of a WCE graph edge would be to simple aggregate the weights of the hyperedges common between the edge's vertices. This setting is the most standard clique expansion setting. In most literature when CE or WCE graphs are being discussed, they are *de facto* referring to this strategy: an initial weighted hypergraph and hyperedge weight summation as the weight generation scheme during WCE. However, we refer it to as Gibson WCE graph as it was first used by Gibson et al. (2000). We define it more formally below.

Definition 33 ***Gibson WCE graph (Gibson et al., 2000):*** *Consider a weighted CE graph ,* $N_s = (V, E_s, w^s)$ *for given weighted hypergraph* $H = (V, G, w)$. *A simple scheme would be to assign weight* w^c_{ij} *as the aggregate of inverse of the cardinalities of all the hyperedges common between the vertices* v_i *and* v_j *viz.:*

$$w^s_{ij} = \sum_{g \in G} \mathbf{H}(g, v_i) \cdot w(g) \cdot \mathbf{H}(g, v_j) \ , \tag{2.20}$$

which amounts to the following adjacency matrix:

$$\mathbf{A_s} = \mathbf{H^\intercal W H} \ , \tag{2.21}$$

where $\mathbf{W}$ *is the hypergraph weight matrix (see Eq. 64). The normalized Gibsons's Laplacian is:*

$$\mathbf{L_s} = (\mathbf{I} - \mathbf{D_v^s}^{-1/2} \mathbf{A_s} \mathbf{D_v^s}^{-1/2}) \tag{2.22}$$

$$= (\mathbf{I} - \mathbf{D_v}^{-1/2} \mathbf{A_s} \mathbf{D_v}^{-1/2}) \ , \tag{2.23}$$

where $\mathbf{D_v^s}$ *is the diagonal matrix containing the vertex degrees which are defined (analogous to Eq. 2.12) as:*

$$\begin{aligned} d^s(v_i) &= \sum_{v_j \in V} \mathbf{A_s}(v_i, v_j) \\ &= \sum_{v_j \in V} \sum_{g \in G} \mathbf{H}(g, v_i)(w(g))\mathbf{H}(g, v_j) \\ &= \sum_{g \in G} \mathbf{H}(g, v_i)(w(g)) \sum_{v_j \in V} \mathbf{H}(g, v_j) \\ &= \sum_{g \in G} \mathbf{H}(g, v_i) w(g) c(g) \\ &= \sum_{g \in G} \mathbf{H}(g, v_i) \cdot \mathbf{W}(g, g) \cdot \mathbf{D_e}(g, g) \ , \end{aligned}$$

which amounts to this nice property:

$$\mathbf{D_v^s} = \mathbf{H^\intercal W D_e} \ . \tag{2.24}$$

Given the above setting, the graph N_s *is the Gibson WCE graph with the adjacency matrix* $\mathbf{A_s}$ *and the normalized Laplacian* $\mathbf{L_s}$.

To summarize, in this subsection we formalize the clique expansion of a hypergraph by associating the tuple containing the adjacency matrix and the Laplacian to the resulting clique expanded graph viz. $[\mathbf{A_{wc}}, \mathbf{L_{wc}}]$. Both the original hypergraph and the resulting clique expanded graph can be weighted or unweighted. Depending on the scheme to assign weights to $\mathbf{A_{wc}}$, we arrive at variety of WCE graphs. We have provide examples of three of the most popular WCE graphs. If the original hypergraph is assumed unweighted, then the two choice of weights, Equations 2.13 and 2.17, result in the Rodrıguez and Bolla WCE graphs, $[\mathbf{A_r}, \mathbf{L_r}]$ and $[\mathbf{A_o}, \mathbf{L_o}]$, respectively. On the contrary, if we starti with a weighted hypergraph and asuume the weighting scheme in Equatrion 2.20, we arrive at the Gibson WCE graph viz. $[\mathbf{A_s}, \mathbf{L_s}]$. Lastly, its is important to note that WCE graph, $N_{wc} = (V, E_c, w^c)$, itself generalizes the unweighted

CE graph, $N_c = (V, E_c)$, when we assign each edge's weight in WCE as one:

$$w_{ij}^{c} = \mathbf{A_{wc}}(v_i, v_j) = 1 = \mathbf{A_c}(v_i, v_j) \ , \forall (v_i, v_j) \in E_c \tag{2.25}$$

and the Laplacian for the CE graph becomes:

$$\mathbf{L_c} = (\mathbf{I} - \mathbf{D_v^c}^{-1/2} \mathbf{A_c} \mathbf{D_v^c}^{-1/2}) \ . \tag{2.26}$$

2.2.2 Star Expanded Graph

In this subsection we consider the graph model based on the process of *star expansion* (see Def. 9) which we had earlier described in Section 1.2. This process converts a group data into a bipartite graph (see Def. 8). It does so by considering groups and entities both as vertices and then connecting each group vertex to all the entity vertices corresponding to the entities which are member of that group. We revisit this process and analyse it in a more structured manner and generalise it for both weighted and unweighted hypergraphs. We therefore, redefine the star expansion process by defining the adjacency matrix of associated with the output unweighted bipartite graph.

Definition 34 ***Star Expanded Graph (SE Graph):*** *Consider an unweighted hypergraph $H = (V, G)$ or a weighted hypergraph $H = (V, G, w)$. We consider each hyperedge as a vertex resulting into a second vertex set $V_g = \{v_{g_j} : g_j \in G\}$. Let us then consider a bipartite graph $N_b = (V_b, E_b)$, where $V_b = (V \cup V_g)$ and E_b as the membership edges between hyperedge vertices and its member vertices viz.:*

$$E_b = \{(v_i, v_{g_j}) : v_i \in V, v_{g_j} \in V_g, v_i \in g_j\} \ . \tag{2.27}$$

The adjacency matrix of this bipartite graph, $\mathbf{A_b} \in \{0,1\}^{(|V|+|E|)\times(|V|+|E|)}$, is defined in

terms of the incidence matrix ($\mathbf{H}$) *of the hypergraph* H*, as:*

$$\mathbf{A_b} = \left[\begin{array}{c|c} \mathbf{0}_{|V|} & \mathbf{H^T} \\ \hline \mathbf{H} & \mathbf{0}_{|E|} \end{array}\right] \tag{2.28}$$

We also associate an incidence matrix, $\mathbf{H_b} \in \{0,1\}^{|E_b| \times (|V|+|E|)}$.

$$\mathbf{H_b}(e_k, v_i) = \mathbf{H_b}(e_k, v_{g_j}) = \begin{cases} 1 & \text{if } e_k = (v_i, v_{g_j}) \in E_b \\ 0 & \text{otherwise .} \end{cases} \tag{2.29}$$

This bipartite graph $N_b = (V, E_b)$*, defined by the matrices* $\mathbf{A_b}$ *and* $\mathbf{H_b}$*, constitute the star expansion (SE) graph defined previously (see Def. 9).*

Analogous to clique expansion (see Equation 2.15), we shall also associate a normalized Laplacian matrix based on the SE graph adjacency matrix as follows:

$$\mathbf{L_b} = (\mathbf{I} - \mathbf{D_v^b}^{-1/2} \mathbf{A_b} \mathbf{D_v^b}^{-1/2}) \ , \tag{2.30}$$

where $\mathbf{D_v^b}$ is the diagonal matrix containing the vertex degrees for the two sets of vertices in V_b, defined as:

$$\begin{aligned} d^b(v_i) &= \sum_{v_{g_j} \in V_g} \mathbf{A_{wc}}(v_{g_j}, v_i) \\ &= \sum_{v_{g_j} \in V_g} \mathbf{H}(v_{g_j}, v_i) \\ &= d(v_i) \end{aligned}$$

$$
\begin{aligned}
d^b(v_{g_j}) &= \sum_{v_i \in V} \mathbf{A_{wc}}(v_{g_j}, v_i) \\
&= \sum_{v_i \in V} \mathbf{H}(v_{g_j}, v_i) \\
&= c(g_j) \ ,
\end{aligned}
$$

and

$$
\mathbf{D_v^b} = \left[\begin{array}{c|c} \mathbf{D_v} & \mathbf{0}_{|V|} \\ \hline \mathbf{0}_{|E|} & \mathbf{D_e} \end{array}\right] \tag{2.31}
$$

which is simply the i-th row sum of the adjacency matrix $\mathbf{A_b}$ and $\mathbf{D_v}$, $\mathbf{D_e}$ are diagonal degree matrices of the hypergraph H. Equations 2.28 and 2.30, therefore, define the unweighted SE graph. We can analogously define a weighted SE graph as follows.

Definition 35 ***Weighted Star Expanded Graph (WSE Graph):*** *Given a weighted hypergraph $H = (V, G, w)$, consider a weighted bipartite graph $N_b = (V_b, E_b, w^b)$, where $V_b = (V \cup V_g)$ and E_b as the membership edges between hyperedge vertices and its member vertices viz.:*

$$
E_b = \{(v_i, v_{g_j}) : v_i \in V, v_{g_j} \in V_g, v_i \in g_j\} \ . \tag{2.32}
$$

Also let $\mathbf{W_b} \in \mathbb{R}^{|E| \times |V|}$ be the weight matrix with elements $\mathbf{W_b}(v_{g_i}, v_i) = w^b(v_{g_i}, v_i)$. Then the weighted adjacency matrix of this bipartite graph, $\mathbf{A_{wb}} \in \mathbb{R}^{(|V|+|E|)\times(|V|+|E|)}$, is defined in terms of the incidence matrix ($\mathbf{H}$) of the hypergraph H and weight matrix $\mathbf{W_b}$, as:

$$
\mathbf{A_{wb}} = \left[\begin{array}{c|c} \mathbf{0}_{|V|} & \mathbf{W_b^T H} \\ \hline \mathbf{H W_b} & \mathbf{0}_{|E|} \end{array}\right] \tag{2.33}
$$

The associated incidence matrix, $\mathbf{H_{wb}} \in \{0,1\}^{|E_b| \times (|V|+|E|)}$ remains same as in the unweighted SE case viz. $\mathbf{H_{wb}} = \mathbf{H_b}$. This bipartite graph $N_{wb} = (V, E_b, w^b)$, defined by the matrices $\mathbf{A_{wb}}$ and $\mathbf{H_{wb}}$, constitute the weighted star expansion (SE) graph.

Analogous to SE graph (see Equation 2.30), we shall also associate a normalized

Laplacian matrix. Note, like in the case of CE graphs, we will use the same notations $d^b(\cdot)$ and $\mathbf{D_v^b}$ for vertex degrees of both weighted SE as well as unweighted SE graphs; depending on the context it will be clear.

Based on the WSE graph adjacency matrix, we define the Laplacian as follows:

$$\mathbf{L_{wb}} = (\mathbf{I} - \mathbf{D_v^b}^{-1/2}\mathbf{A_{wb}}\mathbf{D_v^b}^{-1/2}) \ , \tag{2.34}$$

where $\mathbf{D_v^b}$ is the diagonal matrix containing the vertex degrees for the two sets of vertices in V_b, defined as:

$$\begin{aligned}(\mathbf{D_v^b})^V(v_i) = d^b(v_i) &= \sum_{v_{g_j} \in V_g} \mathbf{A_{wb}}(v_{g_j}, v_i) \\ &= \sum_{v_{g_j} \in V_g} \mathbf{H}(v_{g_j}, v_i) \cdot \mathbf{W_b}(v_{g_j}, v_i)\end{aligned}$$

$$\begin{aligned}(\mathbf{D_v^b})^{V_g}(v_{g_j}) = d^b(v_{g_j}) &= \sum_{v_i \in V} \mathbf{A_{wb}}(v_{g_j}, v_i) \\ &= \sum_{v_i \in V} \mathbf{H}(v_{g_j}, v_i) \cdot \mathbf{W_b}(v_{g_j}, v_i)\end{aligned}$$

and

$$\mathbf{D_v^b} = \left[\begin{array}{c|c} (\mathbf{D_v^b})^V & \mathbf{0}_{|V|} \\ \hline \mathbf{0}_{|E|} & (\mathbf{D_v^b})^{V_g} \end{array}\right] \tag{2.35}$$

which is simply the i-th row sum of the adjacency $\mathbf{A_{wb}}$ and $[(\mathbf{D_v^b})^V, (\mathbf{D_v^b})^{V_g}]$ are diagonal degree matrices for the two sets of vertices in the bipartite graph, $[V, V_b]$, respectively. One important point to note is that WSE graph generalize SE graph and that if we take $w^b(v_{g_j}, v_i) = 1, \forall (v_{g_j}, v_i) \in E_b$, they become identical. Further, Equations 2.33 and 2.34 define a general template for the weighted SE graph but does

not provided any definition for the edge weights. Next we provide a couple of most popular weight assignment schemes. First we describe the most standard scheme, which is to assign scaled hyperedge weights (Zien et al., 1999)–formally:

Definition 36 ***Zien Weighted Star Expanded (WSE) Graph (Zien et al., 1999):*** *Given a weighted hypergraph $H = (V, G, w)$ and the corresponding WSE graph $N_b = (V_b, E_b, w^b)$. Consider the w^b function:*

$$\mathbf{W_b^{zn}}(v_{g_i}, v_i) = w^b(v_{g_i}, v_i) = w(g_i)/c(g_i) \ , \tag{2.36}$$

where the bipartite edge is given a fraction of the hyperedge weight based on hyperedge's cardinality. Interestingly, in this case the degrees of the hyperedge vertices is simply the hyperedge cardinalities viz.:

$$\begin{aligned}
(\mathbf{D_v^b})^{V_g}(v_{g_j}) = d^b(v_{g_j}) &= \sum_{v_i \in V} \mathbf{A_{wb}}(v_{g_j}, v_i) \\
&= \sum_{v_i \in V} \mathbf{H}(v_{g_j}, v_i) \cdot \mathbf{W_b}(v_{g_j}, v_i) \\
&= w(g_i)/c(g_i) \cdot \left(\sum_{v_i \in V} \mathbf{H}(v_{g_j}, v_i) \right) \\
&= \big(w(g_i)/c(g_i)\big) \cdot \big(c(g_i)\big) \\
&= w(g_i),
\end{aligned}$$

therefore, $(\mathbf{D_v^b})^{V_g} = \mathbf{D_e}$. This WSE graph N_b with the weight matrix $\mathbf{W_b^{zn}}$ constitute a Zien WSE graph.

Another method of WSE edge weighting scheme could be to simply assigning the weight of hyperedge as the weight of its star expanded children bipartite edges. This weighting was used by Zhou et al. (2006b) and its explicit connection to bipartite graphs was highlighted by Agarwal et al. (2006b). We describe it formally now.

Definition 37 ***Zhou Weighted Star Expanded (WSE) Graph (Zhou et al., 2006b):*** *Given a weighted hypergraph $H = (V, G, w)$ and the corresponding WSE graph $N_z = (V_z, E_z, w^z)$. Consider the w^z function:*

$$\mathbf{W_b^z}(v_{g_i}, v_i) = w^b(v_{g_i}, v_i) = w(g_i) \ , \tag{2.37}$$

where the bipartite edge weight is simply its parent hyperedge weight. This basically results in the following WSE adjacency martix:

$$\mathbf{A_z} = \left[\begin{array}{c|c} \mathbf{0}_{|V|} & \mathbf{H^T W} \\ \hline \mathbf{HW} & \mathbf{0}_{|E|} \end{array}\right] \ , \tag{2.38}$$

where $\mathbf{W}$ is the diagonal matrix of hyperedge weights and the following degrees:

$$\begin{aligned} d^z(v_i) &= \sum_{v_{g_j} \in V_g} \mathbf{A_z}(v_{g_j}, v_i) \\ &= \sum_{v_{g_j} \in V_g} \mathbf{H}(v_{g_j}, v_i) \cdot \mathbf{W_b}(v_{g_j}, v_i) \\ &= \sum_{v_{g_j} \in V_g} \mathbf{H}(v_{g_j}, v_i) \cdot w(g_i) \\ &= d(v_i) \end{aligned}$$

$$\begin{aligned} d^z(v_{g_j}) = d^b(v_{g_j}) &= \sum_{v_i \in V} \mathbf{A_z}(v_{g_j}, v_i) \\ &= \sum_{v_i \in V} \mathbf{H}(v_{g_j}, v_i) \cdot \mathbf{W_b}(v_{g_j}, v_i) \\ &= \sum_{v_i \in V} \mathbf{H}(v_{g_j}, v_i) \cdot w(g_j) \\ &= w(g_j) \cdot \left(\sum_{v_i \in V} \mathbf{H}(v_{g_j}, v_i) \right) \\ &= w(g_j) \cdot c(g_j) \end{aligned}$$

and the degree matrix $\mathbf{D_v^z}$

$$\mathbf{D_v^z} = \left[\begin{array}{c|c} \mathbf{D_v} & \mathbf{0}_{|V|} \\ \hline \mathbf{0}_{|E|} & \mathbf{WD_e} \end{array}\right] \tag{2.39}$$

This WSE graph N_z *with the weight matrix* $\mathbf{W_b^z}$ *constitute a Zhou WSE graph. Based on the WSE graph adjacency matrix, we define the Zhou's WCE Laplacian as follows:*

$$\mathbf{L_z} = (\mathbf{I} - \mathbf{D_v^z}^{-1/2}\mathbf{A_z}\mathbf{D_v^z}^{-1/2}) \ . \tag{2.40}$$

The Laplacian studied by Zhou et al. (2006b), although based on the Zhou's WSE Laplacian, L_z that we just described, is only based on the eigenvectors associated with only the subset of vertices V and not the entire V_z. This connection was established by Agarwal et al. (2006a) and this Laplacian based only on the vertices V is popularly known as the *hypergraph Laplacian*, $\mathbf{L_h}$, which we define as follows.

Definition 38 ***Zhou Proxy Hypergraph Laplacian (Agarwal et al., 2006a; Zhou et al., 2006b):*** *Given a Zhou WSE hypergraph* $N_z = (V_z, E_z, w^z)$*., the normalized Laplacian associated with it can be written as follows:*

$$\begin{aligned}
\mathbf{L_z} &= (\mathbf{I} - \mathbf{D_v^z}^{-1/2}\mathbf{A_z}\mathbf{D_v^z}^{-1/2}) \\
&= \mathbf{I} - \left[\begin{array}{c|c} \mathbf{D_v}^{-1/2} & \mathbf{0}_{|V|} \\ \hline \mathbf{0}_{|E|} & (\mathbf{WD_e})^{-1/2} \end{array}\right] \left[\begin{array}{c|c} \mathbf{0}_{|V|} & \mathbf{H^T W} \\ \hline \mathbf{HW} & \mathbf{0}_{|E|} \end{array}\right] \left[\begin{array}{c|c} \mathbf{D_v}^{-1/2} & \mathbf{0}_{|V|} \\ \hline \mathbf{0}_{|E|} & (\mathbf{WD_e})^{-1/2} \end{array}\right] \\
&= \left[\begin{array}{c|c} \mathbf{I} & -\mathbf{D_v}^{-1/2}\mathbf{H^T W^{1/2}}\mathbf{D_e}^{-1/2} \\ \hline -\mathbf{D_e}^{-1/2}\mathbf{HW^{1/2}}\mathbf{D_v}^{-1/2} & \mathbf{I} \end{array}\right]
\end{aligned}$$

Now if we consider the eigenvalue problem for this matrix, $\mathbf{L_z}$*, with eigenvectors,* $\mathbf{x^T} = [\mathbf{x}_V, \mathbf{x}_{V_g}]$*, then Agarwal et al. (2006a) show that the eigen-equation for only the* V*'s eigenvectors is:*

$$\left(\mathbf{I} - \mathbf{D_v^{-1/2}}\mathbf{H^T W D_e^{-1}} \mathbf{H D_v^{-1/2}}\right)\mathbf{x}_V = \lambda \mathbf{x}_V \ . \tag{2.41}$$

Then the following matrix, $\mathbf{L_{hyp}} \in \mathbb{R}^{|V|\times|V|}$*, is popularly called the hypergraph Laplacian because of its resemblance with the simple graph Laplacian:*

$$\mathbf{L_{hyp}} = \mathbf{I} - \mathbf{D_v^{-1/2}} \mathbf{H^T} \mathbf{W} \mathbf{D_e^{-1}} \mathbf{H} \mathbf{D_v^{-1/2}} \ . \tag{2.42}$$

We also refer to $\mathbf{L_{hyp}}$ as the "proxy" hypergraph Laplacian as it models only the vertex-vertex relationship and is not higher-order, somethings we had pointed out previously on various occasions. Interestingly, the underlying graph of the hypergraph Laplacian is also a CE graph. Therefore, this hypergraph Laplacian also becomes a cornerstone for the relationship between the clique and star expansions as studied in detail, again, by Agarwal et al. (2006a). Further, Zhou et al. (2006b) also associate an adjacency matrix with a hypergraph as:

$$\mathbf{A_{hyp}} = (\mathbf{H}^T \mathbf{W_e} \mathbf{H} - \mathbf{D_v}) \ , \tag{2.43}$$

which we refer to as the *proxy hypergraph adjacency matrix.*

In analogy with proxy hypergraph Laplacian we just defined, we also define the dual hypergraph Laplacian. Remember in dual hypergraph the role of vertices and hyperedges are interchanged. The dual an incidence matrix is $\mathbf{H_{dual}} \in \{0,1\}^{|V|\times|G|}$ such that $\mathbf{H_{dual}} = \mathbf{H}^\intercal$ (see Def. 23).

Definition 39 ***Proxy Hypergraph Dual Laplacian:*** *We define the dual hypergraph Laplacian* $\mathbf{L_{dual}} \in \mathbb{R}^{|G|\times|G|}$*:*

$$\mathbf{L_{dual}} = \mathbf{I} - \mathbf{D_e^{-1/2}} \mathbf{H} \mathbf{W_v} \mathbf{D_v^{-1}} \mathbf{H^T} \mathbf{D_e^{-1/2}} \ , \tag{2.44}$$

where $\mathbf{W_v}$ *is a diagonal matrix containing the weights of each vertex and* $\mathbf{D_e}$ *is a diagonal matrix containing the degree of each hyperedge.*

For the purpose of the work presented in this thesis, we assume no weights on the nodes and take $\mathbf{W_v} = \mathbf{I}$. Analogously, we also associate an adjacency matrix with a

hypergraph dual as:

$$\mathbf{A_{dual}} = (\mathbf{H_{dual}}^T \mathbf{W_v} \mathbf{H_{dual}} - \mathbf{D_e}) = (\mathbf{H} \mathbf{W_v} \mathbf{H}^T - \mathbf{D_e}) \ , \tag{2.45}$$

which we refer to as the *proxy dual adjacency matrix*.

2.2.3 Line Graphs of Hypergraphs

In this subsection we consider the graph model based on the process of *line expansion* (see Def. 11) which we had earlier described in Section 1.2. This process converts a group data into a line graph (see Def. 10). It does so by considering groups as vertices and then connecting two groups by an edge if they have entities in common. We revisit this process and analyse it in a more structured manner and generalise it for both weighted and unweighted hypergraphs. We therefore, redefine the line expansion process by defining the adjacency matrix of associated with the output unweighted line graph.

Definition 40 ***Line Expanded Graph (Line or LE Graph):*** *Consider an unweighted hypergraph $H = (V, G)$ or a weighted hypergraph $H = (V, G, w)$. Let us define a matrix $\mathbf{J} = \mathbf{H}\mathbf{H}^\intercal - \mathbf{D_e}$. Then we define the adjacency matrix of the line expanded simple graph, $\mathbf{A_l} \in \{0,1\}^{|G|\times|G|}$, as:*

$$\mathbf{A_l}(v_{g_i}, v_{g_j}) = \begin{cases} 1 & \text{if } \mathbf{J}(v_{g_i}, v_{g_j}) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.46}$$

This is the adjacency matrix associated with the line expanded simple graph $N_l = (V_g, E_l)$ where $V_g = \{v_{g_j} : g_j \in G\}$ (see Def. 11).

Analogous to clique expansion (see Equation 2.15), we shall also associate a normalized Laplacian matrix based on the LE graph adjacency matrix as follows:

$$\mathbf{L_l} = (\mathbf{I} - {\mathbf{D_v^l}}^{-1/2} \mathbf{A_l} {\mathbf{D_v^l}}^{-1/2}) \ , \tag{2.47}$$

where $\mathbf{D_v^c}$ is the diagonal matrix containing the vertex degrees which are defined as:

$$d^l(v_{g_i}) = \sum_{v_{g_j} \in G} \mathbf{A_l}(v_{g_i}, v_{g_j}) \ , = \psi(g_i) \tag{2.48}$$

which is simply the i-th row sum of the adjacency matrix $\mathbf{A_l}$ and is equivalent to the hyperedge hyperdegree, $\psi(\cdot)$ (see Definition 26). Equations 2.46 and 2.47, therefore, define the unweighted LE graph. We can analogously define a weighted LE graph as follows.

Definition 41 ***Weighted Line Expanded Graph (WLE Graph):*** *For a given LE graph $N_l = (V, E_l)$ from a hypergraph $H = (V, G)$, we can define an undirected weighted LE graph as $N_{wl} = (V, E_l, w^l)$, where w^l is a weight function that assigns a scalar weight w^l_{ij} to each edge $(v_{g_i}, v_{g_j}) \in E_l$ viz. $w^l_{ij} = w^l(v_{g_i}, v_{g_j})$. The adjacency matrix of this graph is, $\mathbf{A_{wl}} \in \mathbb{R}^{|G| \times |G|}$, as:*

$$\mathbf{A_{wl}}(v_{g_i}, v_{g_j}) = w^l_{ij} \ . \tag{2.49}$$

Further, we will also associate a normalized Laplacian matrix for a given WLE graph:

$$\mathbf{L_{wl}} = (\mathbf{I} - \mathbf{D_v^l}^{-1/2} \mathbf{A_{wl}} \mathbf{D_v^l}^{-1/2}) \ , \tag{2.50}$$

where $\mathbf{D_v^l}$ is the diagonal matrix containing the vertex degrees which are defined as:

$$d^l(v_{g_i}) = \sum_{v_{g_j} \in G} \mathbf{A_{wl}}(v_{g_i}, v_{g_j}) \ , \tag{2.51}$$

which is simply the i-th row sum of the adjacency matrix $\mathbf{A_{wl}}$. One important point to note is that WLE graph generalize LE graph and that if we take $w^l_{ij} = 1, \forall i, j \in [|G|]$, they become identical. Further, Equations 2.49 and 2.54 define the template for the weighted LE graph but does not provided any definition for the edge weights. Next we provide a couple of most popular weight assignment schemes. First we describe the

most standard scheme, which is to assign weights as the number of vertices common between the pair of hyperedges –formally:

Definition 42 ***Weighted Intersection Graph (WI Graph):*** *Given a unweighted hypergraph $H = (V, G)$ and the corresponding WLE graph $N_{wl} = (V_l, E_l, w^l)$. We define a weighted Intersection graph as a WLE graph with the adjacency matrix, $\mathbf{A_{in}} \in \mathbb{R}^{|G|\times|G|}$, as:*

$$\mathbf{A_{in}}(v_{g_i}, v_{g_i}) = w^l(v_{g_i}, v_{g_j}) = |g_i \cap g_i| \ , \tag{2.52}$$

where the intersection graph edge is given weight based on intersection of hyperedges corresponding to the two hyperedge vertices. This also amounts to the following simple matrix formula:

$$\mathbf{A_{in}} = \mathbf{H}\mathbf{H}^\intercal - \mathbf{D_e} \ . \tag{2.53}$$

The normalized Laplacian matrix for a given WI graph:

$$\mathbf{L_{wl}} = (\mathbf{I} - \mathbf{D_v^{in}}^{-1/2}\mathbf{A_{in}}\mathbf{D_v^{in}}^{-1/2}) \ , \tag{2.54}$$

where $\mathbf{D_v^{in}}$ is the diagonal matrix containing the vertex degrees which are defined as:

$$\begin{aligned} d^{in}(v_{g_i}) &= \sum_{v_{g_j}\in G} \mathbf{A_{in}}(v_{g_i}, v_{g_j}) \\ &= \sum_{v_{g_j}\in G} |g_i \cap g_i| \\ &\leq c(g_i)\psi(g_i) \ , \end{aligned}$$

in other words,

$$\mathbf{D_v^{in}} \leq \mathbf{D_e}\mathbf{\Psi_e} \ . \tag{2.55}$$

There can be other WLE edge weighting scheme that also take into account the cardinality as well as the weights of hyperedges. Some examples are as follows. We can

use mean of the proportions of the two hyperedge's intersection viz.

$$\mathbf{A_{in}}(v_{g_i}, v_{g_i}) = w^l(v_{g_i}, v_{g_j}) = \frac{1}{2}\left(\frac{|g_i \cap g_i|}{c(g_i)} + \frac{|g_i \cap g_i|}{c(g_j)}\right) . \tag{2.56}$$

Rather than taking mean, the same can be proportioned by the hyperedge weights as well, resulting to:

$$\mathbf{A_{in}}(v_{g_i}, v_{g_i}) = w^l(v_{g_i}, v_{g_j}) = \left(\frac{|g_i \cap g_i| \cdot w(g_i)}{c(g_i)} + \frac{|g_i \cap g_i| \cdot w(g_j)}{c(g_j)}\right) . \tag{2.57}$$

2.2.4 Hasse Diagrams

Several studies related to group data involve modeling of not just groups (hyperedges) but also subgroups (subhyperedges) (Moore et al., 2012; Ramanathan et al., 2011; Sharma et al., 2017). In the graph based models discussed till now, we have not considered subhyperedges (subsets of hyperedges). In this section we will introduce the concept of *simplicial complex* as well as the *Hasse Diagram* associated with it. These structures are able to model subhyperedges along with hyperedges and vertices.

Simplicial complex are a specialization of hypergraphs (Munkres, 1984) (Figure 2.1), in which additionally each hyperedge has the subset closure property, i.e., each subset of hyperedge (subhyperedge) is also a valid hyperedge. First let us define a subhyperedge and a simplicial complex for a given hypergraph. As in the previous sections, we assume that the original group data has been already translated to a hypergraph.

Definition 43 ***Subhyperedge:*** *Consider a given a unweighted hypergraph $H = (V, G)$ or weighted hypergraph $H = (V, G, w)$. For a given hyperedge $g \in G$, we define a subhyperedge, s, as a proper subset of hyperedge g, viz.*

$$s \subset g \quad s.t. \quad (s \neq \phi) \wedge (s \neq g) . \tag{2.58}$$

Definition 44 ***Abstract Simplicial Complex (SC):*** *Consider a given a unweighted hypergraph $H = (V, G)$. We denote by $S_i = \{s_k^i, \forall k \in \{1, 2, ..., 2^{|g_i|} - 2\}\}$ the set of all*

proper subsets of each hyperedge $g_i \in G$. If we consider the union of all subsets of the hyperedges in G along with G itself, i.e., $C = \{G \cup (\bigcup_{i=1}^{m} S_i)\}$, then we have a (abstract) simplicial complex C and each element $c \in C$ is a simplex which represents a hyperedge or subhyperedge.

The simplicial complex corresponding to a hypergraph basically appends the subhyperedge (equivalently subgroup in group data) information in the hypergraph by enumerating the subhyperedges. Therefore, SC model is a hypergraph model enabled with subhyperedges. To demarcate the transition from original hypergraph, H to the SC, C, we shall also classify the simplices in SC into two groups. First, we simply have the set of hyperedges, G in original hypergraph which are a subset of SC, C, by Definition 44, viz. $G \subset C$. Next we define the set containing the simplices or subhyperedges in C that were never observed in the original hypergraph H, viz.,

$$C_s = \{c | (c \in C) \wedge (c \notin G)\} = (C - G) \ , \tag{2.59}$$

and to summarize:

$$C = G \cup C_s \ . \tag{2.60}$$

To give a concrete example, see Figure 2.1, $C = \{C_1, ..., C_{19}\}$, $G = \{g_1, g_2, g_3\} = \{C_6, C_{19}, C_{18}\}$ and $C_s = (C - G)$.

Next we also classify the simplices in SC one the basis of their cardinality.

Definition 45 ***Rank and Anti-rank of Simplicial Complex:*** *We define K_{max} or $r(C)$ as the rank of the simplicial complex, C, as follows:*

$$K_{max} = r(C) = \max_{c \in C} |c| \ , \tag{2.61}$$

and define K_{min} or $s(C)$ as the anti-rank of the simplicial complex, C, as:

$$K_{min} = s(C) = \min_{c \in C} |c| \ . \tag{2.62}$$

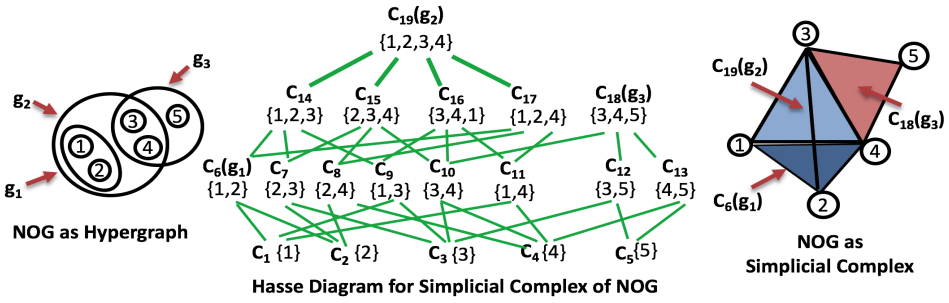

Figure 2.1: Example illustrating a network of groups hypergraph (left) as a simplicial complex (right) and as a Hasse diagram (middle) corresponding to the simplicial complex, for a scenario where the actors {1,2,3,4,5} have collaborated in the past as groups: $g_1 = \{1, 2\}$, $g_2 = \{1, 2, 3, 4\}$ and $g_3 = \{3, 4, 5\}$.

Definition 46 ***Dimension of Simplicial Complex:*** *For a simplex (or (sub)-hyperedge) $\alpha \in C$ we define its dimension as $dim(\alpha) = |\alpha| - 1$. If K_{max} is the maximum cardinality of any simplex in C (i.e. rank(C)) then $(K_{max} - 1)$ is the maximum dimension of any simplex in C or simply the dimension of C.*

Definition 47 ***Set of k-cardinality simplices:*** *Given a simplicial complex C, the set of simplices of cardinality k within the simplicial complex C are defined by the set:*

$$\pi^{\mathbf{k}} = \{\sigma | \sigma \in C \wedge |\sigma| = k\}, \forall k \in \{1, ..., K_{max}\} \ , \tag{2.63}$$

where K_{max} is the rank of the simplicial complex, C

Thus far we have only provided definitions for the abstract simplicial complex but we have yet to develop a model to capture it. As this section is dedicated to graph based models, we here develop a graph based model to capture the SC. For this purpose we treat each simplex in the SC as a vertex of a graph and connect any two vertices the simplices or subhyperedges they represent have an intersection and a *cardinality difference of exactly one.* The resulting graph is called the *Hasse Diagram (HD)* or *Hasse Lattice* of the original simplicial complex (Sharma et al., 2017; Zax, 2012). Further, analogous to other proxy graph expansion schemes in previous subsections, we refer this process of conversion from SC to HD as *Hasse Expansion (HE)*, where HD is

synonymous to HE graph. We define it more formally as follows.

Definition 48 ***Hasse Expanded (HE) Graph (Hasse Diagram) (Sharma et al., 2017):*** *Consider an unweighted hypergraph $H = (V, G)$ and its simplicial complex C. We then define a simple graph called Hasse diagram, $N_h = (V_h, E_h)$, over the vertex set $V_h = \{v_{c_j} : c_j \in C\}$ and with a set of undirected edges:*

$$E_h = \{(v_{c_i}, v_{c_j}) \cup (v_{c_j}, v_{c_i}) | (v_{c_i}, v_{c_j} \in V_h) \wedge (c_i \subset c_j) \wedge (|c_i| = |c_j| - 1)\} \tag{2.64}$$

. The level in the diagram (Figure 2.1) determines the poset relation. Then we define the adjacency matrix of the Hasse expanded simple graph, $\mathbf{A_h} \in \{0, 1\}^{|C| \times |C|}$, as:

$$\mathbf{A_h}(v_{c_i}, v_{c_j}) = \begin{cases} 1 & \text{if } (c_i \subset c_j) \wedge (|c_i| = |c_j| - 1) \\ 1 & \text{if } (c_j \subset c_i) \wedge (|c_j| = |c_i| - 1) \\ 0 & \text{otherwise .} \end{cases} \tag{2.65}$$

Analogous to clique expansion (see Equation 2.15), we shall also associate a normalized Laplacian matrix based on the HE graph adjacency matrix as follows:

$$\mathbf{L_h} = (\mathbf{I} - \mathbf{D_v^h}^{-1/2} \mathbf{A_h} \mathbf{D_v^h}^{-1/2}) \ , \tag{2.66}$$

where $\mathbf{D_v^h}$ is the diagonal matrix containing the vertex degrees which are defined as:

$$d^h(v_{g_i}) = \sum_{v_{c_j} \in C} \mathbf{A_h}(v_{c_i}, v_{c_j}) \ , \tag{2.67}$$

which is simply the i-th row sum of the adjacency matrix $\mathbf{A_h}$. Equations 2.65 and 2.66, therefore, define the unweighted HE graph. One important point to notice is that the topology of Hasse diagram is dependent on the cardinality constraint in Equation 2.64. These cardinality constraint intersection edges of the HD are crucial for modeling and therefore, often its better to leave these edges unweighted. Weighted edges might in-

terfere with the degree of enforcement of the cardinality constraint. However, a more appropriate model is using weights over vertices rather than weighted edges. We therefore, next describe weighted HD with vertex weights. These weighted HD can be derived from both weighted as well as unweighted hypergraph or SC. Let us first define a weighted simplicial complex.

Definition 49 ***Weighted Simplicial Complex (WSC)*** ***(Sharma et al., 2017):*** *Consider an unweighted hypergraph $H = (V, G)$ or a weighted hypergraph $H = (V, G, w)$. Let C be the simplicial complex associated with H. If we also associate a weight $w^x(c) \in \mathbb{R}, \forall c \in C$, then we attain a Weighted Simplicial Complex, $\diamondsuit = (C, w^x)$.*

Next, based on WSC we can define its Weighted Hasse Daigram as follows:

Definition 50 ***Weighted Hasse Diagram (WHD)*** ***(Sharma et al., 2017):*** *Consider an unweighted hypergraph $H = (V, G)$ or a weighted hypergraph $H = (V, G, w)$. Let $\diamondsuit = (C, w^x)$ be the weighted simplicial complex associated with H. Also let $N_h = (V_h, E_h)$ be the unweighted Hasse diagram associated with the SC, C. The we define a weighted Hasse diagram for the weighted SC, $\diamondsuit$, as $N_{wh} = (V_h, E_h, w_h)$ such that each vertex of HD gets weight of the corresponding simplex it represent in the WSC:*

$$w^h(v_{c_i}) = w^x(c_i) \ . \tag{2.68}$$

Again nether of the above definitions for WSC or WHD describe any weight assignment scheme. Here we provide example of an intuitive approach for assigning weights given a general weighted HG. It is also applicable for unweighted HG by assuming unit weights for the hyperedges. Let us first define a function that maps hyperedges G with simplex in C_s. Each $c \in C_s$ also has a set of groups $Q(c) \subseteq G$, of which it is a subgroup of, viz.:.

$$Q(c) = \{x | (x \in G) \wedge (c \subset x)\} \ . \tag{2.69}$$

We then define our weight function w^x which gives the weights to all the simplices or (sub)hyperedges in C, given the hyperedges G and their weight function w, as follows:

$$w^x(c) = \begin{cases} w(c) + \left(\sum_{x \in Q(c)} w(x) \right) & \text{when } c \in G \\ \sum_{x \in Q(c)} w(x) & \text{when } c \in C_s \end{cases} \tag{2.70}$$

In words, for a hyperedge in the original hypergraph, H, we simply take its weight, $w(c)$, and also add the weights of the hyperedges it is a subset of. In the case of subhyperedge (which is not a hyperedge itself) we simply add the counts of the hyperedges it is a subset of. For example, Sharma et al. (2017) use the the number of times a hyperedge has been observed in the group data as its weight, $w(c)$, and then develop a WSC model using the simplex weighting scheme in Equation 2.70.

2.3 Hypergraph Models

In the last Section 2.2 we discussed various graph-based models for hypergraph representing group data. All these graph "proxy" models although partly capture some relationships pertaining to group data like node-node, node-group or group-group relations, but either fail to capture them simultaneously. We had noted this observation as the conclusion of Section 1.2. Further, more importantly they do not model the joint interaction of nodes within a hyperedge. In fact all these proxy graph techniques try to approximate hyperedge or set-level information with dyadic edge-level information, resulting in loss of information.

In this section, we will develop models that capture hypergraph, unlike the previous section's proxy approaches, in a more head on fashion. Proxy graph methods use two dimensional affinity matrices to capture dyadic edge-level information, on similar lines to capture higher-order (hyperedge-level) information we employ higher-order affinity matrices called *tensors*.

In fact, we leverage the knowledge that a k-way tensor can be used to represent a k-uniform hypergraph (Cooper & Dutle, 2012); and to capture hypergraphs in a principled

manner. This fact was first employed in the computer vision community by Shashua et al. (2006). Before defining our model, let us first provide some basic definitions.

Definition 51 ***Rank and Anti-rank of Hypergraph (Berge, 1984):*** *We define K_{max} or $r(C)$ as the rank of the hypergraph, $H = (V, G)$, as follows:*

$$K_{max} = r(H) = \max_{g \in G} c(g) \ , \tag{2.71}$$

and define K_{min} or $s(C)$ as the anti-rank of the simplicial complex, C, as:

$$K_{min} = s(H) = \min_{g \in G} c(g) \ . \tag{2.72}$$

Definition 52 ***k-uniform hypergraph (k-graph) (Berge, 1984):*** *A hypergraph $H = (V, G)$ is k-uniform if each hyperedge has cardinality k, viz.:*

$$c(g) = k \ \ , \forall g \in G \ . \tag{2.73}$$

It can also be defined by the following condition using the hypergraph ranks:

$$r(H) = s(H) \ . \tag{2.74}$$

Definition 53 ***Symmetric Tensor (Comon et al., 2008):*** *A (cubical) tensor $\mathcal{A}$ over a set $\mathbb{S}$ of dimension n and order k is a collection of n^k elements $a_{p_1,p_2,..,p_k} \in \mathbb{S}$ where $p_j \in [n]$. A cubical tensor is said to be symmetric if entries which use the same index sets are the same. That is $\mathcal{A}$ is symmetric if:*

$$a_{p_1,p_2,..,p_k} = a_{p_{\sigma(1)},p_{\sigma(2)},..,p_{\sigma(k)}} \ , \forall \sigma \in \mathfrak{S}_k \ , \tag{2.75}$$

where $\mathfrak{S}_k$ is the symmetric group on $[k]$.

With the basic definitions in place, let us now define the hypergraph model using tensors. We start with first uniform hypergraph model, and then later generalize it for

non-uniform (or general) hypergraphs.

Definition 54 ***Uniform Hypergraph Tensor (UHT):*** *Consider a weighted k-uniform hypergraph or k-graph, $H^k = (V^k, G^k, w^k)$. Corresponding to this k-uniform hypergraph, we can define a k^{th} order n-dimensional symmetric tensor $\mathcal{A}^k_{\boldsymbol{hyp}} = (a_{p_1,p_2,..,p_k}) \in \mathbb{R}^{[k,n]}$ whose elements are as follows:*

$$a_{p_1,p_2,..,p_k} = \begin{cases} w^k(g_i) & for\ \{v_{p_1}, v_{p_2}, ..., v_{p_k}\} \in g_i \in G^k \\ 0 & otherwise\ \ , \end{cases} \tag{2.76}$$

where $|g_i| = k, \forall i \in \{1, ..., |G^k|\}$. We refer to this tensor $\mathcal{A}^k_{\boldsymbol{hyp}}$ as k-uniform hypergraph tensor.

Note that symmetry here implies that the value of element $a_{p_1,p_2,..,p_k}$ is invariant under any permutation of its indices $(p_1, p_2, .., p_k)$. Therefore, for any given hyperedge $g_i \in G$, we initialize all the indices in the tensor $\mathcal{A}^k_{\mathbf{hyp}}$ which correspond to any permutation of its vertices, viz., $p_{\sigma(1)}, p_{\sigma(2)}, .., p_{\sigma(k)}$, $\forall \sigma \in \mathfrak{S}_k$, with the hyperedge's weight, $w^k(g_i)$. There are various choices of weights possible. If we take $w^k(g_i) = 1$, then the resulting hypergraph tensor, $\mathcal{A}^k_{\mathbf{hyp}}$, is nothing but a higher-order adjacency matrix, also called *adjacency hypermatrix*. This is also the unweighted hypergraph case where all hyperedge are assigned unit weight. A special case of this adjacency hypermatrix is the *normalized adjacency hypermatrix* (Cooper & Dutle, 2012) where have the following weighting scheme:

$$w^k(g_i) = \frac{1}{(k-1)!} \tag{2.77}$$

is used. For any other weights, derived from domain for example, these hypergraph tensors are also referred to as *affinity hypermatrices* or *affinity tensors* of the hypergraph (Ghoshdastidar & Dukkipati, 2016; Shashua et al., 2006).

Following Ballard et al. (2011), we also define the lexicographically ordered index

set for hyperedges:

$$\mathcal{P}^k = \big\{\mathbf{p}|\mathbf{p} = (p_1, p_2, .., p_k) \text{ where } \{v_{p_1}, v_{p_2}, ..., v_{p_k}\} \in g_i, \\ \forall g_i \in G \text{ s.t. } |g_i| = k \text{ and } p_1 < p_2 < ... < p_k\big\} \ . \tag{2.78}$$

Each index, $\mathbf{p}$, in the index set, $\mathcal{P}^k$, uniquely represent all the tensor indices associated with a given hyperedge. For example for a hyperedge $g_i = (v_4, v_7, v_9)$, the index is $\mathbf{p} = (4, 7, 9)$, which represents the tensor indices: $\{(4, 7, 9), (4, 9, 7), (7, 4, 9), (7, 9, 4), (9, 4, 7), (9, 7, 4)\}$, all of which correspond to the entries in the tensor, $\mathcal{A}^3_{\mathbf{hyp}}$, initialized for the hyperedge $g_i = (v_4, v_7, v_9)$. Notice that $\mathcal{P}^k$ contains unique (non-repetitive) indexes as there is only a single $\mathbf{p}$ corresponding to each of the different hyperedges $g_i \in G^k$. Consequently, we have $|\mathcal{P}^k| = |G^k|$.

Uniform Hypergraph Tensor, as defined in Definition 54, therefore, defines a model for uniform hypergraph using symmetric tensors. We now generalize it for non-uniform (general) hypergraphs. For that we first define a *partial hypergraph*.

Definition 55 ***Partial Hypergraph (Berge, 1984):*** *Given a hypergraph $H = (V, G)$ consider a set $J \subset G$. Then we can define another hypergraph $H' = (V', J)$, where V' is:*

$$V' = \{v | v \in V \wedge (\exists\ e \in J \,|\, v \in e)\} \ . \tag{2.79}$$

This hypergraph H' is called the partial hypergraph generated by the set J.

Definition 56 ***k-Uniform Partial Hypergraph (or k-partial):*** *Given a hypergraph $H = (V, G)$ consider the set of all hyperedges of cardinality k viz. a set $G^k \subset G$ such that $c(g) = k$, $\forall g \in G^k$. Then we obtain a partial hypergraph $H^k = (V^k, G^k)$ generated by the hyperedge set G^k, which is also a k-graph. We call this hypergraph H^k as the k-Uniform Partial Hypergraph or k-partial of the original hypergraph H.*

With the above definitions in place, we now define the general (non-uniform) hypergraph model in terms of tensors.

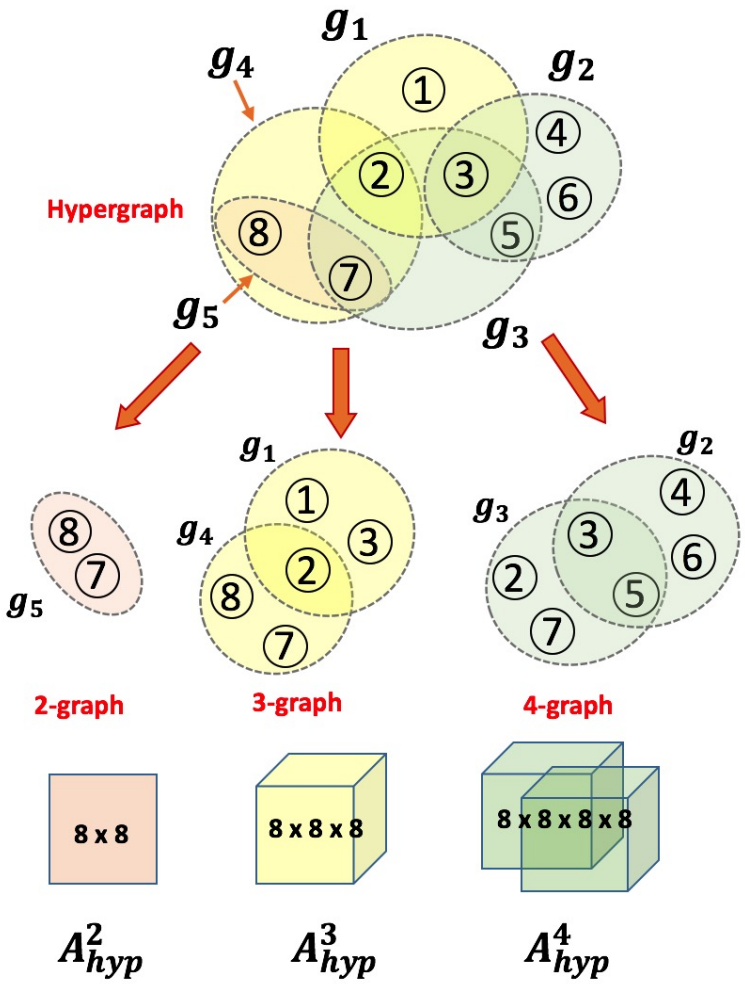

Figure 2.2: Thesis diagram describing the conversion of a hypergraph, $H = (V, G)$ to its hypergraph tensor model $\mathcal{A}_{\mathbf{hyp}}$. The setup is as follows:

- $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$
- $G = \{g_1, g_2, g_3, g_4, g_5\}$
- $\mathcal{A}^2_{\mathbf{hyp}}$ with $\mathcal{P}^2 = \{(7, 8)\}$
- $\mathcal{A}^3_{\mathbf{hyp}}$ with $\mathcal{P}^3 = \{(2, 7, 8), (1, 2, 3)\}$
- $\mathcal{A}^4_{\mathbf{hyp}}$ with $\mathcal{P}^4 = \{(3, 4, 5, 6), (2, 3, 5, 7)\}$

Definition 57 ***Non-uniform Hypergraph Tensor (HT) model:*** *We have a weighted hypergraph* $H = (V, G, w)$ *with rank,* K_{max}*, and anti-rank,* K_{min}*. Consider the set of k-partial hypergraphs for* $k = (K_{min}, ..., K_{max})$ *as* $(H^{K_{min}}, .., H^k, .., H^{K_{max}})$*. It is easy to see that* $H = (H^{K_{min}} \cup \cdot\cdot \cup H^k \cup \cdot\cdot \cup H^{K_{max}})$*. weighted k-uniform hypergraph or k-graph,* $H^k = (V^k, G^k, w^k)$*. Taking together the collection of uniform hypergraph tensors, corresponding to these k-partial hypergraphs, viz.* $\mathcal{A}_{\boldsymbol{hyp}} = (\mathcal{A}_{\boldsymbol{hyp}}^{K_{min}}, \cdots, \mathcal{A}_{\boldsymbol{hyp}}^{K_{max}})$*, constitutes the Non-uniform (General) Hypergraph Tensor Model. Consequently, we also have the collection of ordered index sets:* $\mathcal{P} = (\mathcal{P}^{K_{min}}, \cdots, \mathcal{P}^{K_{max}})$*, associated to it.*

As far as weights are concerned, a point to clarify is that:

$$w^k(g) = w(g) \; \forall \; g \in G \; , \tag{2.80}$$

which amounts from the fact that the weight function, w, is applicable to the entire set G which is also the union of all the k-partial hyperedges, viz. $G = (G^{K_{min}} \cup \cdot\cdot \cup G^k \cup \cdot\cdot \cup G^{K_{max}})$. We illustrate the conversion of a hypergraph to a hypergraph tensor model using a toy example in Figure 2.2.

Note on higher-order information retention:

If we compare the tensor-based hypergraph models to the graph-based methods from previous section, it is easy to observe why tensor-based techniques are more effective in retaining higher-order information. Notice that a k-partial hypergraph tensor, $\mathcal{A}_{\mathbf{hyp}}^{\mathbf{k}} \in \mathbb{R}^{[n,k]}$, has the capability to model all the $\binom{n}{k}$ hyperedges. Graph, or 2-graph, based methods, on the other hand are limited to the the $\mathcal{A}_{\mathbf{hyp}}^{\mathbf{2}} \in \mathbb{R}^{[n,2]}$ adjacency matrix, or 2-uniform HT space. In order to fit into this space, they require an approximation step like the various expansion methods (see Definitions 29, 34, 40, etc.). However, tensor-based models, as proposed in this section, have the capacity of directly representing hyperedges without any information destroying procedure like the various expansions described in last section. Therefore, hyperedge-level higher-order information gets easily retained in the higher-order tensors. On similar lines we can also attempt to retain information at

the level of dual hyperedges. Like the hyperedge-level information encodes the various vertices that jointly form a hyperedge, a dual-hyperedge-level information encodes which all hyperedges a vertex is common to or is simultaneously a member of. We therefore, propose another tensor object corresponding to the dual hypergraph in order to retain this complementary higher-order information within dual hyperedges.

Dual Hypergraph Models

Similar to hypergraph tensors, proposed previously, we can also define a *dual (hypergraph) tensor*, corresponding to *hypergraph dual* where the roles of nodes and hyperedges are interchanged. We consider all the hyperedges in the hypergraph dual that are of cardinality k. This basically corresponds to all the vertices in the original hypergraph which have a degree of k, i.e., they are part of exactly k hyperedges in the original hypergraph. Corresponding to this k-uniform hypergraph dual, we can define a k^{th} order m-dimensional symmetric dual tensor $\mathcal{A}^k_{\mathbf{dual}} = (a_{q_1,q_2,..,q_k}) \in \mathbb{R}^{[k,m]}$ whose elements are initialized as follows:

$$a_{q_1,q_2,..,q_k} = 1 \tag{2.81}$$

where $\{g_{q_1}, g_{q_2}, ..., g_{q_k}\} \ni v_j$ and $d(v_j) = k, \forall j \in \{1, ..., n\}$. Note that this tensor is also symmetric and rest all the elements in the tensor are zeros. Again, we define the lexicographically ordered index set for *dual* hyperedges (vertices in the *original* hypegraph):

$$\begin{aligned} \mathcal{Q}^k = \big\{ \mathbf{q} | \mathbf{q} = (q_1, q_2, .., q_k) \text{ where } v_j \in \{g_{q_1}, g_{q_2}, ..., g_{q_k}\}, \\ \forall v_j \in V \text{ s.t. } |d(v_j)| = k \text{ and } q_1 < q_2 < ... < q_k \big\} \ . \end{aligned} \tag{2.82}$$

Again, notice that $\mathcal{Q}^k$ contains unique (non-repetitive) indexes as there is only a single $\mathbf{q}$ corresponding to each of the different dual hyperedge (vertex in the original hypergraph) i.e. $v_i \in V$. Consequently, we have $|\mathcal{Q}^k| = |\{v_i : d(v_i) = k\}|$.

Finally, the entire collection of dual tensors, $\mathcal{A}_{\mathbf{dual}} = \{\mathcal{A}^k_{\mathbf{dual}}\}$, along with their index sets $\{\mathcal{Q}^k\}$, $\forall k \in \{d_{min}, .., d_{max}\}$ (d_{min} and d_{max} are the maximum and the minimum

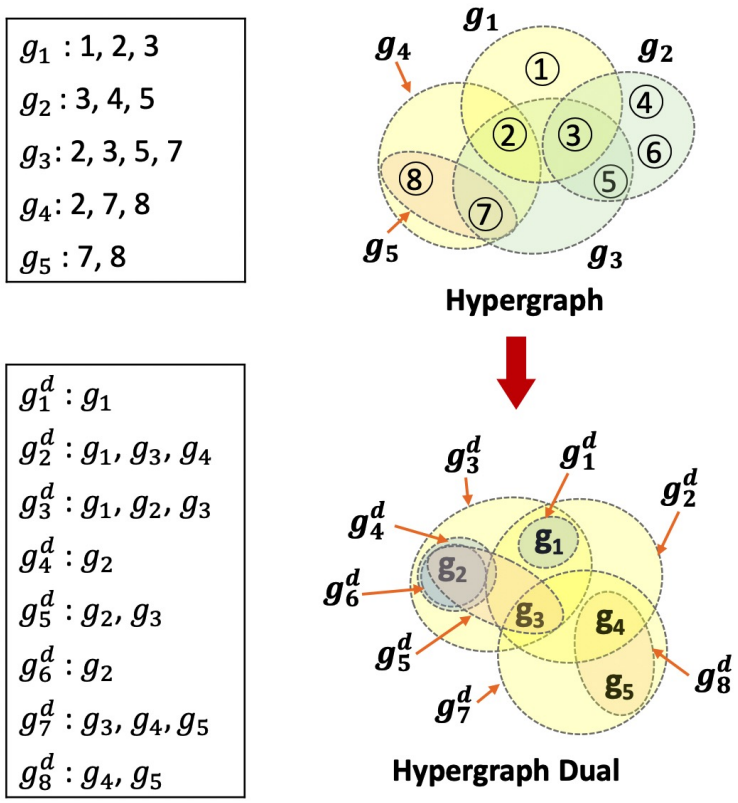

Figure 2.3: Thesis diagram describing the conversion of a hypergraph, $H = (V, G)$ to its dual hypergraph, $H_{dual} = (V_{dual}, G_{dual})$. The setup is as follows:

- $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$
- $G = \{g_1, g_2, g_3, g_4, g_5\}$
- $V_{dual} = \{g_1, g_2, g_3, g_4, g_5\}$
- $G_{dual} = \{g_1^d, g_2^d, g_3^d, g_4^d, g_5^d, g_6^d, g_7^d, g_8^d\}$

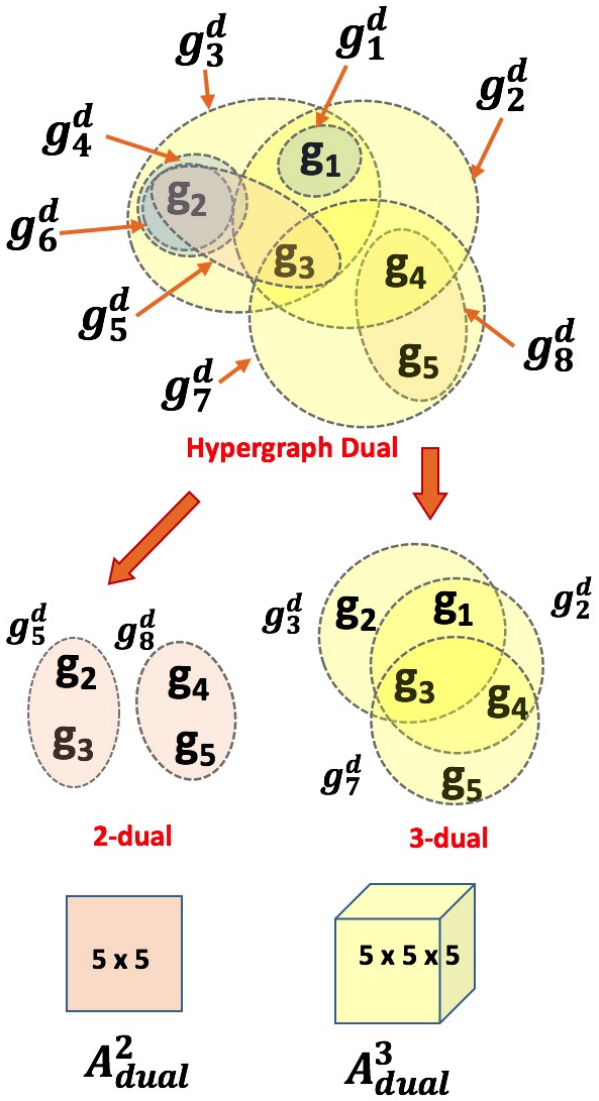

Figure 2.4: Thesis diagram describing the conversion of a hypergraph, $H = (V, G)$ to its dual tensor model $\mathcal{A}_{\mathbf{dual}}$. The setup is as follows:

- $V_{dual} = \{g_1, g_2, g_3, g_4, g_5\}$
- $G_{dual} = \{g_1^d, g_2^d, g_3^d, g_4^d, g_5^d, g_6^d, g_7^d, g_8^d\}$
- $\mathcal{A}^2_{\mathbf{dual}}$, $\mathcal{P}^2 = \{(2,3),(4,5)\}$
- $\mathcal{A}^3_{\mathbf{dual}}$, $\mathcal{P}^3 = \{(1,2,3),(1,3,4),(3,4,5)\}$

vertex degree in the original hypergraph), constitutes the *dual hypergraph tensor model.*

Definition 58 ***Non-uniform Dual Tensor (DT) model:*** *We have a hypergraph* $H = (V, G)$ *with vertex degrees ranging from* d_{min} *to* d_{max}*. Correspondingly, we also have a dual hypergraph* $H^* = (G, V)$ *with cardinalities ranging from* d_{min} *to* d_{max}*. Consider the set of d-partial dual hypergraphs for* $d = (d_{min}, ..., d_{max})$ *as* $((H^*)^{d_{min}}, .., (H^*)^d, .., (H^*)^{d_{max}})$*. It is easy to see that* $H^* = ((H^*)^{d_{min}} \cup \cdot\cdot \cup (H^*)^d \cup \cdot\cdot \cup (H^*)^{d_{max}})$*. Taking together the collection of uniform dual tensors, corresponding to these d-partial dual hypergraphs, viz.* $\mathcal{A}_{\boldsymbol{dual}} = (\mathcal{A}_{\boldsymbol{dual}}^{d_{min}}, \cdots, \mathcal{A}_{\boldsymbol{dual}}^{d_{max}})$*, constitutes the Non-uniform (General) Dual Tensor Model. Consequently, we also have the collection of ordered index sets:* $\mathcal{Q} = (\mathcal{Q}^{d_{min}}, \cdots, \mathcal{Q}^{d_{max}})$*, associated to it.*

Using a toy example we illustrate the conversion of a hypergraph to its hypergraph dual (see Figure 2.3) and then the conversion from this dual to it dual tensor model is shown in Figure 2.4.

2.4 Temporal Hypergraph Models

In Section 1.2 we defined various categories of group data that shall be of interest for us and then defined four different abstractions. In previous sections of this chapter, we have developed hypergraph models for only the *static* groups: *Unweighted Static Groups* and *Weighted Groups* abstractions (see Definitions 1 and 2). However, these abstractions do not take into consideration the temporal information. On the other hand the *dynamic* abstractions, namely, *Temporal Groups* (see Definition 3) and *Weighted Temporal Groups*, model time information as well. In this we shall focus on developing hypergraph models for these dynamic or temporal groups. First let us define temporal weighted groups which generalize the temporal groups.

Definition 59 ***(Temporal Weighted Groups)*** *Consider a finite set* $V = \{1, 2, ..., n\}$ *of n entities. We call a subset* $g_i \subseteq V$*, a group. We consider a finite set of m groups* $G = \{g_i | i \in \{1, .., m\}, g_i \subseteq V\}$*. We also consider the temporal activity pattern these groups.*

*We define a group interaction event when the entities of the i^{th} group g_i interacted at time t_j. In terse, we also refer to it as group g_i was **active** at time t_j. We then define group interaction logs as a set of tuples $L = \{(g_i, t_j) : i \in \{1, 2, ...m\}, j \in \{1, 2, ..., T\}\}$ where the tuple (g_i, t_j) implies the group activity or interaction event. We also have a weight associated with each active group g_i at time t_j given by $f_w(g_i, t_j)$ where $f_w(\cdot, \cdot)$ is the temporal group weighting function. Then the tuple (G, V, L, f_w) together define a collection of **temporal weighted groups**.*

Like with static groups, for analyzing group structured data, we treat the temporal group abstractions as a hypergraph as well. First we define a temporal hypergraph.

Definition 60 ***Temporal Hypergraph:*** *Consider a time duration of $[0, (\delta + 1) * T]$, which we divide into various snapshot windows of equal size, δ. Snapshot with index t refers to a time period: $(\delta*(t-1), \delta*t)$. Now consider a set of n vertices $V = \{v_1, ..., v_n\}$. We can define a temporal hypergraph, $\mathscr{H}$, as a collection of static hypergraphs for each snapshot, viz. $\mathscr{H} = (H^{(1)}, \cdots, H^{(T)})$, where $H^{(t)} = (V^{(t)}, G^{(t)}, w_{(t)})$ is the (static) hypergraph corresponding to the snapshot t. Here, $V^{(t)} \subseteq V$ and $G^{(t)}$ is a set of hyperedges where each hyperedge $e_j \in G^{(t)}$ is also $e_j \subseteq V^{(t)}$. We also associate with each hyperedge e_j, a scalar weight $w_{(t)j} = w_{(t)}(e_j)$. Further, $n_t = |V^{(t)}| \leq n$ and $m_t = |G^{(t)}| \leq m$ are the number of vertices and hyperedges in hypergraph for the snapshot t.*

We name the process of conversion from temporal groups to temporal hypergraph as *temporal hypergraph conversion*, which we formally define as follows:

Definition 61 ***Temporal Hypergraph Conversion:*** *The temporal hypergraph conversion algorithm constructs a temporal hypergraph $\mathscr{H}$ from a collection of temporal weighted groups (G, V, L, f_w). This algorithm considers each entity v_i as a vertex, resulting into a vertex set V. We then construct a static hypergraph $H^{(t)} = (V^{(t)}, G^{(t)}, w_{(t)})$*

for each snapshot t by using the group logs that occurred in that snapshot, viz.,

$$G^{(t)} = \{g|(g,t) \in L\} \tag{2.83}$$

$$V^{(t)} = \{v|v \in g, g \in E^{(t)}\} \tag{2.84}$$

$$w_{(t)j} = w_{(t)}(g_j) = f_w(g_j, t) \ . \tag{2.85}$$

After the conversion for each snapshot $t \in [1,T]$, we achieve the collection of static hypergraphs for each snapshot, viz. $\mathscr{H} = (H^{(1)}, \cdots, H^{(T)})$.

We can also convert a temporal hypergraph to a static hypergraph using the conversion defined below.

Definition 62 ***Temporal to Static Hypergraph Conversion:*** *Consider a temporal hypergraph $\mathscr{H} = (H^{(1)}, \cdots, H^{(T)})$, where $H^{(t)} = (V^{(t)}, G^{(t)}, w_{(t)})$ is the (static) hypergraph corresponding to the snapshot t. We can associate to it a static hypergraph $H = (V, G, w)$ such that:*

$$G = \{G^{(1)} \cup \cdots \cup G^{(T)}\} \tag{2.86}$$

$$V = \{V^{(1)} \cup \cdots \cup V^{(T)}\} \tag{2.87}$$

$$w_j = f\big(w_{(1)}(g_j), \cdots, w_{(T)}(g_j)\big) \tag{2.88}$$

where $f(\cdot)$ is some function which outputs a scalar weight for a given group given its weights across all snapshots. In short we can also write:

$$H = H^{(1)} \cup \cdots \cup H^{(T)} \tag{2.89}$$

Once we have performed the conversion of temporal group data to a hypergraph, we shall no longer be addressing group data in any further discussions. We will be concerned only to this hypergraph obtained from the conversion. Like the static case, for temporal data also we will introduce algebraic structures pertaining to temporal hypergraph. One

of the major obstacle in modeling temporal data is to capture time along with topology. In order to capture this extra time dimension, we turn our attention again to tensors, which are multidimensional, or N-way, array (Pearson & Zhang, 2012) and has proven to capture multi-dimensional data effectively (Bader & Kolda, 2007). For example, Tensors allow to handle time as a separate dimension. This provides more flexibility to creatively manipulate the temporal dimension. Moreover, the temporal patterns can be captured using tensors to predict future patterns rather than just immediate future. Recently tensors have already proved effective in predicting temporal link prediction by Dunlavy et al. (2011). This has encouraged us to capture hypergraphs as well as their proxy graphs using tensors where the initial dimensions capture the hypergraph topology and the last dimension captures the temporal information.

We will now gradually visit the various algebraic objects we had developed for static hypergraphs from previous section and try to generalize them to tensor-based temporal hypergraph objects. We shall start with the most basic one hypergraph incidence matrix, $\mathbf{H}$, and generalize it to incidence tensor.

Definition 63 ***Hypergraph Incidence Tensor:*** *Consider a temporal hypergraph $\mathscr{H} = (H^{(1)}, \cdots, H^{(T)})$, where $H^{(t)} = (V^{(t)}, G^{(t)}, w_{(t)})$ is the (static) hypergraph corresponding to the snapshot t. We can associate to it an incidence tensors $\mathcal{H} \in \{0,1\}^{m \times n \times T}$, where $m = |\{G^{(1)} \cup \cdots \cup G^{(T)}\}|$ is the total number of hyperedges across snapshots and $n = |\{V^{(1)} \cup \cdots \cup V^{(T)}\}|$ is the number of vertices of the hypergraph. The (g_i, v_j, t)-th entry of this tensor is 1 if j-th vertex is a part of i-th hyperedge in snapshot t i.e.*

$$\mathcal{H}(g_i, v_j, t) = \begin{cases} 1 & v_j \in g_i, g_i \in G^{(t)} \\ 0 & otherwise \quad , \end{cases} \tag{2.90}$$

Each t-th snapsot slice, $\mathcal{H}(:,:,t)$, of the incidence tensor, therefore, represents the incidence matrix of a the hypergraph snapshot $H^{(t)}$ padded with zero rows and columns in order to make the dimensions of each snapshot slice same $(m \times n)$.

Definition 64 ***Hypergraph Weight Tensor:*** *Consider a temporal hypergraph* $\mathscr{H} = (H^{(1)}, \cdots, H^{(T)})$*, where* $H^{(t)} = (V^{(t)}, G^{(t)}, w_{(t)})$ *is the (static) hypergraph corresponding to the snapshot* t*. We can associate to it a hyperedge weight tensor,* $\mathcal{W} \in \mathbb{R}^{m \times m \times T}$*, whose each snapshot is a diagonal, such that:*

$$\mathcal{W}(g, g, t) = w_{(t)}(g) \ , \ \forall g \in G \ . \tag{2.91}$$

2.4.1 Temporal Graphs for Hypergraphs

In Section 2.2 we had developed various graph-based proxy hypergraph models. We had proposed four possible expansions of the original hypergraph resulting into four different graphs: Clique, Bipartite, Line and Hasse Diagrams. Here we generalize these graphs to temporal graphs. In order to do so we reconsider the four weighted adjacency matrices which define these proxy graphs: $\mathbf{A_{wc}}$, $\mathbf{A_{wb}}$, $\mathbf{A_{wl}}$ and $\mathbf{A_h}$. We will generalize these matrices to tensors by adding the time dimension.

Again consider a temporal hypergraph $\mathscr{H} = (H^{(1)}, \cdots, H^{(T)})$, where each $H^{(t)} = (V^{(t)}, G^{(t)}, w_{(t)})$ is the (static) hypergraph corresponding to the snapshot t. Also let $H = (V, G, w)$ be the static hypergraph associated to this temporal hypergraph, $\mathscr{H}$. Further, to reiterate $n_t = |V^{(t)}|$, $m_t = |G^{(t)}|$, $n = |V|$ and $m = |G|$.

Given the above setup, we start with the weighted clique expanded graph (WCE-graph) and define a *Temporal Adjacency Tensor*, $\mathcal{T}_{\mathbf{wc}} \in \mathbb{R}^{n \times n \times T}$ as follows:

$$\mathcal{T}_{\mathbf{wc}}(v_i, v_j, t) = \begin{cases} \mathbf{A}^{(\mathbf{t})}_{\mathbf{wc}}(v_i, v_j) & \mathrm{v}_i, v_j \in V_c^{(t)} \\ 0 & \mathrm{v}_i, v_j \notin V_c^{(t)} \ , \end{cases} \tag{2.92}$$

where $\mathbf{A}^{(\mathbf{t})}_{\mathbf{wc}} \in \mathbb{R}^{n_t \times n_t}$ is the adjacency matrix of the WCE-graph, $N_c^{(t)} = (V_c^{(t)}, G_c^{(t)}, w^c_{(t)})$, for the t-th snapshot's static hypergraph $H^{(t)}$. We then have, $\mathscr{N}_c = (N_c^{(1)}, \cdots, N_c^{(T)})$, as the *Temporal WCE* graph corresponding to the temporal hypergraph $\mathscr{H}$ and has the temporal adjacency tensor, $\mathcal{T}_{\mathbf{wc}}$, associated to it.

Next we consider the weighted star expanded graph (WSE-graph) and define a

temporal adjacency tensor, $\mathcal{T}_{\mathbf{wb}} \in \mathbb{R}^{(m+n)\times(m+n)\times T}$ as follows:

$$\mathcal{T}_{\mathbf{wb}}(v_{g_i}, v_j, t) = \begin{cases} \mathbf{A}^{(\mathbf{t})}_{\mathbf{wb}}(v_{g_i}, v_j) & \mathrm{v}_{g_i}, v_j \in V_b^{(t)} \\ 0 & \mathrm{v}_{g_i}, v_j \notin V_b^{(t)} \end{cases}, \tag{2.93}$$

where $\mathbf{A}^{(\mathbf{t})}_{\mathbf{wb}} \in \mathbb{R}^{(m_t+n_t)\times(m_t+n_t)}$ is the adjacency matrix of the WSE-graph, $N_b^{(t)} = (V_b^{(t)}, G_b^{(t)}, w^b_{(t)})$, for the t-th snapshot's static hypergraph $H^{(t)}$. We then have, $\mathscr{N}_b = (N_b^{(1)}, \cdots, N_b^{(T)})$, as the *Temporal WSE* graph corresponding to the temporal hypergraph $\mathscr{H}$ and has the temporal adjacency tensor, $\mathcal{T}_{\mathbf{wb}}$, associated to it.

Next we consider the weighted line expanded graph (WLE-graph) and define a temporal adjacency tensor, $\mathcal{T}_{\mathbf{wl}} \in \mathbb{R}^{m\times m\times T}$ as follows:

$$\mathcal{T}_{\mathbf{wl}}(v_{g_i}, v_{g_j}, t) = \begin{cases} \mathbf{A}^{(\mathbf{t})}_{\mathbf{wl}}(v_{g_i}, v_{g_j}) & \mathrm{v}_{g_i}, v_{g_j} \in V_l^{(t)} \\ 0 & \mathrm{v}_{g_i}, v_{g_j} \notin V_l^{(t)} \end{cases}, \tag{2.94}$$

where $\mathbf{A}^{(\mathbf{t})}_{\mathbf{wl}} \in \mathbb{R}^{m_t\times m_t}$ is the adjacency matrix of the WLE-graph, $N_l^{(t)} = (V_l^{(t)}, G_l^{(t)}, w^l_{(t)})$, for the t-th snapshot's static hypergraph $H^{(t)}$. We then have, $\mathscr{N}_l = (N_l^{(1)}, \cdot\cdot\cdot, N_l^{(T)})$, as the *Temporal WLE* graph corresponding to the temporal hypergraph $\mathscr{H}$ and has the temporal adjacency tensor, $\mathcal{T}_{\mathbf{wl}}$, associated to it.

Next we consider the Hasse expanded graph (HE-graph). Let, $N_h^{(t)} = (V_h^{(t)}, G_h^{(t)})$, be the HE-graph for the t-th snapshot's static hypergraph $H^{(t)}$. We then have, $\mathscr{N}_h = (N_h^{(1)}, \cdots, N_h^{(T)})$, as the *Temporal HE* graph corresponding to the temporal hypergraph $\mathscr{H}$. We also define the static HE graph, $N_h = (V_h, G_h)$ corresponding to the temporal HE-graph, such that $G_h = \{G_h^{(1)} \cup \cdot\cdot \cup G_h^{(T)}\}$ and $V_h = \{V_h^{(1)} \cup \cdot\cdot \cup V_h^{(T)}\}$. Let $n^h = |V_h|$ and $n_t^h = |V_h^{(t)}|$.

Given this setup we can define a temporal adjacency tensor, $\mathcal{T}_{\mathbf{h}} \in \mathbb{R}^{n^h\times n^h\times T}$, corresponding to the HE-graph as follows:

$$\mathcal{T}_{\mathbf{h}}(v_i, v_j, t) = \begin{cases} \mathbf{A}^{(\mathbf{t})}_{\mathbf{h}}(v_i, v_j) & \mathrm{v}_i, v_j \in V_h^{(t)} \\ 0 & \mathrm{v}_i, v_j \notin V_h^{(t)} \end{cases}, \tag{2.95}$$

where $\mathbf{A_h^{(t)}} \in \mathbb{R}^{n_t^h \times n_t^h}$ is the adjacency matrix of the HE-graph, $N_h^{(t)}$.

2.4.2 Temporal Hypergraphs

In Section 2.3 we had developed symmetric tensors based hypergraph models, both for uniform as well as general non-uniform hypergraphs. We had proposed hypergraph tensors for hypergraph as well as its dual. Here we generalize these hypergraph models to temporal hypergraphs. In order to do so we reconsider hypergraph tensor, $\mathcal{A}_{\mathbf{hyp}}$ and its dual $\mathcal{A}_{\mathbf{dual}}$. We will further generalize these tensors to temporal tensors by adding the time dimension.

Definition 65 ***Temporal Nonuniform Hypergraph Tensor (THT) model:*** *Consider a temporal hypergraph $\mathscr{H} = (H^{(1)}, \cdots, H^{(T)})$, where $H^{(t)} = (V^{(t)}, G^{(t)}, w_{(t)})$ is the (static) hypergraph corresponding to the snapshot t. Also let $H = (V, G, w)$ be the static hypergraph associated to this temporal hypergraph, $\mathscr{H}$. Further, to reiterate $n_t = |V^{(t)}|$, $m_t = |G^{(t)}|$, $n = |V|$ and $m = |G|$. Also for the t-th snapshot consider the Non-uniform Hypergraph Tensor Model, $\mathcal{A}_{\boldsymbol{hyp}}^{(t)}$, which is a collection of uniform hypergraph tensors, corresponding to the k-partial hypergraphs for t-th snapshot, viz. $\mathcal{A}_{\boldsymbol{hyp}}^{(t)} = (\mathcal{A}_{\boldsymbol{hyp}}^{K_{min}^t(t)}, \cdots, \mathcal{A}_{\boldsymbol{hyp}}^{K_{max}^t(t)})$. Here the rank, K_{max}^t, and anti-rank, K_{min}^t are dependent on the snapshot index t.*

Given this setup we can define a Temporal Non-uniform Hypergraph Tensor Model, $\mathcal{T}_{\boldsymbol{hyp}} = (\mathcal{T}_{\boldsymbol{hyp}}^{K_{min}}, \cdots, \mathcal{T}_{\boldsymbol{hyp}}^{K_{max}})$, where K_{min} and K_{max} are the rank and anti-ranks of H (the static hypergraph associated with the temporal hypergraph). Here, $\mathcal{T}_{\mathbf{hyp}}^{\mathbf{k}} = (a_{p_1,p_2,..,p_k,t}) \in \mathbb{R}^{[k,n]\times T}$, is a symmetric tensor for all except along its last temporal dimension and is

initialized as follows:

$$a_{p_1,p_2,..,p_k,t} = \begin{cases} \mathcal{A}^{\mathbf{k(t)}}_{\mathbf{hyp}}(v_{p_1},..,v_{p_k}) & v_{p_1},..,v_{p_k} \in V^{k(t)} \wedge k \in [K^t_{min}, K^t_{max}] \\ 0 & otherwise \quad . \end{cases}$$

$$= \begin{cases} w^k_{(t)}(g_i) & for\ \{v_{p_1}, v_{p_2}, ..., v_{p_k}\} \in g_i \in G^{k(t)} \\ 0 & otherwise \quad , \end{cases}$$

where $w^k_{(t)}$ is the weight function associated with $H^{k(t)} = (V^{k(t)}, G^{k(t)}, w^k_{(t)})$, which is the k-partial hypergraph for $H^{(t)}$. Lastly, we also have the collection of ordered index sets for all the snapshots: $\mathcal{P}_{\boldsymbol{temp}} = (\mathcal{P}^{(1)}, \cdots, \mathcal{P}^{(T)})$, where $\mathcal{P}^{(t)}$ is the ordered set associated to $\mathcal{A}^{(t)}_{\boldsymbol{hyp}}$.

Similarly, we can also generalize the hypergraph dual tensor (defined in Section 2.3) to a temporal setting.

Definition 66 ***Temporal Nonuniform Dual Tensor (TDT) model:*** *Consider a temporal hypergraph $\mathscr{H} = (H^{(1)}, \cdots, H^{(T)})$, where $H^{(t)} = (V^{(t)}, G^{(t)}, w_{(t)})$ is the (static) hypergraph corresponding to the snapshot t. Also let $H = (V, G, w)$ be the static hypergraph associated to this temporal hypergraph, $\mathscr{H}$. Further, to reiterate $n_t = |V^{(t)}|$, $m_t = |G^{(t)}|$, $n = |V|$ and $m = |G|$. Also for the t-th snapshot consider the Non-uniform Dual Tensor Model, $\mathcal{A}^{(t)}_{\boldsymbol{dual}}$, which is a collection of uniform dual tensors, corresponding to the d-partial dual hypergraphs for t-th snapshot, viz. $\mathcal{A}^{(t)}_{\boldsymbol{dual}} = (\mathcal{A}^{d^t_{min}(1)}_{\boldsymbol{dual}}, \cdots, \mathcal{A}^{d^t_{max}(T)}_{\boldsymbol{dual}})$. Here cardinalities of the hypergraph dual, $(H^{(t)})^*$, or vice versa degrees of the hypergraph, $H^{(t)}$, are ranging from d^t_{min} to d^t_{max} and are dependent on the snapshot index t.*

Given this setup we can define a Temporal Non-uniform Dual Tensor Model, $\mathcal{T}_{\boldsymbol{dual}} = (\mathcal{T}^{d_{min}}_{\boldsymbol{dual}}, \cdots, \mathcal{T}^{d_{max}}_{\boldsymbol{dual}})$, where d_{min} and d_{max} are the rank and anti-ranks of H^* (the static hypergraph dual associated with the temporal hypergraph). Here, $\mathcal{T}^{\mathbf{k}}_{\mathbf{dual}} = (a_{q_1,q_2,..,q_k,t}) \in \mathbb{R}^{[k,m]\times T}$, is a symmetric tensor for all except along its last temporal dimension and is

initialized as follows:

$$a_{q_1,q_2,..,q_k,t} = \begin{cases} \mathcal{A}^{\mathbf{k(t)}}_{\mathbf{dual}}(g_{q_1},\cdot\cdot,g_{q_k}) & g_{q_1},\cdot\cdot,g_{q_k} \in G^{(t)} \wedge k \in [d^t_{min}, d^t_{max}] \\ 0 & otherwise \quad . \end{cases}$$

$$= \begin{cases} 1 & for\ v_i \in \{g_{q_1}, g_{q_2}, ..., g_{q_k}\}, v_i \in V^{(t)}, d(v_i) = k \\ 0 & otherwise \quad . \end{cases}$$

Lastly, we also have the collection of ordered index sets for all the snapshots: $\mathcal{Q}_{\boldsymbol{temp}} = (\mathcal{Q}^{(1)}, \cdots, \mathcal{Q}^{(T)})$*, where* $\mathcal{Q}^{(t)}$ *is the ordered set associated to* $\mathcal{A}^{(t)}_{\boldsymbol{dual}}$*.*

2.5 Hypergraph Structured Datasets

In the previous sections of this chapter we have focused on the various hypergraph models that are available to capture the various groups abstractions that we had discussed in Section 1.2. In this section we describe the variety of datasets that were employed as well as curated during the research work conducted for this thesis. We divide these datasets into three main categories. First, are those datasets that are derived from the observations of real-world groups interactions. Next section focuses on providing an overview of how these data sets were curated and their basic statistics. Second category is that of attribute derived groups, where the groups are derived on the basis of their features. This we describe in Section 2.5.2. Lastly, in Section 2.5.3, we expound on the synthetic datasets and the various techniques for generating them. All these datasets are publicly available at University of Minnesota's, MESH project website (http://mesh.cs.umn.edu).

2.5.1 Real-world Group Interaction based Hypergraph Data

One of foremost motivations for this thesis has been the understanding of social interactions and therefore, we have primarily focused on social group datasets.

With the advent of high-speed internet, collaborations are no longer restricted by

physical proximity and this internet generated online space has allowed for new types of group interactions more than ever before. A group of individuals, irrespective of their demographics or location, can perform a task online. This task might be writing software code (Contractor, 2013) (or a Wikipedia article or a Google Doc) by a group of coders (or editors), or can be a business meeting involving video chat with colleagues (using services like Skype or Zoom), or it can even be a group of player playing online multi-player games (Ahmed et al., 2011) like World of Warcraft. More importantly, these technologies offer minute-by-minute traces of group interactions over extended periods of time, providing information about both the structure and content of relationships. This includes data on individual characteristics (profile), their connections (social graph) and behavior (individual and interactions). Analysis of this data using latest computational techniques is the rapidly growing area of Computational Social Science (Lazer et al., 2009).

Understanding the dynamics of such small (social) groups is of increasing research interest in various sub-disciplines in the social sciences (Poole et al., 2004). Moreover, this research has applications such as optimizing performance of human groups (Boh et al., 2007; Cheney et al., 2010; Lungeanu et al., 2014) measured by metrics such as "collective intelligence" (Woolley et al., 2010). Social groups are broadly divided into small groups and large groups. A key difference between large groups and small groups is that membership in the former is largely based on identity, i.e. a member identifying himself with the group. In contrast, a small group is defined principally by the (regular) interaction between group members, often driven by some purpose, professional or personal. Moreover, small group formation motives and communication processes, which are task centered, are very different from those involved in building friendship ties in a friendship network or joining a community, e.g., joining a news interest group, being part of a Facebook community, subscribing to a YouTube channel, or publishing within a particular research discipline. Lastly, well-defined small groups, in contrast with large groups, typically have size ≤ 20.

We shall be dealing mostly with small groups data but we have also assorted several

large group data sets as well. We have also further subdivided the small group datasets into various genre based on domain defined categories. We refer each genre of a particular small group as a "subgroup". Further, we refer to large groups as communities and the large group dataset as community dataset. In nutshell, we have three kinds of group datasets: small group datasets, subgroup datasets and community datasets. In the following three subsections we describe each of these datasets along with their collection procedures.

Small Group Datasets

These datasets are listed in Table 2.1 and described in detail in Table 2.2. Sub-genre of small groups are described in Table 2.3 Due to the task specific nature, there is high social interaction between people in small groups. We further divide the datasets according to the nature of their tasks.

Research Collaboration Datasets First kind of small group datasets that we consider are extracted from the publicly available research publication records. These datasets reveal the collaboration between coauthors who constantly interact and work together to publish their research. We construct a hypergraph network where the individual authors are considered nodes and the group of coauthors of a research paper is considered as a hyperedge. The publishing date of a research paper is used to generate temporal statistics. Hence, different instances of the same hyperedge means multiple publications over time. Along with temporal information of a group these datasets also provide their field of research which further help us to generate subhypergraphs based on their fields to evaluate the differences in their properties. We now describe in detail each of the research publication datasets employed in this study.

ArXiv: It is a repository of electronic prints of scientific publications. ArXiv[1] hosts literature from various scientific fields including Physics, Mathematics and Computer Science. We build the dataset by scraping information such as date, title, authors list

[1] https://arxiv.org/

and subject id's of publications from 1992 to 2018 for each scientific field. We consider that the authors of a publication must have interacted and collaborated, thus act as a small group and constitute a hyperedge. In this chapter we study publications from 20xx to 20yy and extracted zzzz papers from which we generated xxx authors and yyy hyperedges. Along with the complete hypergraph which contains publications of all the fields, we also extract the subhypergraphs based on the scientific field of the publication.

DBLP: DBLP (Tang et al., 2008) is a computer science bibliography which provides a comprehensive list of research papers published in computer science conferences. For our experiments we extracted v.10 dataset[2]. Dataset contains list of published papers where each paper is in JSON schema which contains title, authors, publication venue, year, number of citations, references and abstract of that paper. In this chapter we study the publications from 20xx to 20yy and extracted zzzz papers from which we generated xxx authors and yyy hyperedges. We further divide the complete hypergraph of computer science into sub-hypergraphs. The sub-hypergraphs consists of the sub-fields of Computer Science like Machine Learning, Software Engineering, Computer Security etc. The dataset does not explicitly provide the sub-field of a paper. To obtain it we maintain lists[3] of the main venues(conferences) of each major sub-field in computer science. Each paper is allocated to its sub-field based on its publication venue.

Pubmed: PubMed provides MEDLINE database which comprises references and abstracts on medicine, nursing, dentistry, health care systems, and preclinical sciences. We used Entrez Programming Utilities which is a public API to access PubMed. The data can also be directly downloaded from NCBI[4]. For our experiments we scraped the complete Pubmed data from 1900 to 2018. However, in this chapter we study data from 20xx to 20yy and extracted zzzz papers from which we generated xxx authors and yyy hyperedges. The API provides a comprehensive list of subcategories such as Anatomy, Communicable Diseases, Ethics etc. and lists the venues(conferences and journals)

[2] https://www.aminer.cn/citation
[3] https://scholar.google.com/citations?view_op=top_venues&hl=en&vq=eng
[4] https://www.ncbi.nlm.nih.gov/

associated with them. Each research paper contains information of its publishing venue which is used with this list to further divide the complete biomedical dataset into sub-hypergraphs of each subcategory.

US Patent: The United States Patent and Trademark Office is an agency in the U.S. Department of Commerce that issues patents to inventors and businesses for their inventions. They provide USPatent[5] dataset comprising detailed information on U.S. patents granted between January 1963 and December 1999. In this chapter we study patents from 20xx to 20yy and extracted zzzz papers from which we generated xxx authors and yyy hyperedges. Patents are further divided into six main technological categories such as Computers and Communications, Drugs and Medical, Electrical and Electronics, Chemical, Mechanical and Others. The dataset contains the category information of each patent, which can be understood from subcategories.txt present at the website. We further construct sub-hypergraphs based on these categories.

Group Communication Datasets The priniciple activity of groups is to exchange information and ideas collectively in order to make decisions. Nowadays people use such online messaging and email services which have overcome the physical barrier to constantly communicate with multiple people over large distances. We use such publicly available group communication datasets to construct hypergraph network where each user is represented using a node. Each email or group chat is considered to be an idea discussed between the sender and the multiple receivers, therefore all the users linked to it are considered as a hyperedge. The email or message sent date is used to generate temporal statistics. Different instances of the same hyperedge signifies multiple emails and message exchanges between the same group of people over time. We have discussed each group communication datasets below in detail.

Enron Email: It is a corpus created by collecting email communication data between 158 senior management employees of Enron Corporation. The data was acquired by Federal Energy Regulatory Commission during the investigation of company's col-

[5] https://data.nber.org/patents/

lapse and made publicly available afterwards.[[Give Dataset description]]

Eu-mail: The dataset is a collection of emails between members of a European research institution, where each email address has been anonymized and it is represented by a node (Leskovec et al., 2007). The original data source only contains (sender, receiver, timestamp) tuples, where timestamps are recorded at 1-second resolution. All the members involved in an email exchange form a hyperedge. The dataset consists of 1005 nodes and 25148 hyperedges.

Manufacturing Company Email: It is an internal e-mail communication between employees of a mid-sized manufacturing company (Michalski et al., 2011). The email addresses have been anonymized and are represented by nodes. Tthe email communication covers a time period of nine full months of 2010 starting from 2010-01-01 to 2010-09-30. Multiple recipients of the same e-mail are represented as separate rows without distinguishing the recipient type. The sender and the receivers of an email communication form a hyperedge.

Online Collaboration Space Datasets With the advent of cloud services and collaboration platforms, it has become trivial for teams of users to collaborate online. Such resources allow users to work remotely on the same projects and documents for long period of time, which indicates high amount of interaction among team members. We build the hypergraph network using publicly available as well as acquired[how to write this properly] online collaboration datasets, where each team member is represented by a node. We however don't consider whole team to be a hyperedge because generally teams consist of highly interactive small groups. We discuss small group extraction for each online collaboration dataset below.

Github: Often called the social networking site for programmers, Github is a code sharing and publishing platform where programmers commit(edit) changes to the files of a repository(project directory). For our experiments we use a publicly available Github Commit dataset[6] containing timestamped commit information on 16M repositories. The

[6] https://data.world/vmarkovtsev/452-m-commits-on-github

data is in json format, where key represents a repository and value contains a list of commits on that repository. For each repository we construct a time series of its commits by programmers. We then use a density based data clustering algorithm DBSCAN to form groups of programmers who commit in small intervals. We make an assumption that people who commit close to each other must be interacting in a group and thus form a hyperedge. In this chapter we study data from 20xx to 20yy and extracted zzzz papers from which we generated xxx authors and yyy hyperedges.

Gaming Collaboration Massively Multiplayer Online Games (MMOG) typically allows from hundreds to thousands players at the same time to play the same game. These games generally divide gamers in teams/guilds to collaborate and perform specific tasks. Such games provide real time audio chatting and visual aids to locate your team members, cooperate and complete the objectives. Due to the high social interaction involved between team members during the game, a team acts as a small group and represented by a hyperedge. The gaming collaboration datasets that we use for our experiments are proprietary in nature, and are not freely available for general public usage.

Everquest II: It is a proprietary dataset containing anonymized user log data of EQ II players collected over a period of 27 months. EQ II enables social interaction with other players through grouping and the creation of guilds. Player interaction is encouraged by integrated voice chat, a built-in mail system, global chat channels, and a global marketplace. EQ II supports guild formation by introducing special tasks, guild oriented quests and experience points. Such activities are used to extract collaborative small groups (Ahmed et al., 2014), represented by hyperedges.

KingSoft: We acquired Chevaliers Romance III (CR3) MMOG data, a role playing game, from a leading digital gaming company in China, Kingsoft. It allow players to choose from eight character class archetypes with distinct complementary skills and abilities which provides a basis for in game collaboration (Lu et al., 2014). Players socially interact with each other in form of guild systems, which are represented using

hyperedges.

Movie Collaboration Dataset The cast and crew members of a movie work together to create a movie. As observed, usually it takes months to make a movie, throughout that time period cast members work together, regularly as a small group. Each cast member is represented by a node and the complete cast of a movie forms a hyperedge. We use the following publicly available movie collaboration datasets for our research.

IMDB: IMDB[7] is one of the largest movie rating and review platforms. It's an online database of information related to films including cast, production crew and personal biographies, plot summaries. We extract movie title, release date and genre information from basics.tsv and crew members from principals.tsv. Movies typically have genre information which is further used to contruct sub-hypergraphs of each genre.

UCI Movie Dataset: It's a movie rating dataset[8] collected by DEC Systems Research Center over a period of 18 months. After preprocessing we consider 2,465 movies with genre information to construct sub-hypergraphs based on each genre.

Human Group Movement Datasets Emergence of location based social networking platform has enabled us to analyze social and spatial properties of users who live in close proximity to each other. Such platforms allow users to check in at places in their local vicinity, either through a dedicated mobile application or through a website. Users are able to see who is nearby and who has been there before. These platforms allow registered users to connect with their existing friends and also meet with new people based on the places they check-in. For our research we try to find small groups of users who check-in together frequently. We call such small groups, hyperedges. We use publicly available check in data from various platforms discussed below.

Gowalla: It is a location-based social networking website where users share their locations by checking-in. Each line of the dataset consists of a user id, location id and timestamp of check-in. We use a sliding window of two hours to cluster users into a

[7] https://datasets.imdbws.com/
[8] https://archive.ics.uci.edu/ml/datasets/Movie

group if they have same location id. We then use FP-Growth algorithm to extract group of users who check-in together more than the minimum threshold. We assume that if a certain number of people check-in together at the same locations then they must know each other. These users are called nodes and their group is a hyperedge.

Community Datasets

Table 2.4 and Table 2.5 describe the community datasets. Individuals explicitly join communities that organize around specific topics, interests, and affiliations. These individuals do not show high level of interaction within the community as compared to small groups. We further divide the datasets based on the nature of their formation.

Interest based Communities Social Media platforms allow its users to create and maintain their profiles and form connections with other users in the form of friendships or by following them. Users build connections and form groups or circles with like minded people who pursue similar interests through interaction. Due to the limited interaction between users in a group, these groups show properties of a community. We consider such user-defined groups which have more than 2 nodes as communities. All the following dataset have been extracted from SNAP.

Livejournal: It is a Russian social networking service where user lists another user as friend. Users can find or create a community according to their interests. Anyone who joins a community can make posts to it as they would on a regular journal. We use community dataset, where each line in the dataset represents a community (Yang & Leskovec, 2015).

Friendster: Before 2015, Friendster was a social networking website, where users could share videos, photos, messages and blogs with other members. It allowed users to form functional groups to which other users then join to construct virtual communities. Each row in the dataset is a community which is represented by a hyperedge (Yang & Leskovec, 2015).

Orkut: It was a social networking site owned and operated by Google where users

Table 2.1: Small Group Datasets Details

Dataset Name	**Group Activity / Type**	**Hyperedges**	**Group**	**Vertices**	**Actor**
Pubmed	Research Publication	8323170	Group of Researches	4775501	Researcher
USPatent	Patent Publicaiton	1050965	Group of Researches	1012236	Researcher
DBLP	Research Publication	697954	Group of Researches	694150	Researcher
ArXiv	Research Publication	849722	Group of Researches	878663	Researcher
EQ2	Online Game Task	11240-	Group of Game Players	10933-	Game Player
Kingsoft	Online Game Task	969896	Group of Game Players	64310	Game Player
Enron	Email Communication	3016-	Group of Members in an Email Communication	184-	Member of Organization
EU Mail	Email Communication	25148	Group of Members in an Email Communication	1005	Member of Organization
Manufacturing Company Email	Email Communication	5685	Group of Members in an Email Communication	167	Member of Organization
Apache	Code Editing	80910-	Codes Editing Same Software Code	3360-	Software Coder
Github	Code Editing	1145227	Codes Editing Same Software Code	1942330	Software Coder
IMDB	Movie Making	2934033	Movie Making Group	3350792	Movie Actors
UCI Movie Dataset	Movie Making	2463	Movie Making Group	3544	Movie Actors
Gowalla	Human Group Movement	110121	Group Check-in	23580	Individual

Table 2.2: Small Group Datasets Description

Dataset Name	**Group Activity / Type**	**Duration**	**Number of Entities**	**Number of Interactions**	**Hypergraph Network Description**
Pubmed	Research Publication	117 years (1900-2016)	4775501 authors	10373008 publications	Dataset contains publications of medicine, nursing, dentistry, health care systems, and preclinical sciences. Coauthors of a publication are considered as a hyperedge.
USPatent	Patent Publicaiton	37 years (1963-1999)	1012236 authors	1468248 publications	Dataset comprises detailed information on U.S. patents. Coauthors of a publication are considered as a hyperedge.
DBLP	Research Publication	65 years (1953-2017)	694150 authors	834603 publications	DBLP provides a comprehensive list of research papers published in computer science conferences. Coauthors of a publication are considered as a hyperedge.
ArXiv	Research Publication	37 years (1992-2018)	878663 authors	1166695 publications	ArXiv is a repository of electronic prints of scientific publications. Coauthors of a publication are considered as a hyperedge.
EQ2	Online Game Task	-	Group of Game Players	-	Game Player
Kingsoft	Online Game Task	5 months (May 2010 - Sept 2010)	64310 players	18875764 team snapshot	Its a MMOG role playing game where players socially interact in form of guilds, represented by hyperedges.
Enron	Email Communication	-	Group of Members in an Email Communication	-	Member of Organization
EU Mail	Email Communication	28 months Unix time (9/20/2003-12/1/2005)	1005 members	235260 emails	The dataset is a collection of emails between members of a European research institution. Each email contains sender and receivers who together can be represented by a hyperedge.
Manufacturing Company Email	Email Communication	9 months Unix Time (1/1/2010-9/30/2010)	167 employees	58337 emails	It is an internal e-mail communication between employees of a manufacturing company. Each email contains sender and receivers who together can be represented by a hyperedge.
Apache	Code Editing	-	Codes Editing Same Software Code	-	Software Coder
Github	Code Editing	-	-	-	Github is a code sharing and publishing platform where programmers commit(edit) changes to the files of a repository(project directory). Programmers who commit in small intervals are considered as a hyperedge.
IMDB	Movie Making	-	-	-	IMDB is one of the largest movie rating and review platforms. The dataset contains information related to videos(film/TV Series/Documentaries) and their cast members. Cast members of each video type are represented by a hyperedge.
UCI Movie Dataset	Movie Making	-	-	-	It's a movie rating dataset containing information of cast members for each movie. Cast of a movie is represented using a hyperedge.
Gowalla	Human Group Movement	-	23580	-	It is a location-based social networking website where users share their locations by checking-in. Users who check-in together frequently are considered as a small group, therefore represented by a hyperedge.

Table 2.3: Sub-groups of Datasets with details

Dataset Name	Group Activity / Type	Sub Groups	Hypergraph Network Description
Pubmed	Research Publication	Nuclear Medicine, Critical Care, Nursing, Gynecology, Endocrinology, Rheumatology, Neoplasms, Audiolog, Vascular Disease, Neurosurger, Neurolog, Obstetrics, Dermatology, Pathology, Pharmacology, Social Sciences, Traumatology, Veterinary Medicine, Therapeutics, Urology, Psychology, Emergency Medicine, Biochemistry, Pulmonary Medicine, Microbiology, Behavioral Sciences, Orthopedics, Biotechnology, Medicine, General Surgery, Metabolism, Pediatrics, Clinical Laboratory Techniques, Gastroenterology, Cardiology, Public Health, Anesthesiology, Hematology, Anatomy, Epidemiology, Ophthalmology, Geriatrics, Dentistry, Molecular Biology, Genetics, Ethics, Biomedical Engineering, Physiology, Tropical Medicine, Nutritional Sciences, Otolaryngology, Drug Therapy, Virology, Psychiatry, Toxicology, Medical Informatics, Communicable Diseases	The NLM catalog API provides a comprehensive list of venues(conferences and journals) associated with each Broad Subject Terms. The publishing venue information of each research paper is compared with this list to identify its Broad Subject. We construct sub-hypergraphs based on these Broad Subject Terms.
USPatent	Patent Publicaiton	Chemical, Computers and Communications, Drugs and Medical, Electrical and Electronic, Mechanical	The NBER website contains subcategories.txt file which have information about five major technological categories of patents. The dataset appoint categories to each patent. We create sub-hypergraphs based on these categories.
DBLP	Research Publication	Web Mobile and Multimedia Technologies, Computer Networks and Communications, Information Systems, Computational Theory and Mathematics, Biomedical Engineering and Medical Informatics, Computer Security and Cryptography, Machine Learning Data Mining and Artificial Intelligence, Human Computer Interaction, Hardware Robotics and Electronics, Computer Linguistics and Speech Processing, General Computer Science, Signal Processing, Computer Graphics and Computer Aided Design, Image Processing and Computer Vision, Software Engineering	We use Google Scholar to maintain lists of the main venues(conferences/journals) of each major category in computer science and engineering. Each research paper contains publication venue information which is compared with the venue lists to allocate them to a category. We construct sub-hypergraphs based on these categories.
ArXiv	Research Publication	Computer Science, Physics, Mathematics, Statistics, Electrical Engineering and Systems Science, Economics, Quantitative Biology, Quantitative Finance	ArXiv clusters research publications under Broad Subject Terms which are used to construct sub-hypergraphs.
IMDB	Movie Making	Sci-Fi, Crime, Romance, Action, Animation, Music, Adult, Comedy, War, Horror, Film-Noir, Western, News, Reality-TV, Thriller, Adventure, Mystery, Short, Talk-Show, Drama, Other, Documentary, Musical, History, Family, Fantasy, Game-Show, Sport, Biography	Movies in the IMDB dataset typically have genre information based on the recommendation of movie makers and IMDB contributers. We contruct sub-hypergraphs based on these genres.
UCI Movie Dataset	Movie Making	Disaster, Drama, Documentary, Epic, Famlily, Fantasy, History, Horror, Music, Mystry, Noir, Porn, Romantic, Satire, Science Fiction, Surreal, Suspense, West, Action, Adventure, Biopic, Comedy	The dataset contains genres information for movies, which is used to construct sub-hypergraphs

Table 2.4: Community Datasets

Dataset Name	**Group Activity / Type**	**Hyperedges**	**Group**	**Vertices**	**Actor**
LiveJournal	Interest Group	576120	Group of Individuals Sharing interest in Journalism	1147948	Individual
Friendster	Interest Group	1500380	Group of Individuals Sharing interest in Blogs	7944949	Individual
Orkut	Interest Group	11065973	Group of Individuals Sharing posts in Social Media	2322299	Individual
Youtube	Interest Group	14870	Group of Individuals Sharing interest in Online YouTube Videos	52675	Individual
Deezer HR	Interest Group	84	Group of Individuals following music of same genre	54573	Individual
Deezer HU	Interest Group	84	Group of Individuals following music of same genre	47538	Individual
Deezer RO	Interest Group	84	Group of Individuals following music of same genre	41773	Individual
DBLP	Research Community	13423	Researchers who published in same journal/conference	260998	Individual
EVE Corporation Network -	Company Employees -	16386-	Employees of the same corporation	52675-	Individual
Google Plus Circles	Ego Network	463	People in the social circle of the user	23591	Individual
Facebook Circles	Ego Network	191	People in the social circle of the user	2884	Individual
Twitter Circles	Ego Network	3632	People in the social circle of the user	22964	Individual
Twitter Followers	User's followers	70065	Group of individuals who follow a user	70097	Follower
Google Plus Followers	User's followers	123	Group of individuals who follow a user	193235	Follower
Bitcoin Alpha	Ratee's raters	2774	Group of individuals who rate a user	3286	Rater
EVE mentor network	Mentee's of a Mentor	44253	Group of individuals who were mentored by a mentor	77094	Mentee
Wikipedia	Candidate's voters	2379	Group of voters who voted for a candidate	6110	Voter

Table 2.5: Community Datasets Description

Dataset Name	**Group Activity / Type**	**Number of Entities**	**Number of Communities**	**Community Description**
LiveJournal	Interest Group	1147948	576120	Livejournal allows users to find and join or create a new community according to their interests. Each community is represented by a hyperedge.
Friendster	Interest Group	7944949	1500380	It allowed users to form groups which then other users can join to construct virtual communities to share multimedia content. Such groups are represented by a hyperedge.
Orkut	Interest Group	2322299	11065973	Group of Individuals Sharing posts in Social Media
Youtube	Interest Group	52675	14870	Group of Individuals Sharing posts in Social Media
Deezer HR, HU, RO	Interest Group	54573, 47538, 41773	84	Each song is mapped to a genre which is used to compile the preferred genre list of a user through the liked song list. A group of users following a genre acts as a community, which is represented as a hyperedge.
DBLP	Research Community	260998	13423	Researchers who published in same journal/conference become a part of the community based on publishing venue. We use such communities to construct hyperedges.
EVE Corporation Network	Company Employees	52675-	16386-	Employees of the same corporation form community by association. Each community is represented by a hyperedge.
Google Plus Circles	Ego Network	23591	463	It allows users to manually categorize their friends into social circles. Each user can maintain multiple circles. Each of these circles acts as a hyperedge.
Facebook Circles	Ego Network	2884	191	The dataset is made after a survey of 10 users, who were manually asked to identify all the circles to which their friends belonged. These circles are used to construct hyperedges.
Twitter Circles	Ego Network	22964	3632	Twitter allows its users to maintain lists comprising persons, organisations, celebrities, news media, etc. Such lists are represented using hyperedges.
Google Plus Followers	User's followers	193235	123	Google Plus allowed its users to follow other users, such that each user has a list followers. We use these lists to construct hyperedges.
Twitter Followers	User's followers	70097	70065	Group of individuals who follow a user form a follower list with respect to that user. We use these lists to construct hyperedges.
Bitcoin Alpha	Ratee's raters	3286	2774	It allow user's to rate each other to show trust/distrust. For every user we consider all the people who rated them with positive trust value, to be in a single community, which can be represented by a hyperedge.
EVE mentor network	Mentee's of a Mentor	77094	44253	Group of individuals who were mentored by a mentor are represented using a hyperedge.
Wikipedia	Candidate's voters	6110	2379	Group of voters who voted for a candidate are represented by a hyperedge.

could connect to each other. It allowed users to form communities that organize around specific topics, interests, and affiliations which other users explicitly join and share content Feld (1981).

Youtube: It is a video sharing platform where users upload, watch, like and subscribe to the videos. It allows users to form groups which other users can then subscribe (Mislove et al., 2007).

Deezer: Deezer is a French online music streaming service which allows users to connect and form friendships. It contains friendship networks of users from 3 European countries. The dataset is in json format where the key represents user ID and the value represents a set of user's prefereend genres. Users show a personal interest when they like a song. Each song is mapped to a genre which is used to compile the prefererred genre list of a user through the liked song list. Therefore, a group of users following a genre is an interest based group, hence a community. From the dataset we extract 84 communties from the 84 distinct genres for 3 countries.

Affiliation based Communities : Individuals form groups based on being affiliated with a community for eg: researchers publishing in a journals, people working in a company. They might not have high intra community interactions but they form groups based on their association with the community.

DBLP: DBLP is a computer science bibliography which provides a comprehensive list of research papers in computer science. Communities in a scientific domain correspond to people working in common fields of science. However, publication venues serve as good proxies for scientific areas **?**. Although the authors who published on the same venue might not have collaborated with each other but they show their affinity for the the venue and therefore form a community. Each line in the dataset represents a community of users.

EVE dataset: It is a corporation employee dataset, where employees belonging to the same corporation form a community by association.

Google Plus: It was a social network owned by Google. Google plus allows users to

follow each other and group different types of relationships into Circles. We construct two different hypergraphs from the dataset. First, we construct a hypergraph from its follower followee network, wherein we assume that all the users following a user forms a community w.r.t. that user. Second, the user aggregate other users in his/her circle which constitutes a community.

Facebook: It is a social networking where users mutually connect with each other and declare their frienship. Each user has a friend list which contains all of his/her friends. We assume such list to be a community w.r.t. that user.

Twitter: It is a microblogging and social networking service on which users post and interact with messages known as "tweets". It allows users to follow other users. Each user has a "following" list which maintains the list of all the others users which it follows. We extract follower list for each user using this dataset and consider it a community.

Follower Followee network : Such networks allow people to follow each other. We assume that all the people following a person form a community with respect to the followee.

Bitcoin Alpha: It is a peer-to-peer payment network where people trade in cryptocurrencies, by broadcasting digitally signed messages to the network. Since cryptocurrency traders are anonymous, there is a need to maintain a record of users' reputation to prevent transactions with fraudulent and risky users. They allow users to rate other users in a scale of -10 (total distrust) to +10 (total trust). A postive rating signifies that the user is willing to do a transaction with rated user in future. For every user we consider all the people who rated them with positive trust value to be in a single community.

Wikipedia: Its an online encyclopedia editted collaboratively by a community of volunteer editors. It is maintained by group of administrators with some additional access. An election is held in order for a user to become an administrator, where wikipedia community votes and decides who to promote to adminship. All the volunteers

Data	Hyperedges (m)	Vertices (n)	Max. Cardinality	Avg. Cardinality	Max. Vertex Degree	Avg. Vertex Degree
zoo	101	15	10	6.53	83	44
voter	432	16	13	7.91	272	213.69
autism-child	291	14	13	7.46	217	155
autism-adolo	103	14	12	7.59	82	55.86
autism-adult	681	14	13	5.81	504	282.5
synthetic	7020	80	6	3.3	330	290.3

Table 2.6: Hypergraph Statistics for various Datasets

who vote for a particular volunteer forms his community. The network contains all the Wikipedia voting data from the inception of Wikipedia till January 2008.

2.5.2 Attribute derived Hypergraph Data

We consider both real-world as well as synthetic datasets. For the former, we make use of five popular real-world datasets from the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml). The five selected datasets have data points whose feature vectors contain mostly boolean-valued features. (From each of the datasets, we removed the very few non-boolean valued features.) We then consider each data point (sample) as a hyperedge with features as vertices. All the features (vertices) which have value one for a given sample (hyperedge) are considered vertices of this sample hyperedge. In short, we treat the data matrix (sample-feature mapping) as the hypergraph incidence matrix (hyperedge-vertex mapping). Below we describe the five datasets:

1. **zoo**: In this dataset, there are several animals each described with a set of boolean attributes like, for example, does it have a feather, or is it airborne. There are several classes of animals, and the aim is to classify animals correctly into its class.

2. **voter**: In this, the aim is to classify congressman as democrat versus republican based on 16 key votes, where each vote is boolean (yea or nay). Each congressmen's hyperedge contains only "yay" vertices.

3. **autism-child, autism-adolo, autism-adult**: These three datasets contain psychological evaluation on a cohort of children, adolescents, and adults, respectively,

for classification into having ASD disorder or not. Attributes are boolean item responses to behavioral questions. We treat each item (psychological evaluation question) as a vertex, and with each positively responded item (vertex) becomes part of the corresponding individual's hyperedge.

2.5.3 Synthetic Hypergraph Data

For generating **synthetic** hypergraphs, we employ the recently proposed hypergraph stochastic block-model (**hSBM**) (Ghoshdastidar & Dukkipati, 2015) which are generalization of the traditional stochastic graph model to a hypergraph setting. This model is fairly straightforward. Here we describe a slightly augmented process as we need labels for hyperedges (rather than partition labels for vertices in hSBM). We start with a set of vertices and divide them into two equal sets (we restrict ourselves to only two vertex partitions but can be easily extended for more). We then randomly generate hyperedges between any vertices with probability p_{inter} and within vertices in the same partition with probability p_{intra}. In order to generate clusters or communities one chooses $p_{inter} < p_{intra}$, resulting in more dense connections within each partition rather than across partitions. For a given cardinality c, we set our probability vector $\mathbf{p}(c) = [p_{inter}, p_{intra}] = [5, 40] \cdot (\log n / n^{(c-1)})$, where n is the number of vertices and order $O(K \cdot \log n / n^{(c-1))}$ is recommended to realize sparse regime i.e. $O(n \log n)$ hyperedges. The particular value of constant K is chosen to realize not too sparse hypergraphs and a sufficiently high number of higher-order hyperedges. We realize 25 hypergraphs using the above process with roughly $15n \log n$ to $25n \log n$ hyperedges. For our experiments we restrict to $[2, 6]$ cardinality hyperedges (average statistics shown in Table 2.6). We then label hyperedges into three different classes: those consisting of only vertices from first vertex partition, those from an only second partition, and those with a mix of vertices from both partitions. The aim, in this case, is to solve this 3-class classification problem.

Chapter 3

Group Stability and Attrition Prediction: A Cross-sectional Analysis

This chapter serves as the entry point to the first part of this thesis. In this and the next two chapters, we study hypergraph inference mechanisms. Specifically this chapter addresses the problem of *old hyperedge prediction* by dividing it into two sub-problems of group stability prediction and group attrition prediction.

We start by again motivating that the study of small collaborations or teams is an important endeavor both in industry and academia. The social phenomena responsible for formation or evolution of such small groups is quite different from those for dyadic relations like friendship or large size guilds (or communities). In small groups when social actors collaborate for various tasks over time, the actors common across collaborations act as bridges which connect groups into a network of groups. Evolution of groups is affected by this network structure. Building appropriate models for this network is an important problem in the study of group evolution. This work focuses on the problem of group recurrence prediction. In order to overcome the shortcomings of two traditional group network modeling approaches: hypergraph and simplicial complex, we

propose a hybrid approach: *Weighted Simplicial Complex (WSC)*. We develop a *Hasse diagram* based framework to study WSCs and build several predictive models for group recurrence based on this approach. Our results demonstrate the effectiveness of our approach.

Next Section 3.1 introduces and motivates the problem, followed by preliminaries and problem statement in Section 3.2. We discuss the problem formulation and the methodology in Section 3.3. Section 3.4.2 is dedicated to the various experimental evaluations conducted and describes the various datasets employed, which is followed by the the conclusion.

3.1 Introduction

With the advent of high-speed internet, collaborations are no longer restricted by physical proximity. A group of individuals, irrespective of their demographics or location, can perform a task online. This task might be writing software code (or a Wikipedia article or a Google Doc) by a group of coders (or editors), or can be a business meeting involving video chat with colleagues or collaborations in writing paper (Sharma et al., 2014; Simmons et al., 2016; Singhal et al., 2016a;b) or teaming up in online games (Roy et al., 2017; Sharma & Srivastava, 2017; Singhal et al., 2013b). Understanding the dynamics of such small (social) groups is of increasing research interest in various subdisciplines in the social sciences (Poole et al., 2004), and is of interest to applications that require high efficiency in the performance of human groups (Lungeanu et al., 2014; Singhal et al., 2014b).

This chapter addresses the problem of *group evolution*, with specific focus on understanding the causal factors driving the evolution. The overall objective is to build a model that can predict how a group will evolve in the future, based on its history. One aspect of special interest is *group recurrence*, which can be stated thus: *Which group(s) (or its subgroup(s)) among the groups observed so far, will continue to function as a group, i.e. perform some task again in the near future?*

Prior studies have demonstrated the significance of recurrence in network structure (see (Zuckerman, 2004) and references therein). Most work on group evolution in social networks focuses on the evolution of arbitrary size communities or groups (Patil et al., 2013b). The sizes of these groups are usually large and the boundaries of the community depends on the definition of membership used. In this chapter we study well-defined small groups which typically have size ≤ 20. A key difference between large groups and small groups is that membership in the former is largely based on identity, i.e. a member identifying himself with the group. In contrast, a small group is defined principally by the (regular) interaction between group members, often driven by some purpose, professional or personal. The focus of this paper is to study the evolution of small groups and, in contrast to classical social science literature, the objective is to build models that can predict future behavior, with the final goal of identifying potential causal mechanisms for small group evolution.

In contrast to prior work, we highlight the distinct nature of small groups and develop models inspired from social science theories of small groups (Poole et al., 2004). A group can be formed depending on the requirement (fiat teams) or a set of actors can make an autonomous decision to work together (self assembly (Contractor, 2013)). In either case, individuals find it easier to work with familiar actors (Lungeanu et al., 2014), making *frequency* of activity by a group an important metric. Also, over time, actors build new relationships while working in different groups. A shared collaboration history is therefore created, where the same individuals are part of multiple groups, acting as bridges between groups, and resulting in a *network of groups (NOG)* (Figure **??**). This is the *network* perspective of small groups (Monge & Contractor, 2003) where the network of groups plays a central role in the group formation process. Moreover, group formation motives and group communication processes, which are *task centered*, are very different from those involved in building friendship ties in a friendship network or joining a community, e.g., joining a news interest group, being part of a Facebook community, subscribing to a Youtube channel, or publishing within a particular research discipline (Singhal & Srivastava, 2014; Singhal et al., 2013a; 2017). Recently,

some attempts have been made to model networks as higher order relational structures such as simplicial complexes (Moore et al., 2012; Ramanathan et al., 2011) and hypergraphs (Sharma et al., 2014; 2015). A hypergraph is a generalized graph where edges, now called hyperdges, instead of representing a relationship between a pair of vertices, represent a relationship between a set of vertices. If the relationship holds for every subset of the hyperedge, the hypergraph is called a *simplicial complex*. Although hypergraphs are more general, if the problem or the data has a special structure then simplicial complexes are more appropriate. For the *group recurrence* problem, we need to predict recurrence of not just observed groups but also the subgroups. Thus, simplicial complexes are more applicable to our problem

For the *group recurrence* problem we also want our model to capture any prior knowledge associated with each group or subgroup that might indicate cohesion among group members, or the context associated with the group. We use the concept of a *weighted simplicial complex*, which is a simplicial complex where each simplex has a prior weight associated with it. We develop several schemes to generate these prior weights, modeling different prior knowledge scenarios.

We observe that a simplicial complex, from a frequent pattern mining perspective (Aggarwal & Han, 2014), is the trivial set of all the frequent patterns of frequency equal to one, mined from the transactions database of hyperedges. This motivates the use of a Hasse diagram (Figure **??**) (Skiena, 1990) (similar to enumeration trees in pattern mining) as a graph representation for the simplicial complex. If we associate a weight with each node (representing simplicies) of the Hasse diagram it represents a weighted simplicial complex. We hypothesize that the topology of these groups plays a critical role in how past occurrences influence future occurrences of other (sub)groups. Using the Hasse diagram, we apply a modification of the HyperPrior algorithm (Tian et al., 2009), for generating label diffusion-based machine learning models, as well as develop hierarchical label spreading algorithms for recurrence prediction. These algorithms make use of the weighted simplicial complex topology while exchanging the occurrence information between the subgroup nodes in the Hasse diagram. Our ex-

perimental analysis, conducted using the DBLP and EverQuest II datasets, shows the efficacy of the techniques developed. The main contributions of this study are:

- We present machine learning models to predict recurrence of already observed groups, which takes into account the higher order topology.
- We present a Hasse diagram-based framework to study simplicial complexes, hypergraphs, and frequent pattern mining in a unified manner.
- We show that frequent patterns can be considered as topological entities, with relationships between them guided by higher-order topological properties. To the best of our knowledge this has not been done before.

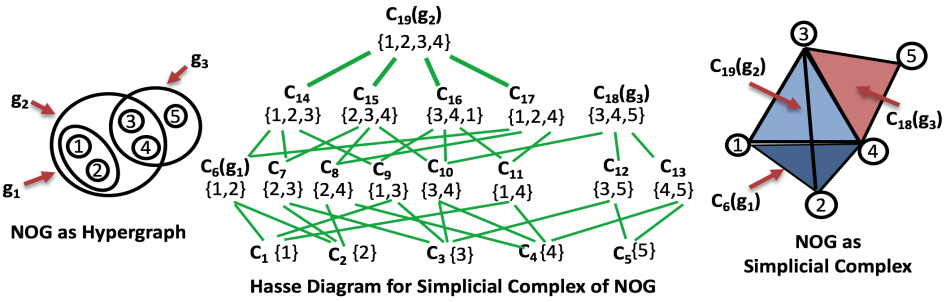

Figure 3.1: Example illustrating a network of groups hypergraph (left) as a simplicial complex (right) and as a Hasse diagram (middle) corresponding to the simplicial complex, for a scenario where the actors {1,2,3,4,5} have collaberated in the past as groups: $g_1 = \{1, 2\}$, $g_2 = \{1, 2, 3, 4\}$ and $g_3 = \{3, 4, 5\}$.

3.2 Problem Statement and Preliminaries

3.2.1 Models for Network of Groups

Although we had defined models in detain in Chapter 2, we again provide preliminaries for convenient reading, and also, the notations employed in this chapter might be more simplified for the discussion within this chapter. We have a set of n *actors* $V = \{v_1, v_2, ..., v_n\}$. A subset of these *actors* can form a *group*. We have a collection

of m such groups observed in the past, denoted by $G = \{g_1, g_2, ..., g_m\}$ where $g_i \subseteq V$ represents the i^{th} *group*. The cardinality $c_i = |g_i|$ of a group is the number of actors in it. We let $R(g)$ denote the number of times group $g \in G$ has occurred. The network of groups can be modeled as a hypergraph (Berge & Minieka, 1973) $H = (V, G)$ where the observed groups G are the hyperedges over the vertex set V of actors. We denote by $S_i = \{s_k^i, \forall k \in \{1, 2, ..., 2^{|g_i|} - 2\}\}$ the set of all proper subsets of each group $g_i \in G$. If we consider the union of all subsets of the sets in G along with G itself, i.e., $C = \{G \cup (\bigcup_{i=1}^{m} S_i)\}$, then we have a (abstract) simplicial complex C and each element $c \in C$ is a simplex which represents a group or subgroup. If we also associate a weight $W(c) \in \mathbb{R}, \forall c \in C$, then we attain a *weighted simplicial complex* $\diamondsuit = (C, W)$. For convenience we also define the set containing the subgroups in C that were never observed in the past, i.e., $C_s = \{c | (c \in C) \wedge (c \notin G)\} = (C - G)$. Each $c \in C_s$ also has a set of groups $Q(c) \subseteq G$, of which it is a subgroup of, i.e., $Q(c) = \{x | (x \in G) \wedge (c \subset x)\}$. We define an occurrence function O which gives the occurrence count to all the groups in C as follows:

$$O(c) = \begin{cases} R(c) + \left(\sum_{x \in Q(c)} R(x) \right) & \text{when } c \in G \\ \sum_{x \in Q(c)} R(x) & \text{when } c \in C_s \end{cases} \tag{3.1}$$

In words, for an observed group we simply take the number of times it has occurred, $R(c)$, and also add the counts of the groups it has been a subset of. In the case of subgroups (those groups that haven't occurred in the past) we simply add the counts of the groups it has been a subset of. For a simplex (or (sub)group) $\alpha \in C$ we define its dimension as $dim(\alpha) = |\alpha| - 1$. If K_{max} is the maximum cardinality of any simplex in C then $(K_{max} - 1)$ is the maximum dimension of any simplex in C or simply the dimension of C.

The set of simplices of cardinality k within the simplicial complex C are defined by the set: $\pi^{\mathbf{k}} = \{\sigma | \sigma \in C \wedge |\sigma| = k\}, \forall k \in \{1, ..., K_{max}\}$. For the example in Figure 3.1, $C = \{C_1, ..., C_{19}\}$, $G = \{g_1, g_2, g_3\} = \{C_6, C_{19}, C_{18}\}$ and $C_s = (C - G)$.

We also define a *Hasse diagram*, N_h, for the simplicial complex C. The level in

the diagram (Figure 3.1) determines the poset relation. We use the undirected graph derived from the Hasse diagram N_h over the vertex set $V(N_h) = C$ and with a set of undirected edges $E(N_h) = \{(x, y) \cup (y, x) | (x, y \in V(N_h)) \wedge (y \subset x) \wedge (|y| = |x| - 1)\}$. In the case of a weighted simplicial complex $\diamondsuit = (C, W)$, we associate with each vertex the weight of the corresponding simplex it represents, i.e., $W(v), \forall v \in V(N_h)$. Note, we can also associate a weight with the edges but in this study we assume all edges have a unit weight. We denote $\mathbf{A_h}$ to be the adjacency matrix of size $(|C| \times |C|)$ associated with the Hasse diagram graph N_h. This process of representing a hypergraph as a Hasse diagram is also referred to as the *Hasse expansion*, which we had discussed in detail in §2.2.4, Def. 48.

3.2.2 Problem Statement

We are interested in prediction of groups formed by two processes: group recurrence and subgroup recurrence. In group recurrence, a group $g_i \in G$, called a *recurring group*, observed in the past can again occur in the future. Our first problem is to predict a score for each of the groups in G. This score reflects the possibility of the given group occurring again in the future. In subgroup recurrence, a group $c_i \in C_s$ which has never been observed as a group in the past, might occur in the future. We refer to such groups as *recurring subgroups*. Our second problem is to predict a score for each of the groups in C_s, which reflects its possibility to be formed in future. We restrict ourselves to the prediction of only the recurring groups and subgroups and not groups composed of entirely new actors.

3.3 Methods

In this section, we first enumerate several ways of assigning prior weights. We then describe three different methods (along with several variants) to solve the problems described in the previous section. Each method models the tendency of a given group to be formed in the near future by assigning a score $\mathbf{S}(c)$ to each group in $c \in C$,

returning a final vector of scores **S**. The first method uses a simple group count-based approach and the next two methods consider the hierarchical structure of the higher order topology within the Hasse diagram.

3.3.1 Schemes for assigning initial weights

Several studies on small groups have shown that social actors tend to collaborate with actors with whom they have already developed strong working relationships (Lungeanu et al., 2014) and that repeated ties within a group positively affect its performance (Contractor, 2013). There are a number of ways to assign a prior weight to represent the strength of the relationships between group members. Kapoor et al. (Kapoor et al., 2013) defined several weights for the problem of node centrality, of which we utilize two. The first, shown in (3.2), corresponds to a frequency-based definition and simply counts the number of times a group has performed some task together. The second, shown in (3.3), enforces that the average attachment of any two individuals (or the attention span of a member towards each other member) in a group decreases in proportion to the size of the group.

$$\mathbf{W}(c) = O(c), \forall c \in C \tag{3.2}$$

$$\mathbf{W}(c) = \frac{\log(O(c)) + 1}{|c|}, \forall c \in C \tag{3.3}$$

The weights in (3.2) and (3.3) initialize all groups (observed) as well as subgroups (unobserved), i.e., all the simplices. We, therefore, also design slightly different variants where we only initialize the observed groups, which emphasizes the hypergraph model of the network:

$$\mathbf{W}(c) = \begin{cases} O(c) & \text{if } c \in G \\ 0 & \text{if } c \in C_s \end{cases} \tag{3.4}$$

$$\mathbf{W}(c) = \begin{cases} \frac{\log(O(c)) + 1}{|c|} & \text{if } c \in G \\ 0 & \text{if } c \in C_s \end{cases} \tag{3.5}$$

In the following sections we will define several algorithms which will use these four initialization schemes. We will use the suffixes: **Simp-C**, **Simp-W**, **Hyp-C**, and **Hyp-W** to refer to the initializations in (3.2)-(3.5), respectively.

3.3.2 Count Based Scores (CBS)

We build the first set of scores using only the occurrence information available. For this we simply take the score vector **S** as the weight defined in (3.2) and (3.3), denoted the **CBS-C** score and **CBS-W** score, respectively. The **CBS-C** score, gives each group a value which is determined by the number of times the group members have worked together in past. Whereas, **CBS-W** assigns score based upon the cohesion among the group members.

3.3.3 Hasse Diagram based Models

CBS scores utilize counts of group recurrences, wherein each group was considered in isolation but do not consider the network of groups. This network encodes information about the observed groups, the unobserved groups, and the topological relations between them. Occurrences of a group affect the probability of other groups in the network to collaborate in the future. We develop two approaches applied to a Hasse diagram representation of a weighted simplicial complex to capture the local and global relational information.

Algorithm 1 GetHDSScores $(N_h, \mathbf{y}, K_{max}, \alpha)$

$\mathbf{f} \leftarrow \mathbf{y}$, $C \leftarrow V(N_h)$ #
Get the simplicial complex corresponding to the Hasse diagram
for $k = K_{max} - 1$ to 1 **do**
  for all $c \in \pi^{\mathbf{k}}$ **do**

$$\mathbf{f}(c) \leftarrow \mathbf{f}(c) + \alpha \left(\sum_{x \in (Q(c) \cap \pi^{\mathbf{k+1}})} y(x) \right)$$

  end for
  return f
end for

3.3.4 Hasse diagram spread-based scores (HDS Scores)

This class of methods is based upon the intuition that observed groups in the Hasse diagram influence the subgroups below it in the hierarchy. Influence spread can happen in a variety of ways. There are several possible counter-intuitive group phenomena.

We model these in a holistic fashion by spreading scores over the Hasse diagram. We propose that if we observe a node g_i in the Hasse diagram then it spreads its score (s_i) down the hierarchy. It can send the same, more, or less of its score to its children. In general, it can send αs_i ($\alpha \geq 0$) score to its children. These children update their scores and spread the score down the hierarchy recursively. This is shown in Algorithm 1. We initialize the algorithm using the vectors ($\mathbf{y} = \mathbf{W}$) in equations (3.2)-(3.5) to get four different scores, $\mathbf{S} = GetHDSScores(T, \mathbf{y}, K_{max}, \alpha)$, which we denote as **HDSSimp-C**, **HDSSimp-W**, **HDSHyp-C** and **HDSHyp-W**, respectively.

3.3.5 Hasse diagram diffusion-based scores:

The spread-based scores are local in the sense that the final score of a node is only determined by its initial score and the scores of its parent(s). But, in general, the nodes representing groups in the network are connected by many pathways. Therefore, it is reasonable to assume that a potential group may be affected by occurrence of non-parent groups in the network. In order to take into account this structure of the entire Hasse diagram, we apply a modification of the graph label propagation algorithm HyperPrior (Tian et al., 2009).

Each vertex (group) is initialized with a label, which encodes prior information about the recurring tendency of that node. These labels (information) then diffuse (exchange information) via random-walks through the Hasse diagram network structure. After the random-walks stabilize, the final label for each vertex is the score indicating its recurrence possibility. The final label at a given vertex represents the chances that a random walk originating from other nodes ends at this vertex. Hence, this score is a combination of both the group's initial tendency to occur plus an adjustment based on the knowledge from other groups in the network, i.e., the random walk outcomes. This adjustment models a network guided similarity between the vertex and the other nodes. Vertices that are near in the network should end up receiving similar labels/scores.

More formally, let $\mathbf{y}$ be the vector of initial labels for the vertices in the Hasse

diagram N_h with incidence matrix $\mathbf{A_h}$. Vector $\mathbf{y}$ is initialized by any of the weights in (3.2)-(3.5). As in a graph-based learning task, we learn the final label (score) vector $\mathbf{f}$ by taking into account the competing aims of similar labels for vertices connected by an edge in the Hasse diagram and of similar labels between the initial and final vectors. We capture these competing aims in the following cost minimization objective:

$$\min_{\mathbf{f}} \mathbf{f}^T \mathbf{L_h} \mathbf{f} + \beta \|\mathbf{f} - \mathbf{y}\|^2 \tag{3.6}$$

where, $\mathbf{L_h} = \mathbf{I} - \mathbf{D_v^h}^{-\mathbf{1/2}} \mathbf{A_h} \mathbf{D_v^h}^{-\mathbf{1/2}}$ is the normalized Hasse Laplacian (see (2.66)) and $\mathbf{D_v^h}$ is a diagonal matrix consisting of the vertex degrees (see (2.67)). The first term in (3.6) is a smoothing term which ensures that vertices (groups) sharing an edge (having common group members) have similar scores. This term therefore, enforces the Hasse diagram structure while learning the labels. The second term measures the difference between the given initial labels and the final vertex scores. It can be shown (Zhou et al., 2004) that the solution to (3.6) is equivalent to the solution of the following linear system:

$$\mathbf{f}^* = (1 - \mu)(\mathbf{I} - \mu\theta)^{-1}\mathbf{y}, \tag{3.7}$$

where $\mu = 1/(1 + \beta)$, $\theta = \mathbf{D_v^h}^{-\mathbf{1/2}} \mathbf{A_h} \mathbf{D_v^h}^{-\mathbf{1/2}}$, and $\mathbf{f}^*$ is the vector of final labels of the group nodes. Note that, $\mathbf{f}^*(c)$ is the aggregate tendency $\mathbf{S}(c)$ of a group $c \in C$ to reoccur. Therefore, we have: $\mathbf{S} = \mathbf{f}^*$.

Similar to spread-based scores, we denote the scores here by the following notations: **HDDSimp-C**, **HDDSimp-W**, **HDDHyp-C**, and **HDDHyp-W** when intialized using (3.2)-(3.5). Our aim is to predict a score for the recurring groups (i.e., $g \in G$) and recurring subgroups (i.e., $c \in C_s$). For each of the methods above, we get a final vector that contains the scores for all the groups. We partition the vector $\mathbf{S}$ into two vectors $\mathbf{S_{rg}}$ and $\mathbf{S_{rs}}$ of sizes $|G|$ and $|C_s|$, respectively, such that $\mathbf{S_{rg}}(c) = \mathbf{S}(c), \forall c \in G$ and $\mathbf{S_{rs}}(c) = \mathbf{S}(c), \forall c \in C_s$. In summary, we obtain three score vectors $\mathbf{S_{rs}}$, $\mathbf{S_{rg}}$ and $\mathbf{S}$ for

each of the above methods.

3.4 Experimental Analysis

3.4.1 Dataset and Statistics

Datasets: The first dataset we apply our methods to is a massive multiplayer online role-playing game (MMORPG) dataset obtained from the Sony's EverQuest II (EQ II) game (www.everquest2.com). The game provides an online environment where multiple players can log in and collaborate in groups to perform various quests and missions. The server logs from this game, provided by Sony, were used to extract group interactions. Here, we treat a set of players performing a task or mission as a group in the EQ II network. The EQ II data contains logs for 21 weeks of data for training and testing. We divide them into seven training/testing splits, each of which has a two-week long training period followed by a one-week testing period.

The second dataset is the DBLP dataset (obtained from www.aminer.org) containing computer science publications from 1930-2015. The set of co-authors on a paper form a group in the DBLP network. Note that in both EQII and DBLP networks, the groups can perform multiple game tasks or co-author multiple papers. We make eleven train-test splits as follows: $(1992-95/96-98)$, $(1993-95/96-98)$, $(1993-95/96-99)$, $(1991-97/98-10)$, $(1997-00/01-03)$, $(1998-00/01-03)$, $(1998-00/01-04)$, $(2002-05/06-08)$, $(2003-05/06-08)$, $(2003-05/06-09)$ and $(2001-07/08-10)$; following the format: *(train period start year−train period end year/ test period start year−test period end year)*. These splits were designed to observe the effect of varying training and testing period lengths as well as varying the entire train/test evaluation period. We have evaluated other variations of period lengths and other decades in the DBLP data, but in this chapter we limit our discussion to the train/test periods we just described.

Statistics: Recall that we have two kinds of groups: (1) recurring groups that are observed in training and observed again in testing and (2) recurring subgroups that are

Table 3.1: Recurrence Statistics of the various Train/Test Periods

Dataset	Training Actors	Testing Actors	% Old Actors in Testing	% New Actors in Testing	Training Groups	Testing Groups	% Recurring Groups in Testing	% New Groups in Testing	% Groups with Old Actors	% Groups with New Actors
EQ II	3051	2215	81.67	18.33	1775	1219	67.92	32.08	88.93	11.07
Avg.			**84.06**	**15.94**			**74.01**	**25.99**	**90.51**	9.49
DBLP	677K	640K	40	60	549K	433K	12.06	87.94	84.53	15.47
Avg.			**34.73**	**65.27**			**11.65**	**88.35**	**81.17**	18.83

Table 3.2: Different Dimension Face Recurrence Statistics

	% Train Groups	% Test Groups	% Train Groups	% Test Groups	% Train Groups	% Test Groups	% Train Groups	% Test Groups
	EQ II Splits				**DBLP Splits**			
Simplex Dimension	RG+RS (Exact)		RG+RS (New Vertices)		RG+RS (Exact)		RG+RS (New Vertices)	
≥ 1	15.57	**21.25**	15.70	**21.43**	4.63	**3.09**	6.28	**4.20**
≥ 2	8.97	**12.72**	9.02	**12.80**	2.78	**1.79**	3.41	**2.19**
	RS (Exact)		RS (New Vertices)		RS (Exact)		RS (New Vertices)	
≥ 1	0.70	**0.71**	0.76	**0.77**	1.70	**0.89**	3.08	**1.61**
≥ 2	0.20	**0.23**	0.25	**0.29**	0.76	**0.38**	1.24	**0.63**
	RG (Exact)		RG (New Vertices)		RG (Exact)		RG (New Vertices)	
≥ 1	57.18	**20.55**	57.50	**20.66**	14.89	**2.21**	17.53	**2.60**
≥ 2	45.68	**12.49**	45.77	**12.51**	10.02	**1.41**	11.17	**1.57**

observed in testing but are only observed as a subgroup of some group that occurred in training. We shall refer to the former set as RG, the latter set as RS, and the combined set as (RG+RS). Table 3.1 contains several statistics for (RG+RS). However, due to space constraints, we only show statistics for the last split from each dataset, as well as the average statistics across the splits. In Table 3.1, an actor in the testing phase is considered "**old**" if it was observed in the training period, otherwise it is considered "**new**". Note that for any group with new actors in the testing phase, we can only test whether the subgroup with old actors is a recurring group or subgroup from the training period. These statistics are based on the distinct groups from the testing and training periods, so as to avoid any bias from the multiplicity of certain group interactions. We observe that on an average around 90% of the EQ II network groups and around 81% of the DBLP network groups formed in the test period contain at least one old actor. Only within these groups can we possibly search for recurring groups or subgroups. Note, 74% of the EQ II groups and around 12% of DBLP groups in testing period are exact recurrences and included in the set RG. This demonstrates that the recurring group process is more common in the EQ II network, whereas the recurring subgroup process is the more common feature in the DBLP network.

In Table 3.2, we record the statistics of the groups in training that recur in testing and of the groups in testing that are recurring groups or subgroups. We only consider groups of size ≤ 6 (i.e., faces of dimension ≤ 5) and also omit vertex recurrences since those are reported in Table 3.1.

For dimensions ≥ 1, the set RG+RS accounts for 20% of the testing groups in the EQ II network and $3-4\%$ in the DBLP network. For dimensions ≥ 2, the set RG+RS accounts for approximately 12% of the testing groups in the EQ II network and only 2% in the DBLP network. These subtle observations indicate that GR and SR processes are responsible for a significant portion of future formed groups. Therefore, modeling these processes is an important step towards higher order link prediction.

3.4.2 Evaluation Methodology and Experimental Setup

We evaluate the performance of these methods as classifiers using the area under the curve (AUC) statistic of the receiver operating characteristics (ROC) (Van Trees, 1968). Using the three score vectors as the model output we calculated AUC scores for two sets of prediction test scenarios. The first set includes the exact occurrences found in the testing period (referred to as **"(Exact)"**) and the other set includes occurrences found with new vertices in the testing period (referred to as **"(New Vertices)"**). The following six scenarios are considered for each set:

1. **RG+RS(v)**: Predicting both recurring groups and subgroups that are dyadic edges or other higher order faces. Note that for any group with new actors in the testing phase, we can only test whether the subgroup with old actors is a recurring group or subgroup from the training period.

2. **RG+RS(v+e)**: Predicting both recurring groups and subgroups that are only triangles or other higher order faces. We only consider groups of size ≤ 6 and also omit vertex recurrences since those are reported in Table 3.1.

3. **RS(v)**: Predicting only recurring subgroups that are edges or other higher order faces.

4. **RS(v+e)**: Predicting only recurring subgroups that are triangles or other higher order faces.

5. **RG(v)**: Predicting only recurring groups that are edges or other higher order faces.

6. **RG(v+e)**: Predicting only recurring groups that are triangles or other higher order faces.

The optimal parameters were chosen for each split separately via grid search on the following parameter space: $\alpha = \{0.01, 0.1, 0.3,\ 0.5, 0.7, 0.9, 1, 2, 5, 10, 20\}$ and $\mu =$

Table 3.3: AUC Scores for **EQ II** and **DBLP**

Method	EQ II Exact RG+RS (v)	RG+RS (v+e)	RS (v)	RS (v+e)	RG (v)	RG (v+e)	New Vertices RG+RS (v)	RG+RS (v+e)	RS (v)	RS (v+e)	RG (v)	RG (v+e)
HDDHyp-W	**0.96**	**0.98**	**0.67**	**0.87**	0.79	0.83	**0.96**	0.97	**0.68**	**0.86**	0.79	0.83
HDDHyp-C	**0.96**	**0.98**	0.63	0.78	**0.83**	**0.87**	**0.96**	**0.98**	0.64	0.78	0.82	**0.86**
HDDSimp-W	0.83	0.81	0.63	0.67	0.78	0.81	0.83	0.81	0.62	0.64	0.78	0.81
HDDSimp-C	0.78	0.75	0.52	0.6	0.83	0.86	0.78	0.75	0.52	0.59	**0.83**	0.85
CBS-W	0.77	0.68	0.6	0.54	0.78	0.81	0.76	0.68	0.59	0.5	0.78	0.81
CBS-C	0.76	0.72	0.5	0.49	0.82	0.85	0.76	0.72	0.5	0.47	0.82	0.85
HDSHyp-W	**0.96**	0.97	0.65	0.71	0.79	0.82	**0.96**	0.97	0.65	0.7	0.79	0.82
HDSSimp-W	0.7	0.59	0.58	0.52	0.76	0.8	0.7	0.59	0.58	0.48	0.76	0.8
HDSHyp-C	0.95	0.97	0.58	0.63	**0.83**	0.86	0.95	0.97	0.59	0.63	0.82	**0.86**
HDSSimp-C	0.68	0.61	0.49	0.48	0.82	0.85	0.67	0.6	0.48	0.44	0.82	0.85
	DBLP											
HDDHyp-W	**0.9**	**0.89**	**0.8**	**0.78**	0.69	0.68	**0.82**	**0.85**	0.73	**0.72**	0.7	0.68
HDDHyp-C	0.89	**0.89**	0.79	0.78	0.69	0.68	**0.82**	**0.85**	0.73	**0.72**	0.69	0.68
HDDSimp-W	0.77	0.79	0.73	0.73	0.7	0.69	0.77	0.77	**0.74**	0.71	0.71	0.69
HDDSimp-C	0.75	0.76	0.71	0.72	**0.71**	**0.7**	0.74	0.74	0.73	0.7	**0.72**	**0.7**
CBS-W	0.67	0.64	0.65	0.61	0.69	0.66	0.69	0.64	0.7	0.63	0.7	0.66
CBS-C	0.65	0.63	0.59	0.58	0.64	0.62	0.65	0.62	0.63	0.59	0.65	0.62
HDSHyp-W	0.89	0.88	0.75	0.73	0.69	0.66	**0.82**	0.84	0.73	0.7	0.7	0.66
HDSSimp-W	0.59	0.53	0.6	0.54	0.69	0.66	0.63	0.55	0.67	0.57	0.7	0.66
HDSHyp-C	0.88	0.87	0.72	0.71	0.64	0.62	0.8	0.83	0.68	0.67	0.65	0.62
HDSSimp-C	0.49	0.43	0.5	0.47	0.64	0.61	0.54	0.46	0.57	0.51	0.65	0.62

$\{10^{-7}, 10^{-6}, 10^{-5},\ 10^{-4}, 10^{-3},\ 0.01, 0.1, 0.3, 0.5, 0.7, 0.9, 0.99\}$. All the Hasse diagrams considered in the above methods have un-weighted edges.

3.4.3 Results and Discussion

We compare the twelve different AUC scores, described in the prior section, for the ten methods developed in this paper. Results are reported in Table 3.3 for the EQ II and DBLP data. We have three different kinds of scores: CBS (Section 3.3.2), HDS (Section 3.3.4) and HDD (Section 3.3.5). Both the **CBS-W** and **CBS-C** scores are only count based and don't take into account any topological relationship between groups. On the other hand, the HDS and HDD methods take into account topology by exchanging information locally and globally, respectively. One of our main hypotheses is that topological structure affects the group recurrence behavior. We are also unaware of any methods for small group recurrence and therefore chose **CBS-W** and **CBS-C** scores as our baseline. Note, as described in Section 3.3.1, all the three genre of methods can be either count based (referred using suffix **-C**), or cohesion metric based (denoted

by suffix **-W**). The count based variants do not take into account the cardinality of the (sub)groups whereas the cohesion metrics are cardinality based.

Effect of Topology: We observe from Table 3.3 (the best scores are highlighted in bold) that the Hasse diagram-based methods consistently outperform the count-based methods. This supports our hypothesis that Hasse diagram-based methods, which take into account topology, indeed, are more informative about the group recurrence process.

We also compare the methods against four criteria: (a) How do the prediction methods fare for recurring subgroups as compared to recurring groups?; (b) How well do the methods predict at dimensions of dyadic edges and above, i.e., the -(v) cases, compared with how well they predict at dimensions of triadic groups and above, i.e., the -(v+e) cases?; (c) How well do the methods predict the "Exact" occurrences versus the "New Vertices" occurrences?; and (d) How do the count-based **"-C"** methods compare with the cohesion-based **"-W"** methods?

We observe that in order to predict recurring subgroups, **HDDHyp-W** outperforms all other methods whether the subgroup was an "exact" occurrence or a "new vertices" occurrence in testing. This suggests that exchange of information from the groups observed in the past to the groups not observed in the past via the Hasse diagram topology and the global-based label diffusion process is more crucial for influencing the appearance of subgroups not observed in the past. In fact, the poor accuracy of the **HDDSimp** methods indicates that weights placed on (possibly unobserved) subgroups of observed groups used as prior information cause bias and hurt the predictive power of the model. Given that **HDDHyp-W** is initialized using the cohesion weights in (4), the normalization of counts only on the prior observed group occurrences in the diagram is important for recurring subgroup prediction. Moreover, the performance of predicting triangles or higher order groups (RS(v+e)) is higher for the EQ II data and comparable for the DBLP data to that of predicting dyadic edges or higher (RS(v)) across all HDD methods, implying the important role played by the Hasse diagram structure for higher order group prediction.

On the other hand for recurring group prediction the count-based methods, which

inclu **HDDHyp-C** and **HDDSimp-C** performed best, suggesting that the likelihood of recurrence of already-observed groups is determined more by the simple counts of past concurrences. The count-based HDS methods also give results comparable with that of the HDD methods. This implies that even the local spread of count information is sufficient for recurring group predictions. These **-Simp**-based methods using (1), which take into account the subgroup counts of the groups that occurred in training, provide good results, suggesting that the unobserved subgroups have an important influence on the potential of groups to re-occur.

Finally, we note that across both the datasets and across all the twelve experiments, the **HDD** methods generally perform better than or as good as **HDS** methods. Further results and details shall be made available in a future technical report.

3.5 Conclusions

We consider the problem of predicting small group evolution and focus on the sub-problem on group and subgroup recurrence. We highlight two important group recurrence processes and capture them using weighted simplicial complexes. We use a Hasse diagram corresponding to the simplicial complex as a graph whose nodes correspond to subgroups in the complex. We then build semi-supervised models on top of this graph for group recurrence prediction. We have shown that frequent patterns like small groups can be considered as topological entities, with relationships between them guided by higher order topological properties.

Chapter 4

Group Accretion Prediction: A Cross-sectional Analysis

This chapter revisits the problem of group evolution. But unlike the previous chapter, this chapter addresses the problem of *new hyperedge prediction* by dividing it into two sub-problems of group accretion prediction and subgroup accretion prediction.

We start by again motivating that Small Group evolution has been of central importance in social sciences and also in the industry for understanding dynamics of team formation. While most of research works studying groups deal at a macro level with evolution of arbitrary size communities, in this chapter we restrict ourselves to studying evolution of small group (size ≤ 20) which is governed by contrasting sociological phenomenon. Given a previous history of group collaboration between a set of actors, we address the problem of predicting likely future group collaborations. Unfortunately, predicting groups requires choosing from $\binom{n}{r}$ possibilities (where r is group size and n is total number of actors), which becomes computationally intractable as group size increases. However, our statistical analysis of a real world dataset has shown that two processes: an external actor joining an existing group (*incremental accretion (IA)*) or collaborating with a subset of actors of an exiting group (*subgroup accretion (SA)*), are largely responsible for future group formation. This helps to drastically reduce the $\binom{n}{r}$

possibilities. We therefore, model the attachment of a group for different actors outside this group. In this chapter, we have built three topology based prediction models to study these phenomena. The performance of these models is evaluated using extensive experiments over DBLP dataset. Our prediction results shows that the proposed models are significantly useful for future group predictions both for IA and SA.

The rest of the chapter is as follows. Next Section 4.1 introduces and motivates the problem, section 4.2 is for the related work, section 4.3 describes the problem statement, section 4.4 describes the topology based methods, in section 4.5 the experiments conducted are described and results are discussed, followed by conclusion in section 4.6.

4.1 Introduction

Study of small groups has been an important endeavor in a large number of disciplines like psychology, sociology, communication and information science for past 50 years (Poole et al., 2004). Advent of globalization has lead to changing nature of groups or teams in industry leading to increasing interest in studying their dynamics (Cheney et al., 2010). Large part of the research in the field of Organization Science is dedicated to study effective ways to build teams by combining employee expertise (Boh et al., 2007). With the rising number of large interdisciplinary scientific teams (Wagner et al., 2011), understanding drivers affecting their success is of key importance for science funding agencies while selecting team of scientists (Lungeanu et al., 2014). Other real life applications include building emergency response teams for natural disasters management, automation of team selection for military operations and self-organizing open-software teams (Hahn et al., 2008). While such studies are important, the ever increasing availability of online "group" interaction data for example, social networking sites like Facebook or Twitter, group communication tools like Skype, Google Hangout, Google Docs, Massive Online multi-player games (MMOGs) such as World of Warcraft, etc., makes such studies even more realistic. In scientific research, such dataset have been used to study group dynamics for benefit of both industry and academia (Contrac-

tor, 2013). Similarly, user labeled groups of research artifacts such as research datasets have been studied in detail (Singhal et al., 2014a).

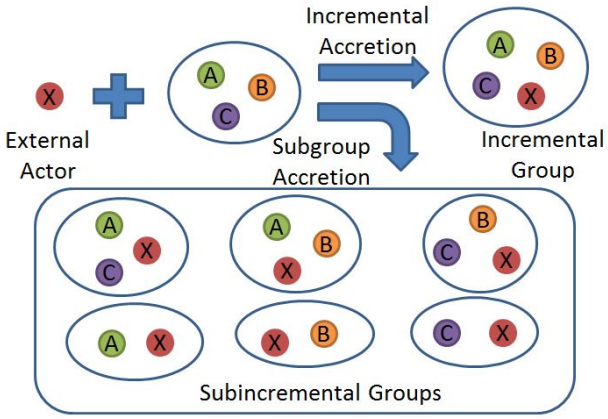

Figure 4.1: Example illustrating *incremental accretion* and *subgroup accretion* processes. Set {A,B,C} is a group and X is an external actor.

Several studies in the past have analyzed the usefulness of various features used for membership prediction in communities (Kang et al., 2013; Patil et al., 2012; 2013a; Sharara et al., 2012). Also, some works have focused on group membership dynamics by simulating how individual join or leave a group (alv, 2011; Ahmad et al., 2011; Johnson et al., 2009), while such studies have not developed prediction models. Objective of our work is to focus on the task of actual future group prediction in contrast to feature analysis or simulation based studies. Moreover, most past works on group evolution in social networks primarily deal with evolution of arbitrary size communities or groups (Chen et al., 2008; Patil et al., 2013b; Sharara et al., 2009). These sizes are usually large and the boundaries of community depend on the definition of membership employed (Spiliopoulou, 2011). In this work we are interested in well defined small groups (size $\leq$ 20) like research collaborations or teams (Beebe & Masterson, 2009) also called as *Bona fide groups* (Putnam & Stohl, 2012) in sociology. Also these small groups are self assembling (Contractor, 2013) where members leave or join groups autonomously and the motivation or theories for group formation is much different from the large communities (Poole et al., 2004). Moreover, these groups are connected by

the social network of actors, resulting in group ties or *network of groups* (Monge & Contractor, 2003) (see Figure 4.2), which plays central role in group formation process. We focus on group accretion which is a subset of the group evolution. Group accretion is the process of size increment in groups by addition of more members. More specifically we define two subproblems: *incremental accretion* and *subgroup accretion*. In the first problem, given a group of size=x, we predict the likelihood of one more member being absorbed in it, to yield a size=$(x+1)$ incremental group (Figure 4.1). The subgroup accretion is the problem of incremental accretion on all the (2^x-2) subgroups of a given group to yield prediction scores of (2^x-2) new incremental groups (Figure 4.1). Intuition behind choice of these problems is that, given a past history of group collaborations, a large percentage of groups in future are formed through these two processes. Our aim finally is to build models that predict future groups that are likely to be formed using these two mechanisms. This work therefore, is an initial step towards a more general higher order group prediction problem.

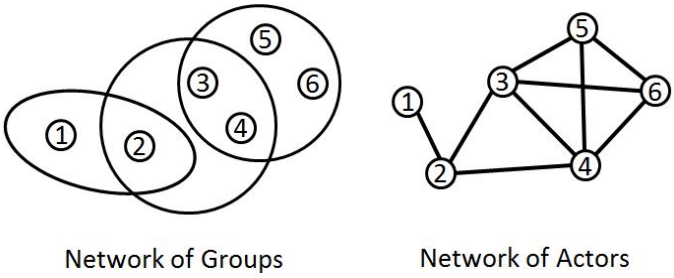

Figure 4.2: Example illustrating *network of groups* (left) and the corresponding *network of actors* (right) where {1,2,3,4,5,6} are the actors

In this chapter we assume that the groups are not isolated but rather interact with each other through *network of groups* and group members make individual decisions to collaborate or not (*self assembly*). We therefore, model the attachment of a group (*group*) to an actor (*node*) outside it. Topology based dyadic link prediction (DLP) methods have proven successful in capturing *node to node* attachment. Guided by both, DLP methods and sociology theories of small groups we have proposed three different methods. First, method is an extension of the popular path enumerating Katz

method (Katz, 1953). Second, is a network alignment based supervised method. With a philosophy similar to Katz, it captures the inter-group communication cycles, which are theoretically hypothesized more appropriate, rather than paths. Lastly, we also propose a label propagation based method where the random walks are guided by the network of groups (a hypergraph (Berge & Minieka, 1973; Sharma et al., 2014)). Based on the extensive set of experiments it turns out our models are able to effectively predict future groups formed by both IA and SA processes.

Now we summarize the key points of this research paper:

- We highlight an important observation that a large chunk of future groups are formed by two different accretion processes from the past groups.
- We have focused on the evolution of small groups and defined new problems for small group accretion.
- We have developed three topology based methods, guided by social theories, to address these problems in a novel manner.
- We have therefore, presented an overall new incremental approach to address the less addressed and intriguing problem of higher order link prediction.

4.2 Related Works

Social Group Evolution There is a vast body of literature regarding community evolution (Spiliopoulou, 2011) and more general network evolution (Aggarwal & Subbian, 2014) in social networks. Definition of community in all these works varies drastically producing from small groups to size as large as hundreds or thousands. Therefore, they take a macroscopic view while answering the questions of how to detect communities and how they evolve over time. Though, recently there have been some works which zoom in and try to understand the evolution from the perspective of individual actors and their relationship with other actor group.

Among them the first category of works focus on building models which can simulate the group formation tasks like leaving, joining or switching between groups (alv, 2011; Ahmad et al., 2011; Johnson et al., 2009). Alvari et. al. (Alvari et al., 2011) provide a game theoretic community/group detection model where the actors in the social network are rational agents performing these tasks while maximizing their utility. Same authors extend their work for evolutionary setting (Alvari et al.) and apply to MMOGs (alv, 2011). MMOG guild formation is also studied as stochastic processes in both network (Ahmad et al., 2011) and network-less settings (Johnson et al., 2009). Our work, rather than simulation, focuses on predicting the exact groups that might occur in future.

Second category deals with analysis of various characteristics of groups (like diversity, cohesion, stability, type, etc.) and their correlation with different factors (size, member properties, etc.) (Chen et al., 2008; Chung et al., 2014).

Third category of papers are devoted to extracting features of different kinds like network, actor, group or communication content and pose the problem of group membership prediction as a classification task. For instance, Patil et al. have done feature extraction for group attrition (Patil et al., 2013b), group stability (Patil et al., 2013a) as well as group destruction (Patil et al., 2012). It differs from our work as they focus on understanding the importance of various features rather than prediction of groups. In a series of papers from Sharara et al. have stressed on the idea of loyalty (affinity) of actor towards different groups and its longitudinal changes (Sharara et al., 2009). This idea has been extended for deducing important actors by using group semantics (like diversity) (Sharara et al., 2012) and applied to analysis of guilds in MOMGs (Kang et al., 2013). Both, Patil et al. and Sharara et al. model actor's attachment or loyalty for different groups whereas we model the opposite: tendency of a group to absorb different actors. Moreover, both of them primarily deal with large communities like conferences in DBLP data. We are specifically focused on predicting small cohesive groups or teams (like a small group of researcher working on a publication in DBLP) where different social phenomenon are at play. We rather treat the problem as extension of DLP to

higher order (group) prediction.

Social Sciences Naturally occurring small groups are called *Bona Fide Groups* (Putnam & Stohl, 2012), which are the focus of this paper. A good reference for small group theories can be found in Poole et al. (2004). Plethora of research deals with application of small group dynamics for understanding teams (Boh et al., 2007). Network perspective of teams has been a new development in the past decade. Various studies like Oh et. al. (Lungeanu et al., 2014; Oh et al., 2004) stress the importance of network structure in determining team performance. More recent research has focused on self-assembling teams in which members autonomously leave or join teams (Contractor, 2013; Lungeanu et al., 2014).

Link Prediction Due to space constraint we point out some key surveys and papers for DLP. An overview of link prediction in general complex network is by Lü & Zhou (2011) and more specifically for social network we refer to Al Hasan & Zaki (2011). We have generalized topology based methods like Katz (1953) (Katz, 1953), proven successful for social networks (Liben-Nowell & Kleinberg, 2007), for higher order links. For network alignment based link prediction we refer Flannick (2008) and Xie et al. (2012).

4.3 Problems and Preliminaries

4.3.1 Problem Definition

In this chapter we consider the scenario where we have a set of individuals or social *actors*. These *actors* self assemble themselves into *groups* to perform tasks at hand or gather for an event. A *group* therefore, is a subset of all the *actors*. Membership of a *group* can change over time. An *actor* can leave or join a *group*, resulting in changes in *group* membership. When two *actors* work or gather together in the same *group* they develop social tie. These social ties therefore, become the edges in the social *network of actors* (NOA). Moreover, the *actors* that are the part of multiple groups act as ties between groups resulting in a *network of groups* (NOG) (as we had mentioned in

introduction). Given a past history of *groups* formed our problem is to predict *groups* that are likely to form in future by two different evolutionary processes described as follows. Given a *group*, it can absorb an *actor* outside this *group* to form a new group in future. This process is called *incremental accretion* (IA) and the group formed by this process is called *incremental group* (IG) (Figure 4.1). In the second process, rather than all the members of a given *group*, only a subset of them absorb an actor outside the group to form another *group*. This process is *subgroup accretion* (SA) and the corresponding group formed is called *subincremental group* (SG) (Figure 4.1). Note that it is possible that SG and/or IG might have been previously observed or not observed in history. We therefore restrict ourselves to predict only the IG and SG type groups formed by IA and SA processes respectively by assigning prediction scores to them.

An example of such a scenario is collaborations among authors to work on publication. As authors write papers they develop social relations with each other. As authors works in multiple research collaborations they become intermediates between these different research collaborations. Related example can be open-source software development teams.

4.3.2 Problem Statement

Although we had defined models in detain in Chapter 2, we again provide preliminaries for convenient reading, and also, the notations employed in this chapter might be more simplified for the discussion within this chapter. We have a set of n *actors* $V = \{v_1, v_2, ..., v_n\}$. A subset of these *actors* form a *group*. We have a collection of m such groups observed in past, denoted by $G = \{g_1, g_2, ..., g_m\}$ where $g_i \subseteq V$ represents the i^{th} *group*. Cardinality $c_i = |g_i|$ of a group is the number of actors part of it. We have two networks. First, NOG is a hypergraph (Berge & Minieka, 1973) represented as a set $N_g = (V, G)$ with G as the hyperedges over the vertex set V. We also have an incidence matrix $\mathbf{H}$ for N_g of size $(|G| \times |E|)$ with elements defined as $\mathbf{H}(g, v) = 1$ if $v \in g$ else 0. Second, NOA is a *Clique Expanded (CE)* graph, $N_a = (V, E)$ where

$E = \{e_1, .., e_w\}$ are the dyadic edges defined over vertex set V (see Def. 29). Adjacency matrix $\mathbf{A_c}$ of size $(|V| \times |V|)$ for N_a has elements $\mathbf{A_c}(p, q) = 1$ for $(p, q) \in V$ such that $\exists i, \{p, q\} \subseteq g_i$ else 0.

In IA, a *group* $g_i \in G$ can absorb an actor $a \in \{V - g_i\}$ to produce $g_i^a = \{g_i \cup a\}$. Let $g_i^{in} = \{g_i^a\}_{a \in \{V - g_i\}}$ be the set of all the IGs for i^{th} *group*. Our aim therefore, in IA problem is to predict a score to each of the IGs in set g_i^{in} $\forall i \in \{1, 2, .., m\}$.

Considering the second case of SA problem. We define a proper subgroup of g_i as $s_i \subset g_i$ where $s_i \neq \phi$. A subgroup s_i can absorb an actor $a \in \{V - s_i\}$ to produce $s_i^a = \{s_i \cup a\}$. Let $g_i^{sa} = \{s_i^a\}_{a \in \{V - s_i\}}$ be the set of all the SGs for i^{th} *group*. Our aim therefore, in SA problem is to predict a score to each of the SGs in set g_i^{sa} $\forall i \in \{1, 2, .., m\}$.

4.4 Methods

In this section we describe three methods to solve the problems described in the previous section. Each method models the affinity of a given *group* towards an *actor* outside the group in different ways. As pointed out earlier, these methods are inspired from the existing dyadic link prediction (DLP) techniques as well as sociology theories. Success of topology based DLP methods encouraged us to focus on topology derived methods. First, method is a generalization of unsupervised path counting based DLP methods to predict the IGs and SGs. The second approach, is a semi-supervised learning based on network alignment algorithms (Flannick, 2008). It captures the cycles that pass through both the given group and the rest of graph. The third approach, uses a semi-supervised hypergraph label propagation approach. Each of these methods provide a score $\mathbf{S}(i, j)$ between i^{th} *group* and j^{th} *actor*, representing the similarity or affinity between them.

4.4.1 Generalized Katz Score (GKS)

Among the similarity score based methods in DLP, methods based on counting ensemble of paths have been most successful. More specifically Katz (1953) (Katz, 1953) measure

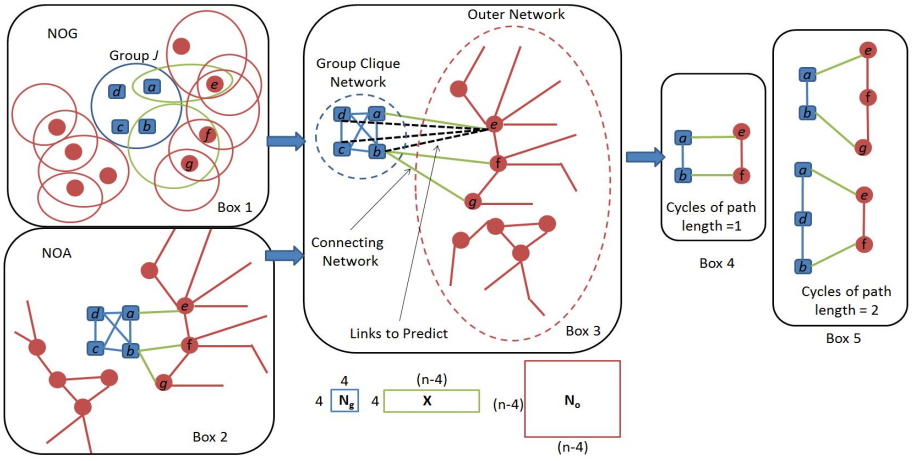

Figure 4.3: Example illustrating different networks used in BRWS for a sample group J consisting of actors $\{a, b, c, d\}$. Box 1 and Box 2 shows the NOG and NOA around Group J actors. In Box 3, we have the 3 network: group clique network (blue), the outer network (red) and bipartite inter network (green). Adjacency matrices used in BRWS are at bottom. Box 4 and Box 5 are example of some cycles of different maximum path length involving actor e. These example cycles (along with other cycles not shown) are used in BRWS to calculate scored of the black dotted edges from group members to actor e. Note for top cycle in box 5 has path length of 2 over outer network and 1 over clique network. However, these lengths are vice versa for the bottom cycle.

has been shown to outperform all other similarity scores (Liben-Nowell & Kleinberg, 2007). Given the NOA graph N_a the Katz Score (KS) between any two nodes i and j is defined as:

$$\mathbf{K}(i,j) = \sum_{l=1}^{\infty} \beta^l |paths_{i,j}^{(l)}| \tag{4.1}$$

where $\mathbf{K}$ is $(|V| \times |V|)$ size matrix containing KS for different pair of vertices, $|paths_{i,j}^{(l)}|$ is number of l length paths between the nodes i and j, and $\beta \in (0,1)$ is parameter that controls the extent to which long paths are penalized. From a sociology perspective number of path between any two nodes capture their social proximity. For example two scientists, in a co-authorship network, who are close to each other in this network, should have many common colleagues or are in similar social circles. Therefore, they are more likely to collaborate. Also smaller length paths reflect greater proximity and are more important, hence, are less penalized or contribute more. In matrix terms it is calculated as:

$$\mathbf{K} = \sum_{l=1}^{\infty} \beta^l \mathbf{A_c}^l \tag{4.2}$$

where $\mathbf{A_c}$ is the adjacency matrix for N_a. Success of this subtle KS method has encouraged us to generalize it to capture the affinity of a *group* for an *actor* outside it. We therefore, define the Generalized Katz Score (GKS) between $i^{th}group$ and $j^{th}actor$ node as follows:

$$\mathbf{S}(i,j) = \frac{1}{c} \sum_{p \in g_i} \sum_{l=1}^{\infty} \beta^l |paths_{p,j}^{(l)}| = \frac{1}{c} \sum_{p \in g_i} \mathbf{K}(p,j) \tag{4.3}$$

where g_i is the $i^{th}group$ of cardinality c. This score is the average proximity, measured as KS, of different *actors* within the group to a given external *actor*. For higher order groups the proportion of the group members close to an external individual is more relevant. Therefore, taking average is intuitive, as it tells that on an average how close is this external individual to the given group. It captures the chances he might get absorbed in this group by taking into account the size of group.

4.4.2 Bi-random Walk Score (BRWS)

GKS makes use of the communication paths to measure the affinity of the each group members to a person outside the group separately. In this section, we model the scenario when outside individual is know by multiple members of the group. Consider the group J (blue color) in Figure 4.3 with four actors $\{a,b,c,d\}$ in it. Let us take the external actor e for whom we wish to observe affinity with J. We can observe that e has a direct link only with the member a. But, b has a direct link with a neighbor f of e. Though, there is no direct relation between b and e, but they have an indirect link through actors both internal (a) as well as external (f) to the group. To quantify the affinity between b and e, quantification process should be guided by both these internal and external links. We model this intuition in a holistic way by capturing the cycles like $\{a \rightarrow b \rightarrow f \rightarrow e \rightarrow a\}$. Instead of simply counting, we use these cycles to learn the affinity of group member to an external actor. We cast this scenario as an alignment problem (Flannick, 2008) where the nodes within group have to be aligned with external nodes. One of the recently proposed algorithm, by Xie et al. (Xie et al., 2012), for global network alignment fits very well to our problem with the following modifications.

Again consider the group in Figure 4.3. We take the clique network of the group and the network outside the group and place them apart as shown in box 3 of Figure 4.3. Next we consider three different networks. First network is the group clique network with adjacency matrix $\mathbf{A_{gc}}$ of size $(c \times c)$. Second network is the network outside this group with adjacency matrix $\mathbf{A_{og}}$ of size $((n-c) \times (n-c))$. Third network is the inter-network which connects group member nodes to nodes external to group. Its adjacency matrix is $\mathbf{X}$ of size $(c \times (n-c))$. Our aim is to learn the matrix $\mathbf{R}$ whose each entry $\mathbf{R}(p,q)$ contains the affinity score for between the p^{th} group member and the q^{th} actor among the external actor nodes. We define a regularization framework, same as Xie et. al. (Xie et al., 2012), over the above three networks. The objective function for

minimization is:

$$\min_{\mathbf{R}} \alpha \sum_{u,v,i,j} (\mathbf{A_{gc}} \otimes \mathbf{A_{og}})_{(i,u),(j,v)} (\mathbf{R}(i,u) - \mathbf{R}(j,v))^2 + (1-\alpha) \sum_{i,u} (\mathbf{R}(i,u) - \mathbf{X}(i,u))^2 \quad (4.4)$$

where $\mathbf{A_{gc}} \otimes \mathbf{A_{og}}$ is the Kronecker product of $\mathbf{A_{gc}}$ and $\mathbf{A_{og}}$. Each $(\mathbf{A_{gc}} \otimes \mathbf{A_{og}})_{(i,u),(j,v)}$ is 1 if $\mathbf{A_{gc}}(i,j) = 1$ and $\mathbf{A_{og}}(u,v) = 1$, in other words i^{th} and j^{th} group members are linked (which should always be true as group is a clique) and u^{th} and v^{th} external actors are also linked, otherwise 0. The first term in the objective aligns group member i with external actor u and group member j with external actor v, if (i,j) are neighbors and (u,v) are also linked. This term therefore, enforces smoothness over $\mathbf{R}$. The second term is a regularization term that uses prior knowledge (like a and e are already connected in our example) stored in $\mathbf{X}$. $\alpha \in (0,1]$ controls the trade-off between these two competing constraints. As proposed by Xie et. al., the most efficient method to minimize is by the following random-walks based recursive model:

$$\mathbf{R} = \alpha \mathbf{A_{gc}} \mathbf{R} \mathbf{A_{og}} + (1-\alpha)\mathbf{X} \quad (4.5)$$

The first term on the right hand side in the t^{th} recursive step becomes $\mathbf{A_{gc}^t} \mathbf{R} \mathbf{A_{og}^t}$. This term mimics a random walks across the group network, inter network and the outer network. For $t = 1$ it represents cycles with group network path and outer network path length at most 1 as shown in Figure 4.3. In general in t^{th} step it captures cycles with path of length at most t in both clique and outer network. Notice the decay factor α penalizes the larger path length cycles recursively. For further algorithmic details we request to refer to Xie et al. (2012).

In their paper Xie et al. (2012) have provided various versions of Bi-random Walk algorithm. In our work we use the sequential version **BiRW_seq** (Algorithm 2). Input for the algorithm are: the group network adjacency matrix $\mathbf{A_{gc}}$, the inter network

matrix $\mathbf{I}$, outer network matrix $\mathbf{A_{og}}$, the decay parameter α, and l_g and l_o are the maximum path length allowed while generating cycles in the group and outer networks respectively. While generating cycles **BiRW_seq** does a random-walk on the group network followed by a random-walk on outer network sequentially in each step. In line 3 the algorithm takes a random-walk over the group clique if the length of the path in the group network is still less than l_g. A second step is taken on the outer network if path on the outer network covered till now is less than l_o (line 6). In our implementation both l_g and l_o are taken as 4. On reaching maximum path lengths on both networks $\mathbf{R}$ is returned.

Algorithm 2 BiRW_seq($\mathbf{A_{gc}}$, $\mathbf{I}$, $\mathbf{A_{og}}$, α, l_g,l_o)

1: $\mathbf{R}^0 = \frac{\mathbf{I}}{sum(\mathbf{I})}$
2: **for all** $t = 1$ **to** $\max\{l_g, l_r\}$ **do**
3: **if** $t \leq l_g$ **then**
4: $\mathbf{R}^{t_{group}} = \alpha \mathbf{A_{gc}} \mathbf{R}^{t-1} + (1-\alpha)\mathbf{I}$
5: **end if**
6: **if** $t \leq l_o$ **then**
7: $\mathbf{R}^{t} = \alpha \mathbf{R}^{t_{group}} \mathbf{A_{og}} + (1-\alpha)\mathbf{I}$
8: **end if**
9: **end for**
10: **return** $(\mathbf{R})$

Let $\mathbf{R_i}$ represent the learned $\mathbf{R}$ for the group g_i. Then the affinity of group g_i for j^{th} external actor is:

$$\mathbf{S}(i,j) = \frac{1}{c} \sum_{p=1}^{c} \mathbf{R_i}(p,q) \tag{4.6}$$

where q is the index in outer network $\mathbf{A_{og}}$ for i^{th} actor in original NOA ($\mathbf{A_c}$). We therefore, learned affinity $\mathbf{S(i,j)}$, for i^{th} group and external actor, supervised using the existing connection of the group members to outer actor network.

4.4.3 Group Label Propagation Score (GLPS)

In the previous sections we developed methods that capture paths and cycles over the NOA but does not takes into account the NOG. In this section we develop label

propagation based score which takes into account the hypergraph structure of the NOG. Intuitively, we start by giving some initial labels only to the members of a given i^{th} group g_i. These labels then diffuse by random-walks through the hypergraph structure of the NOG. Once the random-walks stabilize, the final label for each external vertex is treated as its affinity score for the given group. The final label at a given external actor vertex represents the chances a random walk originating from group member nodes might end up at this vertex. Therefore, modeling a network guided similarity between the group and the external actor.

To realize the above label diffusion as a hypergraph-based learning task. Let $\mathbf{y}$ be the vector of initial labels to the vertices of the NOG hypergraph $N_g(V, G)$ with incidence matrix $\mathbf{H}$. For a "given" group g_i we have $\mathbf{y}(v) = 1$ if $v \in g_i$ else $\mathbf{y}(v) = 0$. We learn the final label vector $\mathbf{f}$. In order to take into account the hypergraph structure we want that the members (vertices) within "any" group (hyperedge) finally get same labels. Also we want the vertices of "given" group retain their initial labels. We capture these aims in the following cost minimization objective:

$$\min_{\mathbf{f}} \frac{1}{2} \sum_{g \in G} \sum_{u,v \in g} \frac{\mathbf{w}(g)\mathbf{H}(g,u)\mathbf{H}(g,v)}{\delta(g)} \left(\frac{\mathbf{f}(u)}{\sqrt{\mathbf{d}(u)}} - \frac{\mathbf{f}(v)}{\sqrt{\mathbf{d}(v)}} \right)^2 + \mu \|\mathbf{f} - \mathbf{y}\|^2 \quad (4.7)$$

where, $\mathbf{w}$ is vector whose entries contain hyperedges weights, $\mathbf{d}$ is vector containing vertex degrees such that for as vertex v: $\mathbf{d}(v) = \sum_{g \in G | v \in g} \mathbf{w}(g)$ and δ is vector containing hyperedge degrees such that for an edge g: $\delta(g) = \sum_{v \in V} \mathbf{H}(g, v)$. The first term is a smoothing term which makes sure that vertices within the same hyperedge have the same scores. So the more number of common hyperedges they are part of the more similar their score becomes. For example if two authors (vertex) have written several papers (hyperedge) together then they are more similar and should be assigned same scores. This term therefore, enforces the hypergraph structure while learning labels. The second term measures the difference between the given labels and the final vertex

scores. The parameter μ then controls the degree of diffusion. In more compact matrix representation:

$$\min_{\mathbf{f}} \mathbf{f}^T \mathbf{L_{hyp}} \mathbf{f} + \mu \|\mathbf{f} - \mathbf{y}\|^2 \tag{4.8}$$

where,

$$\mathbf{L_{hyp}} = \mathbf{I} - \mathbf{D_v^{-1/2}} \mathbf{H}^T \mathbf{W} \mathbf{D_e^{-1}} \mathbf{H} \mathbf{D_v^{-1/2}} \tag{4.9}$$

is the normalized proxy hypergraph Laplacian (see 38; Zhou et al. (2006b)), where $\mathbf{D_e}$ and $\mathbf{D_v}$ are diagonal matrices consisting of hyperedges and vertex degrees, with $(|G| \times |G|)$ and $(|V| \times |V|)$ sizes, respectively. $\mathbf{W}$ is the $(|G| \times |G|)$ diagonal matrix containing weights of hyperedges. It is easy to show that the solution to equation 4.8 is equivalent to solving the following linear system:

$$\mathbf{f_i^*} = (1 - \alpha)(\mathbf{I} - \alpha\theta)^{-1} \mathbf{y}, \tag{4.10}$$

where $\alpha = 1/(1 + \mu)$, $\theta = \mathbf{D_v^{-1/2}} \mathbf{H}^T \mathbf{W} \mathbf{D_e^{-1}} \mathbf{H} \mathbf{D_v^{-1/2}}$ and $\mathbf{f_i^*}$ is the final label vector learned for the i^{th} group (Zhou et al., 2006b). The j^{th} entry in $\mathbf{f_i^*}$ quantifies the affinity between of i^{th} group for the j^{th} actor. Therefore, we have:

$$\mathbf{S}(i, j) = \mathbf{f_i^*}(j). \tag{4.11}$$

Since, our aim is to predict a score for each of the IGs g_i^{in} (or SGs g_i^{sa}) for all $i \in \{1, ..., m\}$. We treat the affinity of group g_i for j^{th} actor ($\mathbf{S}(i, j)$) as the score which reflects the possibility of IG $g_i^a \in g_i^{in}$ (actor a is v_j) being formed in future. In summary, the prediction score for g_i^a (where a is v_j) is taken as $\mathbf{S}(i, j)$. In fact we also assign $\mathbf{S}(i, j)$ as the score for SG s_i^a, same as that for IG g_i^a. Note that one can build more complicated scores, eg: weighted by the size of subgroup, etc. But for this study we restrict ourselves to this simplified scenario. In experimentation section we shall further discuss how to get the ranked list of most likely future groups using these prediction scores.

Table 4.1: *Splits* with fixed boundary year

Boundary Yr	Split No.	Train	Test
1995	A.1	1992-95 (4 yrs)	1996-98 (3 yrs)
1995	A.2	1993-95 (3 yrs)	1996-98 (3 yrs)
1995	A.3	1993-95 (3 yrs)	1996-99(4 yrs)
2000	A.4	1997-00 (4 yrs)	2001-03 (3 yrs)
2000	A.5	1998-00 (3 yrs)	2001-03 (3 yrs)
2000	A.6	1998-00 (3 yrs)	2001-04 (4 yrs)
2005	A.7	2002-05 (4 yrs)	2006-08 (3 yrs)
2005	A.8	2003-05 (3 yrs)	2006-08 (3 yrs)
2005	A.9	2003-05 (3 yrs)	2006-09 (4 yrs)
2007	**Main Split**	2003-07 (5 yrs)	2008-10 (3 yrs)

4.5 Experimental Analysis

In the following the first section describes the dataset statistics. Second section is about evaluation metrics used followed by experiments and analysis in third section.

4.5.1 Dataset and Statistics

In this chapter we have used the popular DBLP dataset (publicly available at (Tang et al., 2007b)) and extracted all the publication from years 1930-2011 for top 10 venues, as listed in `http://academic.research.microsoft.com/`, each for 22 different sub-fields of computer science (total 220 top venues). Here we analyze some interesting statistical properties of this dataset which has motivated this research. Each publication is written by a group of authors and the same group can write multiple publications as well. For both, statistics and experiments we have divided the dataset into various training and test periods (*splits*) as shown in the Tables 4.1. In Table 4.1, each row is a split with a fixed end year of training set (boundary year). Table 4.2 contains the statistics for *(sub)incremental* groups present in "test" period of different splits. In Table 4.2 we refer an actor or group as **old** if it has been observed in training else we

Table 4.2: Incremental Statistics of Testing Periods for the *Splits* in Table 4.1

Split No.	# New Actor Groups (NAG)	# Old Actor Groups (OAG)	# Total Groups	% New Actor Groups	% Old Actor Groups	# Old IGs	# New IGs	Total IGs	% (of OAG) New IGs	% (of OAG) Old IGs	% (of OAG) Total IGs	% (of Total IGs) New IGs	% (of Total IGs) Old IGs
A.1	7863	2343	10206	77.043	22.95	113	386	499	16.47	4.82	21.29	77.36	22.64
A.2	8004	2202	10206	78.42	21.57	81	350	431	15.89	3.68	19.57	81.21	18.79
A.3	11665	2623	14288	81.64	18.36	91	416	507	15.86	3.47	19.33	82.05	17.94
A.4	14308	3629	17937	79.77	20.23	166	550	716	15.16	4.57	19.73	76.81	23.18
A.5	14543	3394	17937	81.08	18.92	139	470	609	13.85	4.095	17.94	77.17	22.82
A.6	20331	3872	24203	84.00	15.99	146	543	689	14.02	3.77	17.79	78.81	21.20
A.7	25081	6351	31432	79.79	20.20	285	936	1221	14.74	4.49	19.23	76.65	23.34
A.8	25482	5950	31342	81.30	18.98	230	796	1026	13.38	3.86	17.24	77.58	22.41
A.9	34747	6827	41474	83.78	16.46	244	897	1141	13.14	3.57	16.71	78.61	21.38
Avg.				80.76	**19.30**				14.72	4.04	**18.76**	**78.47**	21.52
Main Split (IA)	25149	7624	32773	23.26	76.74	308	1085	1393	14.23	4.04	18.27	77.89	21.11
Main Split (SA)	25149	7624	32773	23.26	76.73	1503	3825	5328	50.17	19.71	**69.88**	71.79	28.20

use **new**. Note in case a group has written multiple papers in a train or test period it is counted only once. We observe that on an average around 20% of the groups formed in test period contain no new authors (old actor groups (OAG)). Out of these upto approx 20% are *incremental* groups (IGs) formed by IA process. Moreover, around 80% of these IGs are new and never observed in training. Also, we notice that approx 70% of the groups (with no new author (OAG)) are *subincremental* groups (SGs) formed by SA. All these percentages are highlighted in the Table 4.2. These subtle observations indicate that IA and SA processes are responsible for a large portion of the OAGs formed in future. Therefore, modeling these processes is an important step towards higher order link prediction. Constrained by space limit we have not shown SG statistics (except for the **main split** in Table 4.2). Also the number of actors per split roughly range from $30K$ to $100K$ ($> 100K$ in **main split**) and there are $10K$ to $30K$ groups in "training" period across various splits.

4.5.2 Evaluation Methodology and Experimental Setup

In this section we describe two different kinds of metrics, **global** and **per-group**. The performance of the proposed approaches is evaluated using the training (2004-07) and testing period (2008-10) of **main split** in Table 4.1. The statistics of main split are shown at bottom of Table 4.2. For the groups in the training period, each method: GKS, BRWS and GLPS, was run to output the scores for all IGs (g_i^{in}) and all SGs (g_i^{sa}) for each group ($i \in \{1, ..., m\}$). Now consider the whole set containing all the IGs for all groups. As there might be many repeating IGs we shall only consider unique IGs while taking the maximum score among all the repeating IGs. We sort this unique set of IGs by their scores and get the highest scoring top-N_{top} IGs. We do the exact same thing for SG case and get top-N_{top} SGs. Out of these top-N_{top} groups (IG or SG) the performance over test set (2008-10) is compared using the following metrics:

$$\text{Precision@}N_{top}\ \text{(IA)} = \frac{\text{Number of groups correctly predicted using IA process from top-}N_{top}\text{ list}}{N_{top}} \tag{4.12}$$

$$\text{Recall@}N_{top}\ \text{(IA)} = \frac{\text{Number of collaborations correctly predicted using IA process from top-}N_{top}\text{ list}}{\text{\# of actual IA generated groups}} \tag{4.13}$$

$$\text{Precision@}N_{top}\ \text{(SA)} = \frac{\text{Number of groups correctly predicted using SA process from top-}N_{top}\text{ list}}{N_{top}} \tag{4.14}$$

$$\text{Recall@}N_{top}\ \text{(SA)} = \frac{\text{Number of collaborations correctly predicted using SA process from top-}N_{top}\text{ list}}{\text{\# of actual SA generated groups}} \tag{4.15}$$

The above **global** metrics (equations 4.12 to 4.15) capture overall predictions across all groups. But often times we are more interested in understanding the future of a single group and how it will evolve in future. For this case we simply sort IGs g_i^{in} (or SGs g_i^{sa}) by their scores in descending order, to get the top-N_{top}^g IGs (or SGs) for each i^{th} group. Therefore, for this case we define the following metrics:

$$i^{th}\text{GroupPrecision@}N_{top}^g\ \text{(IA)} = \frac{\text{Number of groups correctly predicted by IA of } i^{th} \text{ group from top-}N_{top}^g\text{ list}}{N_{top}^g} \tag{4.16}$$

$$i^{th}\text{GroupRecall@}N_{top}^g\ \text{(IA)} = \frac{\text{Number of groups correctly predicted by IA of } i^{th} \text{ group from top-}N_{top}^g\text{ list}}{\text{\# of actual IA generated groups from the } i^{th} \text{ group}} \tag{4.17}$$

for each i^{th} group and take average of these metrics to derive the following average metrics:

$$\text{AvgPrecision@}N_{top}^g\ \text{(IA)} = \frac{\text{Sum of } i^{th}\text{GroupPrecision@}N_{top}^g\ \text{(IA) for all groups in training set}}{\text{Total \# of groups in training set}} \tag{4.18}$$

$$\text{AvgRecall@}N_{top}^g\ \text{(IA)} = \frac{\text{Sum of } i^{th}\text{GroupRecall@}N_{top}^g\ \text{(IA) for all groups in training set}}{\text{Total \# of groups in training set}} \tag{4.19}$$

We refer to the metrics in equations 4.18 to 4.19 as the **per-group** metrics. Metrics analogous to equation 4.16 to 4.19: i^{th}GroupPrecision@N_{top}^g (SA), i^{th}GroupRecall@N_{top}^g (SA), AvgPrecision@N_{top}^g (SA) and AvgRecall@N_{top}^g (SA), are defined for the SA case as well.

All the three methods GKS, BRWS and GLPS were implemented in MATLAB. For GKS, BRWS and GLPS the parameters $\beta = \{0.1, 0.3, 0.5, 0.7, 0.9\}$, $\alpha = \{0.1, 0.2, 0.4, 0.5,$

$0.6, 0.8, 0.9\}$, and $\mu = \{0.1, 0.3, 0.5, 0.7, 0.9\}$, respectively, were tested. Using 10-fold cross-validation the following best values were observed: $\{\beta = 0.5, \alpha = 0.6, \mu = 0.1\}$ for global metrics and $\{\beta = 0.5, \alpha = 0.6, \mu = 0.5\}$ for per-group metrics. All the hypergraphs and graphs considered in any of the methods are all unweighted and $l \leq 4$ is only considered in GKS. Experiments were run using Intel Core i7 (2.8 GHz) CPU with 4 GB RAM.

4.5.3 Results and Discussion

In this section we discuss the results obtained using both, the **global** and the **per-group** metrics, for the **main split**. Note that we omit the results from other splits due to space constraints and moreover, they showed results very similar to the main split. Our **GKS** method simply extends Katz (1953) (Katz, 1953) (which is among the most successful DLP methods (Liben-Nowell & Kleinberg, 2007)). We therefore, consider **GKS** as a strong baseline for our evaluation. We would also like to mention that we had explored a number of matrix factorization (MF) based DLP methods (Al Hasan & Zaki, 2011) like MF, Non-Negative MF, SVD, Tri-MF and their variants with (or without) network regularization and sparsity constraints. But all of them performed trivially in comparison to our methods, so we don't include them.

We now discuss the results for **per-group** and **global** metrics. Notice that in per-group scenario we inspect each group individually and compare its affinity for different actors outside it. Therefore, in this case our comparison is between various "actors" to ascertain which actors are more likely to join a given group. Whereas in case of global metrics we try to compare scores of different "groups" (IG or SG) and tell which are more likely to occur in future. Keeping this in mind let us first consider the per-group metrics results in Table 4.3 for $N^g_{top} = 100$. We observe that both **GLPS** and **BRWS** outperforms **GKS** (baseline) consistently for both IA and SA cases. Notice that both **GLPS** and **BRWS** are semi-supervised algorithms, with former supervised by the hypergraph structure and the later learns possible cyclic connections from the

existing connections between a group with outside actors. In contrast, **GKS** which simply works on path enumeration based similarity calculation and lacks supervision performs bad. The good performance of **GLPS** and **BRWS** suggests that both: (1) hypergraph structure (i.e. how groups as composite entities are connected to each other and therefore, also to the groups in which an external actor participates?) (2) cycles passing through group and an external actor node (i.e. which all group members an external actor knows and through which communication cycles?). Similar, trend is observed in results for global metrics shown in Table 4.4 for $N_{tot} = 10000$. Again, **GLPS** and **BRWS** perform better than **GKS**. However, in this case **GLPS** does better than **BRWS** in SA scenario. A possible explanation lies in the fact that **GLPS** keeps the group as well as subgroup structure intact using the hypergraph model. As an example if we have observed a group P containing actors $\{x, y, z, w\}$ and also its subgroup Q with actors $\{x, y, z\}$. While evaluating group P, **BRWS** will not consider the existence of the P's subgroup Q as it models the NOA not NOG. Whereas, **GLPS** models NOG and keep both group as well as subgroup information intact in the hypergraph laplacian of the NOG. This attribute of **GLPS**, to capture subgroups within other groups, helps it to outperform in SA, where this distinction between groups and its subgroups is quiet critical.

Another point to mention is the low values for global metrics and precision in case of per group metrics. The reason for this is due to the inherent difficulty of the problem at hand. The number of groups formed by *(sub)incremental* accretion processes (positives occurrence), though a considerable portion of groups in testing period, are much smaller than the total possibilities. Given a group of size r and n as the total number of actors ($> 100K$ in main split) in the network. There are $PIA = (n - r)$ and $PSA = \{(2^r - 2) \times (n - r)\}$, number of IGs and SGs possible respectively, from the given group. The large number of actors $n(\approx 10^5)$ makes $PIA(\approx 10^5)$ large and $PSA(\approx 10^6)$ (restricting to group size $r_{max} \leq 6$), even much larger, in worst case. Even though, $PIA < PSA$ but the number of IGs formed in test period on an average is much lesser (20%) as compared to (70%) of SGs (4.5.1). Due to this, in case of per group

Table 4.3: Main Split **per-group** metrics results

	GKS	**GLPS**	**BRWS**
AvgPrecision@100 (IA)	0.0210	0.0349	0.0355
AvgRecall@100 (IA)	0.3176	0.6034	0.6050
AvgPrecision@100 (SA)	0.0198	0.0266	0.0271
AvgRecall@100 (SA)	0.2616	0.5149	0.5135

Table 4.4: Main Split **global** metrics results

	GKS	**GLPS**	**BRWS**
Precision@10000 (IA)	0.0020	0.0075	0.0134
Recall@10000 (IA)	0.0144	0.0538	0.0962
Precision@10000 (SA)	0.0052	0.0666	0.0327
Recall@10000 (SA)	0.0098	0.125	0.0614

metric, chances of finding a positive occurrences within $N^g_{tot} = 100$ groups out of PIA or PSA possibilities, is quiet challenging task. This explains low precision in per group scenario. Global metric scenario is even worse. Assume m (around $30K$ in main split) groups and let us say, for approximation purpose, all are of size r. Then there are $GIA = \{m \times PIA\} \approx 10^9$ and $GSA = \{m \times PSA\ \} \approx 10^{10}$ number of total IGs and SGs possible (assuming $r_{max} \leq 6$, $m \approx 10^4$ and $n \approx 10^5$). In case of global metric finding all positive occurrences within just $N_{tot} = 10^4$ groups out of the huge GIA and GSA possibilities explains the low global precision scores. (Note above is a rough worst approximation in which we have restricted $r_{max} \leq 6$ for illustration. In depth discussion will involve actual group cardinality distribution which we leave due to space limitations.) However, there are a limited number of IA and SA generated groups. We can therefore, hope to cover a significant portion of them within top ranked groups. This makes recall more important measure for us and it attains significantly high values as compared to precision at least for the per group metrics. However, in case of global metrics the number of possibilities are huge, resulting in low values for recall as well.

4.6 Conclusions

In this work we have addressed the problem of evolution of Small Groups while highlighting its differences with the general problem of community evolution. We found statistically that two *group accretion* processes are behind the formation of a large percentage of future groups given past history of group collaborations. We have built different models that capture these two processes while being motivated from different theories from social science. We treat the problem of future group prediction as a higher-order link prediction task and have developed three topology based methods. Extensive experiments carried out using DBLP dataset show that our methods give good results while predicting future groups.

Chapter 5

Group Evolution: A Longitudinal Analysis

This chapter revisits the problem of group evolution. But unlike the previous two chapters, this chapter takes time into account and addresses the problem of *temporal hyperedge prediction* by leveraging higher-order algebraic object called *tensors*.

We start by again motivating that the need for multi-actor collaborations is increasingly visible in various fields of social sciences together with the increase in availability of group (multi-actor) communication data like research collaboration data, emails among groups in an organization, etc. *Hypergraphs* are natural structures used to effectively capture multi-actor interactions which conventional dyadic graphs fail to capture. This work aims to empirically evaluate the hypothesis that hypergraphs preserve the information that simple (dyadic) graphs are likely to destroy. For demonstrating this we have addressed the problem of predicting when collaborations become active by modeling the collaboration network as hypergraph. The problem of predicting future activity in a multi-actor collaboration is therefore treated as the problem of predicting activity over a hyperedge. Given that the higher order edge prediction is an inherently hard problem, in this work we restrict ourselves to predict when a hyperedge (collaboration) that has already been observed in past will become active again. We propose a novel

use of hyperincidence temporal tensors and incidence temporal tensors to capture time varying hypergraphs and graphs respectively. Through this common platform of tensor based modeling we quantitatively compare the performance of the hypergraphs based approach with the conventional dyadic graph based approach. Our experiments using author collaboration network from the DBLP dataset show that hypergraphs are the better representation of group activity. Our results demonstrate strength of hypergraphs to predict higher order collaborations (size>4) which is very difficult using dyadic graphs. Moreover, while predicting collaborations of size>2 hypergraphs in most cases provide better results with an average improvement of approx. 45% in F-Score for different group sizes $\in \{3, 4, 5, 6, 7\}$ (Figure 6). Furthermore, we find that using the tensor based modeling hypergraphs outperform graphs both theoretically in storage and time complexity as well as 2x to 20x faster than graphs in practice.

The rest of the chapter is as follows. Next section 5.1 introduces and motivates the problem, section 5.2 is for the related work, section 5.3 describes the problem statement, section 5.4 explains the hypothesis, section 5.5 describes the topology based methods, in section 5.6 the experiments conducted are described and results are discussed, followed by conclusion in section 5.7.

5.1 Introduction

The problem of understanding group dynamics is central to the field of social sciences. Moreover, the increasing use of internet has led to an exponential increase in amount of online group interaction data. As examples, social networking sites like Facebook or Twitter, group communication tools like Skype, Google Hangout, Google Docs and Massive Online multi-player games such as World of Warcraft, are generating social group data at a massive scale. These social datasets provides minute by minute account of various group activities along with the structure and the content of these relationships (**?**).

In the domain of social science, a lot of studies have been conducted to understand

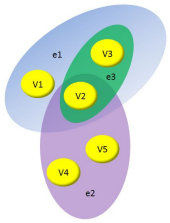

Figure 5.1: Hypergraph

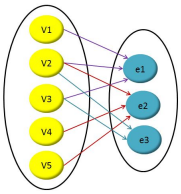

Figure 5.2: Bipartite graph of hypergraph in Figure 5.1

how groups form, their static as well as dynamic attributes and structures, and how they evolve over time (Coleman, 1988). Research collaborations in scientific community are an excellent example of group activities in which individuals of various expertise collaborate to solve a research problem. Collaboration networks from scientific research community have been extensively used for studying team dynamics (Barabâsi et al., 2002; Katz & Martin, 1997; Newman, 2001). Group dynamics has plethora of real life applications for example in building emergency response teams for natural disasters management, automation of team selection for military operations, etc. Therefore, developing computational models which leverage the humongous group data available to study group level phenomenon is of key importance.

The above examples reveal that there can be multiple overlapping collaborations which form a network of collaborations. Modeling such collaborations in dynamic settings where relationship between actors is evolving over time is a challenging task. Unfortunately, most of the prior research in social network analysis deals with dyadic interactions or small well-defined groups (Putnam & Stohl, 1996) rather than at the group level. There are some studies that have dealt with group interactions by collapsing the group into dyadic links (Newman, 2001) and therefore, fail to keep the group level information intact. Ghoshal et. al. (Ghoshal et al., 2009) have used tripartite graphs which captures folksonomy data but is too restrictive to capture variable size social collaborations. Guimera et al. (Guimerà et al., 2005) attempts to model group using node and group attributes which can explain the network structure but fails to deal with individual group evolution.

Hypergraphs are generalization of graphs as well as bipartite graphs (Berge, 1984), which can have more than two node in an edge (rather than simple graphs where only 2 nodes are part of an edge). Therefore, hypergraphs can easily capture the coexistence of more than two entities in a single relation. They have been proposed by Estrada et al. (Estrada & Rodriguez-Velazquez, 2005) for modeling complex networks. Figure 1 shows a hypergraph with five nodes and three edges. However, effective models for capturing hypergraphs are still not clear. We split these models into three categories: bipartite graphs, clique expanded graphs and hypergraphs in their original form (without any transformation to the previous two categories).

Literature is abundant with works that use bipartite network models (Dhillon, 2001; Lind et al., 2005; Zha et al., 2001) but bipartite graphs have rarely been used to capture groups (Faust, 1997; Hayes & Gutierrez, 2004; Newman et al., 2002). Bipartite graphs can also be used for capturing hypergraphs with one set of nodes as the hyperedges and the other as a set of vertices of the hypergraph (Figure 2). We make use of the interesting fact that biadjacency matrix for bipartite representation of hypergraph and the incidence matrix of a hypergraph are exactly the same (Asratian, 1998; Zykov, 1974). This means that for example the incidence matrix of hypergraph in Figure 1 and the bi-adjacency matrix of its corresponding bipartite graph model in Figure 2 are exactly the same. The tensor model we have proposed in this chapter makes use of incidence matrix representation of hypergraph and therefore, our model incorporates the bipartite graph model as well.

Apart from bipartite graphs, another model is the graph structure corresponding a hypergraph obtained by the process of clique expansion (Zien et al., 1999) of each hyperedge. Several clique expansion based methods have been proposed in recent years (Agarwal et al., 2006b; Pu & Faltings, 2012). In their interesting finding, Agarwal et al.(Agarwal et al., 2006b) have shown the theoretical equivalence of several hypergraph based spectral methods to corresponding clique expansion based graph methods. We show that though not for all problems, but rather certain class of problems where the group information needs to remain intact, hypergraphs are likely to give better

results. Hyperedge activity prediction is one such problem that we are addressing. Our hypergraph tensor model when empirically compared with the corresponding clique expanded graph's tensor model, is shown to outperform both in accuracy and space/time complexity.

Although a lot of work done has been done regarding mathematical formulation of hypergraphs in past (Berge, 1984; Berge & Minieka, 1973) as well as recently (Pearson & Zhang, 2012; Xie & Chang, 2013), interest in predictive hypergraph models for real world applications have caught interest recently (Agarwal et al., 2006b; Zhou et al., 2006a). But most of these predictive models (Bu et al., 2010; Pu & Faltings, 2013; Tian et al., 2009) (see related works) transform the problem into an equivalent graph problem by using various transforming mechanisms (Agarwal et al., 2006b; Pu & Faltings, 2012; Zhou et al., 2006a). Also there are several non-predictive works like (Klamt et al., 2009b; Taramasco et al., 2010) which are only interested in statistics of the network modelled as hypergraph. Only recently work by (Xu et al., 2013) attempts to capture the group (hyperedge) information in an intact manner while designing a predictive model. They propose a non-temporal feature-embedding based technique to capture latent features of variable size hyperedges. They formulate the problem of hyperedge prediction as the task of telling whether a given hyperedge in the test set will occur or not. However, this differs from our problem formulation as we predict the top-n hyperedge list after ranking all the hyperedges that have been observed in past. We therefore, have not chosen to compare our work with (Xu et al., 2013).

In order to capture the time varying hypergraphs and graphs we propose a novel application of tensor in the form of incidence or hyper-incidence tensors. We have used research collaborations where a set of authors (*actors*) collaborate for research. Each of these collaboration results in a *publication* and each of these publications represents an instance of this collaboration occurring. This means that the same *collaboration* might show activity recurrently resulting in multiple publications. Each of these *collaboration* is then modeled as a hyperedge in a hypergraph whose each vertex represent an *actor*. Therefore, given a previous history of the collaborations we predict the likelihood of

collaborations getting active again in next time cycle using a supervised approach for hyperedge activity prediction. Our results show that graphs give significantly lower F-Score for higher order groups of size= $\{4, 5, 6, 7\}$ in comparison to hypergraph for most of the cases. In predicting collaborations (hyperedges) higher than size two i.e. more than two entities, hypergraphs in most cases provide better results with an average increase of approx. 45% in F-Score for different sizes $\in \{3, 4, 5, 6, 7\}$ (Figure 6). Furthermore, hypergraphs outperform graphs in terms of both space and time complexity. We verify this through our experiments that tensor models for hypergraphs are approximately 2x to 20x faster than graphs for various phases of the approach proposed. The main contributions of the paper are summarized as follows:

- We show a quantitative comparison between graphs and hypergraphs from an applications perspective.
- We propose a novel application of tensors to capture time varying hypergraphs.
- We have also proposed a novel method of predicting activity of collaborations of higher order using the proposed tensor model of hypergraph.
- We show that hypergraph representation is more accurate with less space and time complexity using DBLP dataset.

5.2 Related Work

Hypergraphs can easily capture the higher-order relationships while incorporating both group and node level attributes. Moreover, research has shown that several social, biological, ecological and technological systems can be better modeled using hypergraphs than using dyadic proxies (Estrada & Rodriguez-Velazquez, 2005). There is an abundant literature of hypergraph theory in past (Berge, 1984; Berge & Minieka, 1973; Zykov, 1974) and many work in the spectral theory of hypergraphs recently (Pearson & Zhang, 2012; Xie & Chang, 2013). The past decade has also seen an increasing interest for

hypergraphs in machine learning community (Tian et al., 2009; Zhou et al., 2006a). Hypergraphs have been used to model complex networks in different fields including biology (Klamt et al., 2009b), databases (Fagin, 1983b) and data mining (Han et al., 1998). In the domain of social sciences, Kapoor et a.l (Kapoor et al., 2013) have proposed centrality metrics for weighted hypergraphs. Tramasco et al. (Taramasco et al., 2010) propose hypergraphs based metrics to evaluate various hypotheses, both semantic and structural, regarding team formation. (Michoel & Nachtergaele, 2012) have used hypergraphs for community detection. (Xu et al., 2013) is a recent work on hyperedge prediction which they formulate in a different way than the ranking based approach proposed in our paper. They address the problem of predicting if a given test hyperedge (or group) will occur or not. For this they develop a feature-embedding technique which leverages upon observed as well as latent social features in a non-temporal setting.

Clique Expanded Graphs Agarwal et al. show theoretically that several unsupervised and supervised spectral methods for hypergraphs are convertible to equivalent graph learning methods (Agarwal et al., 2006b). They propose two: star expansion and clique expansion, graph constructions for hypergraphs and suggest that their associated laplacians can be studied for studying their corresponding hypergraph. But their work lack in empirical testing. Zhou et al. (2006a) propose hypergraph transformation into a graph using Normalized Hypergraph Cut. In a recent work, Pu & Faltings (2012) propose a hyperedge expansion based semi-supervised learning algorithm for hypergraphs that is less sensitive to hyperedge sizes. There are several studies which apply these hyperedge/clique expansion (Agarwal et al., 2006b; Zhou et al., 2006a) based methods for learning over hypergraph (Bu et al., 2010; Pu & Faltings, 2013; Tian et al., 2009).

Bipartite Graphs is a well-established area of mathematics with a huge body of work with numerous mathematical applications (Asratian, 1998). Hypergraphs are also shown to generalize bipartite graphs (Berge, 1984). In context of social networks Newman et al. (2002) have proposed a random graph model for bipartite affiliation networks. Guillaume & Latapy (2004) show that all complex networks can be viewed as

bipartite structures. In machine learning community bipartite graph partitioning has been used for data clustering (Dhillon, 2001; Lind et al., 2005; Zha et al., 2001). Hayes & Gutierrez (2004) have proposed bipartite RDF graphs and show its advantages over the general RDF graphs based approach.

5.3 Hyperedge Prediction Problems and Preliminaries

In this section we describe how higher order collaboration activity prediction can be mapped to hyperedge activity prediction problem.

5.3.1 Problem Statement

In this chapter we have used research collaborations where a set of authors (*actors*) collaborate for research. Each of these collaboration results in a *publication* and each of these publications represents an instance of this collaboration occurring. This means that the same *collaboration* might show activity recurrently resulting in multiple publications. Each of these *collaboration* is then modeled as a hyperedge in a hypergraph whose each vertex represent an *actor*. The problem of *collaboration* activity prediction then boils down to the problem of predicting activity over a hyperedge. Therefore, given a previous history of the collaborations we predict the likelihood of collaborations getting active again in next time cycle. We call this problem as the *old edge* prediction problem or hyperedge activity prediction. From now onwards we will use *old edge* prediction problem or hyperedge activity prediction or simply, hyperedge prediction interchangeably.

5.3.2 Problem Definition

Although we had defined models in detain in Chapter 2, we again provide preliminaries for convenient reading, and also, the notations employed in this chapter might be more simplified for the discussion within this chapter. Let $V = \{v_1, v_2, ..., v_n\}$ be a set of vertices (*actors*). We represent the hypergraph of *collaboration*s using $N_g = (V, G)$

with $\mathbf{H}$ is the incidence matrix of hypergraph which we term as *hyper-incidence matrix*. This matrix $\mathbf{H}$ represent the set of hyperedges (*collaborations*) $G = \{h_1, h_2, ..., h_{m_h}\}$ where each hyperedge $h_k = \{v_1^{h_k}, ..., v_{|h_k|}^{h_k}\} \subseteq V$. Size of $\mathbf{H}$ is therefore, $(m_h \times n)$ and we call s_k as the cardinality (No. of vertices inside h_k) of the hyperedge h_k i.e. $s_k = |h_k|$. Populating this matrix's entries will be discussed in Section 4. We divide time into small snapshots (of size w as shown in Figure 4) with t as its index. $N_c^{(t)}$ is the number of *publications* occurred in snapshot t and there are T number of snapshots in past. $\mathbf{H^{(t)}}$ therefore represents the hyper-incidence matrix for snapshot t. Important thing is that m_h are the number of distinct *collaborations* (hyperedges) in the past (i.e. all previous snapshots). We denote the i^{th} *publication* in the t^{th} snapshot by $c_i^{(t)}$, $\forall i = \{1, 2, ..., N_c^{(t)}\}$. Each of this *publication* $c_i^{(t)}$ represents the occurrance of some *collaboration* (hyperedge) h_k within the snapshot t. A mapping function $\phi(x)$ (many-to-one) is defined which returns the *collaboration* (hyperedge) represented by a given *publication* such that $\phi(c_i^{(t)}) = h_k$, $\forall k \in \{1, ..., m_h\}$.

The problem of *old link* prediction is now defined as follows: Given a past history of collaborations $C_{hist} = \{\mathbf{c}^{(t)}\}_{t=1}^{T}$ (where $\mathbf{c}^{(t)} = \{c_i^{(t)}\}_{i=1}^{N_c^t}$) our goal for the problem of *old link* prediction is to predict the likelihood of future occurrence of each of the hyperedges h_k $\forall k \in \{1, ..., m_h\}$ (i.e. *collaboration*s already observed in past).

5.4 Hypothesis

We claim that modeling social collaborations or interactions as hypergraph is likely to conserve a lot of information that is destroyed when modeled as dyadic graphs. Let us take the example in Figure 3, a *collaboration* of authors A,B and C produced couple of *publications* in a time window of ten years. We aim to predict future likelihood P(A,B,C) of this *collaboration* A,B,C reoccurring or getting active in next time cycle. For hyperedge representation we see that the collaboration A,B,C occurs 2 times in 10 years. Therefore, using a naive frequency based prediction over time, P1(A,B,C)= 2/10 is the likelihood using hyperedge representation. Whereas, on splitting the hyperedge

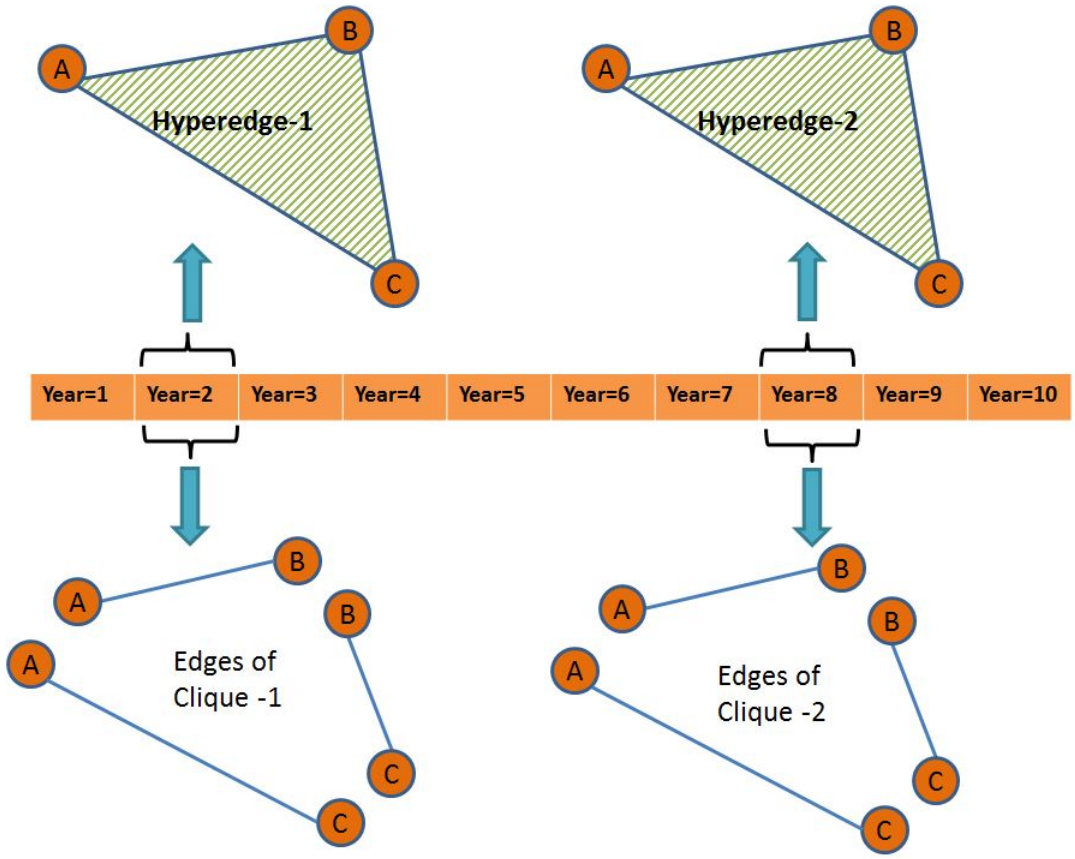

Figure 5.3: Toy Example showing of two *publication*s published by *collaboration* (A, B, C) in year=2 and year=8 with their hyperedge (top) and clique of dyadic edges (bottom) representation.

as cliques of dyadic links, P(A,B) = P(B,C) = P(A,C) = (2/10). Assuming the dyadic edge occurrences as independent events, P2(A,B,C) = P(A,B)x P(B,C) x P(A,C) = (2/10)x(2/10)x(2/10) = (8/1000), is the likelihood using dyadic representation. This is less than what we got from hyperedge prediction (P1). This simple example exhibits the loss of information we are discussing in this chapter. Hypergraph simply keeps the joint probability information intact. Clique expansion destroy the information in ways similar to this toy example.

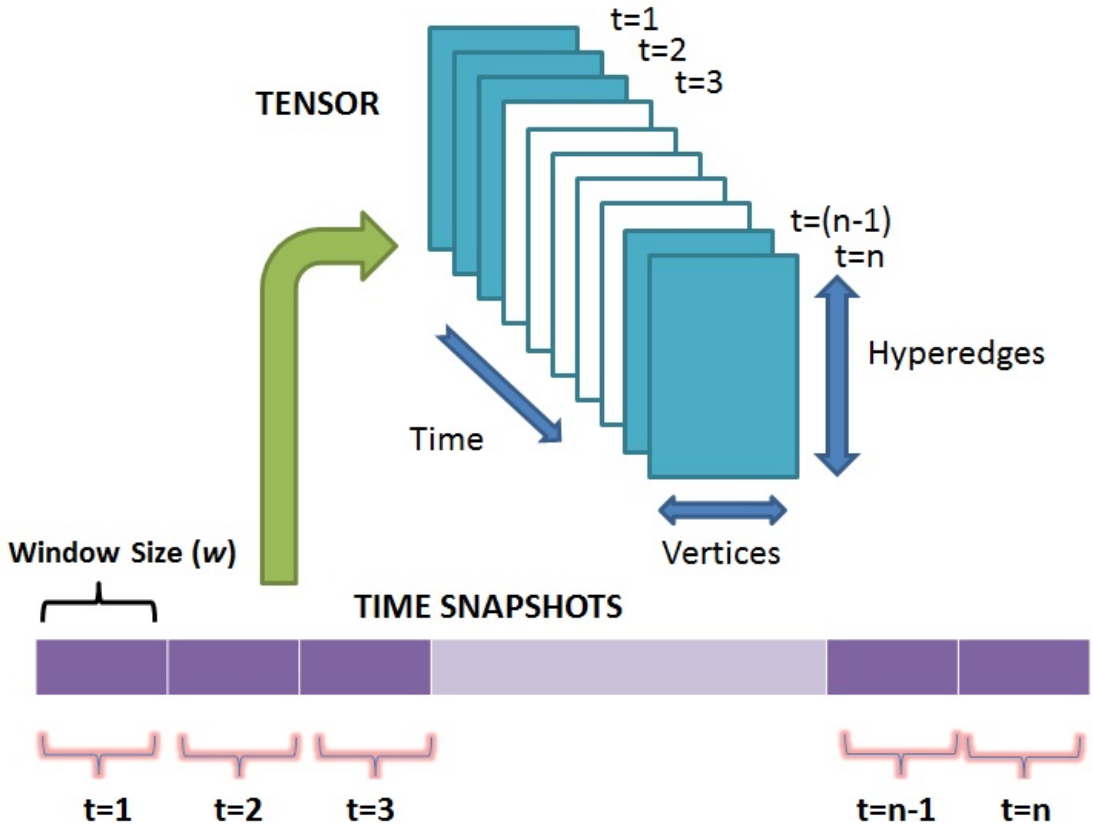

Figure 5.4: A tensor representation of the temporal information (snapshot size=w). Each snapshot data is fed in the corresponding hyper-incidence matrix.

5.5 Proposed Approach

This section describes the approach used to capture the above intuition and build a platform to conduct comparative analysis between graphs and hypergraphs. This section is divided into two sections. In the first section the hypergraph modeling using tensors is explained and the next section described the supervised hyperedge prediction.

5.5.1 Collaboration Modeling

Tensor and Incidence matrix representations

A tensor is a multidimensional, or N-way, array (Pearson & Zhang, 2012) and has

proven to capture multi-dimensional data effectively (Bader & Kolda, 2007). For example Tensors allow to handle time as a separate dimension. This provides more flexibility to creatively manipulate the temporal dimension. Moreover, the temporal patterns can be captured using tensors to predict future patterns rather than just immediate future. Recently tensors have already proved effective in predicting temporal link prediction by Dunlavy et al. (2011). This has encouraged us to capture hypergraphs and graphs using 3-way tensors where the first two dimensions capture the hypergraph or graph incidence matrix and the third dimension captures the temporal information. Keeping the same incidence matrix representation for both graph and hypergraph allows having a parity comparison between the two models. We denote the tensor for graph and hypergraph using $\mathcal{H}_{\mathbf{g}}$ (incidence tensor) and $\mathcal{H}$ (hyper-incidence tensor) which represent array of the snapshots of incidence or the hyper-incidence matrix respectively (Figure 4). Snapshot t refers to a time period: $(w*(t-1), w*t)$.

Similar to hypergraph we represent graph as $N_c = (V, E_c)$ where the graph incidence matrix is $\mathbf{H_g}$ represent the set of edges $E_c = \{e_1, e_2, ..., e_{m_g}\}$. Each edge contains a pair of vertices i.e. $e_k = \{v_i^{e_k}, v_j^{e_k}\} \subseteq V$ (subsection 5.5.1 below describes the method to obtain these edges). For the snapshot t we represent incidence matrix for graph as $\mathbf{H_g^{(t)}}$ and use $\mathbf{H^{(t)}}$ for the hyper-incidence matrix. Here, $\mathbf{H_g^{(t)}}$ has the dimension $(m_g \times n)$ where m_g is the number of distinct dyadic edges between any two actors that have been observed uptil current time. Similarly, the dimension of $\mathbf{H^{(t)}}$ is $(m_h \times n)$ where m_h is the number of distinct multi-actor *collaborations* (hyperedges) between the actors that are observed till now. Note that in $\mathbf{H_g^{(t)}}$ and $\mathbf{H^{(t)}}$ only information of publications in snapshot t is stored but they have same dimension for all values of t.

Therefore, $\mathcal{H}_{\mathbf{g}}(:,:,t) = \mathbf{H_g^{(t)}}$ and $\mathcal{H}(:,:,t) = \mathbf{H^{(t)}}$ both representing array of snapshots of respective incidence matrices. Dimension of $\mathcal{H}_{\mathbf{g}}$ finally becomes $m_g \times n \times T$ and $\mathcal{H}$ becomes $m_h \times n \times T$ dimensional.

Loading Tensors Next step is to extract effective modeling information from historical publication data C_{hist} and feed it into both the graph and hypergraph tensors. The following couple of subsections describe this process for graphs and hypergraphs

Algorithm 3 PREDICT-COLLAB(C_{hist}, $isHypergraph$)

$\mathcal{H}$ tensor (size $m_h \times n \times T$) initialized to all zeros.
$\mathcal{H}_{\mathbf{g}}$ tensor (size $m_g \times n \times T$) initialized to all zeros.
if $isHypergraph$ **then**
  for $c^{(t)} \in C_{hist}$ **do**
    for $c_i^{(t)} \in c^{(t)}$ **do**
      Find k s.t. $\phi(c_i) \equiv h_k$
      for j s.t. $v_j \in \{v_1^{h_k}, ..., v_{|h_k|}^{h_k}\} = h_k$ **do**
        $\mathcal{H}(k, j, t) = \mathcal{H}(k, j, t) + \frac{1}{s_k}$
      end for
    end for
  end for
  $\mathbf{S}_h$ = BUILD-SIMILARITY-MATRIX ($\mathcal{H}$, m_h, n)
  return HYPERGRAPH-PROB-VECTOR ($\mathbf{S}_h$, m_h, n)
else
  for $c^{(t)} \in C_{hist}$ **do**
    for $c_i^{(t)} \in c^{(t)}$ **do**
      Find k s.t. $\phi(c_i) \equiv h_k$
      s_k is the cardinality of hyperedge h_k.
      for Each of the $\binom{s_k}{2}$ dyadic links, d_p of the hyperedge h_k as a clique. **do**
        Find k' s.t. dyadic edge d_p represents the same subset as c_i
        for j s.t. $v_j \in \{v_1^{d_p}, v_2^{d_p}\} = d_p$ **do**
          $\mathcal{H}_{\mathbf{g}}(k', j, t) = \mathcal{H}_{\mathbf{g}}(k', j, t) + \frac{1}{s_k}$
        end for
      end for
    end for
  end for
  $\mathbf{S}_g$ = BUILD-SIMILARITY-MATRIX ($\mathcal{H}_{\mathbf{g}}$, m_g, n)
  return GRAPH-PROB-VECTOR ($\mathbf{S}_g$, m_g, n)
end if
return

Algorithm 4 BUILD-SIMILARITY-MATRIX ($\mathcal{Z}$, n, N_b)

$\mathbf{S}$ similarity matrix of size $n \times N_b$ initialized with all zeros.
K is the number of components.
$[\lambda; \mathbf{A}, \mathbf{B}, \mathbf{C}]$ = CP-ALS($\mathcal{Z}$)
for $k \in \{1, 2, ..., K\}$ **do**
  $\mathbf{S} = \mathbf{S} + \lambda_k \gamma_k \mathbf{a_k} \mathbf{b_k}^\top$
end for
return $\mathbf{S}$

Algorithm 5 HYPERGRAPH-PROB-VECTOR ($\mathbf{S}_h$, m_h, n)

1: $\mathbf{p_h}$ is the probability vector for hyperedge likelihood of length m_h initialized to all one.
2: **for** $i \in \{1, 2, ..., m_h\}$ **do**
3: **for** p s.t. $v_p \in h_i$ **do**
4: $\mathbf{p_h}(i) = \mathbf{p_h}(i) * \mathbf{S}_h(i, p)$
5: **end for**
6: **end for**
7: **return** $\mathbf{p_h}$

separately. We are using the following terms interchangeably: hyperedge and collaboration, occurrence of hyperedge and publication; and vertex and actor.

Hypergraph Case (Line (3-11) of Algorithm 1): All hyper-incidence matrices $\mathbf{H^{(t)}}$ $\forall t$ have the same dimension and thus, the same number, m_h, of unique hyperedges. Each one of these hyperedges h_k $\forall k \in \{1, 2, ..., m_h\}$ represent a unique collaboration between a subset of actors (vertices) i.e. $h_k \subseteq V$. For each of the publication $c_i^{(t)} \in c^t$ for $i = \{1, 2, ..., N_c^{(t)}\}$ find the $k \in \{1, 2, ..., m_h\}$ such that $c_i^{(t)}$ represents the same subset of vertices as h_k i.e. $\phi(c_i^{(t)}) \equiv h_k$. For this index k of the hyperedge, the tensor is filled as $\mathcal{H}(k, j, t) = \frac{m_k}{s_k}$ where j is the index of each vertex which is the part of the hyperedge h_k, s_k is the cardinality of the hyperedge h_k and m_k is the multiplicity of the hyperedge h_k. Multiplicity is calculated as the log (No. of times h_k occurred in t), in other words how many times a particular *collaboration* published some work in snapshot t. This process captures the weight of the hyperedge h_k in the hypergraph tensor. The weight of the hyperedge is modeled as $(\frac{m_k}{s_k})$, as this definition of hyperedge weights is shown to give the best results by Kapoor et al. (2013). This whole process is repeated for all the time snapshots.

Graph Case (Line (14-25) of Algorithm 1): In case of graph also the graph-incidence matrices $\mathbf{H_g^{(t)}}$ $\forall t$ have the same dimension and same number, m_g, of unique edges. Each of these edges g_k represent a unique set (dyadic collaboration) between two vertices (actors), $g_k = \{v_i^{g_k}, v_j^{g_k}\} \subseteq V$. For each publication $c_i^{(t)} \in c^t$ for $i = \{1, 2, ..., N_c^{(t)}\}$ find the $k \in \{1, ..., m_h\}$ such that $\phi(c_i^{(t)}) \equiv h_k$. This hyperedge h_k is

broken in to $\binom{s_k}{2}$ dyadic edges and let us denote each of the dyadic link by d_p. For each of the d_p find the index $k^{'} \in \{1, 2, ..., m_g\}$ for which the d_p represents the same edge as g_k. For this index $k^{'}$ the tensor is filled as $\mathcal{H}_{\mathbf{g}}(k^{'}, j, t) = \frac{m_k}{s_k}$ where j is the index of each vertex which is the part of the edge d_p, s_k is the cardinality of the hyperedge h_k and m_k is the multiplicity of the hyperedge h_k. Thus we model the weight of dyadic link of the clique same as the weight of the original hyperedge (Kapoor et al., 2013). Again, this whole process is repeated for all the time snapshots.

5.5.2 Decomposing the tensors (Algorithm 2)

Next step in the process is to decompose the tensors (loaded in the previous section). These decomposed factors are used in next section for doing *collaboration* prediction. The method proposed in this chapter for decomposition is inspired by CP Scoring using Heuristic (CPH) method of Dunlavy et al. (2011) which has already proven successful for dyadic link prediction. This method exploits the well known CANDECOMP/PARAFAC (CP) (Kolda & Bader, 2009a) tensor decomposition which is analogous to Singular Value Decomposition (SVD) (Golub & Reinsch, 1970) and it converts a tensor into sum of rank one tensors. Given a three dimensional tensor $\boldsymbol{\mathscr{X}}$ with size $J_a \times J_b \times J_c$ its CP decomposition is given by:

$$\boldsymbol{\mathscr{X}} \approx \sum_{f=1}^{F} \lambda_f \mathbf{a}_f \circ \mathbf{b}_f \circ \mathbf{c}_f \tag{5.1}$$

where $\lambda_f \in R^+$, $\mathbf{a}_f \in R^{J_a}$, $\mathbf{b}_f \in R^{J_b}$, and $\mathbf{c}_f \in R^{J_c}$. Each of the products $\lambda_f \mathbf{a}_f \circ \mathbf{b}_f \circ \mathbf{c}_f$ is called the *components* whereas $\mathbf{a}_f$, $\mathbf{b}_f$ and $\mathbf{c}_f$ are called the $factors$ of the decomposition. Note that though $\mathbf{a}_f = \mathbf{b}_f = \mathbf{c}_f = 1$ but these factors are not orthogonal to each other as it is the case in SVD. Also λ_f is the weight for the f^{th} component. CP decomposition gives a unique solution (i.e. its *components* are unique), unlike other tensor decomposition methods, resulting in an attractive method for prediction as the factors can be used directly (Dunlavy et al., 2011). Note that matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ contain the factors $\mathbf{a}_f$,$\mathbf{b}_f$ and $\mathbf{c}_f$ as column vectors.

Algorithm 6 GRAPH-PROB-VECTOR ($\mathbf{S}_g$, m_g, n)

$\mathbf{p_g}$ is the probability vector for edge likelihood of length m_g initialized to all one.
for $i \in \{1, 2, ..., m_g\}$ **do**
  s_i is the cardinality of hyperedge h_i.
  for Each of the $\binom{s_i}{2}$ dyadic links d_p, of the hyperedge h_i as a clique. **do**
    for p s.t. $v_p \in d_p$ **do**
      $\mathbf{p_g}(i) = \mathbf{p_g}(i) * \mathbf{S}_g(i, p)$
    end for
  end for
end for
return $\mathbf{p_g}$

Based upon CPH the similarity between the object i and j is contained in a similarity matrix $\mathbf{S}$ as the entry at (i, j). This matrix is defined as follows:

$$\mathbf{S} = \sum_{k=1}^{K} \gamma_k \lambda_k \mathbf{a_k} \mathbf{b_k}^\top \tag{5.2}$$

where

$$\gamma_k = \frac{1}{T_{buf}} \left(\sum_{t=T-T_{buf}+1}^{T} \mathbf{c}_k(t) \right) \tag{5.3}$$

$\mathbf{a_k}\mathbf{b_k}^\top$ for the component k basically represent the similarity between the object pairs in the k^{th} component. Let the similarity matrix for graph be $\mathbf{S_g}$ (from decomposition of $\mathcal{H}_\mathbf{g}$) and for hypergraph be $\mathbf{S_h}$ (from decomposition of $\mathcal{H}$). Compression over T_{buf} number of past years (buffer) captures the intuition that only the recent past publications are relevant for prediction.

5.5.3 Predicting Collaborations

In this step the similarity matrices $\mathbf{S}_g$ and $\mathbf{S}_h$ are used for predicting the edges or hyperedges. Interpretation of the similarity matrix in our approach is as follows. $\mathbf{S}_g(i, j)$ is the likelihood of the i^{th} dyadic edge occurring in future and also contains vertex j. Similarly, for case of hypergraph $\mathbf{S}_h(i, j)$ is the likelihood of the i^{th} hyperedge along with vertex j inside it. In short after the tensor decomposition (and the subsequent

compression along time dimension) our method outputs a similarity value for all the *actors* for each *collaboration* indicating how likely each of these *actors* can start working with this *collaboration*.

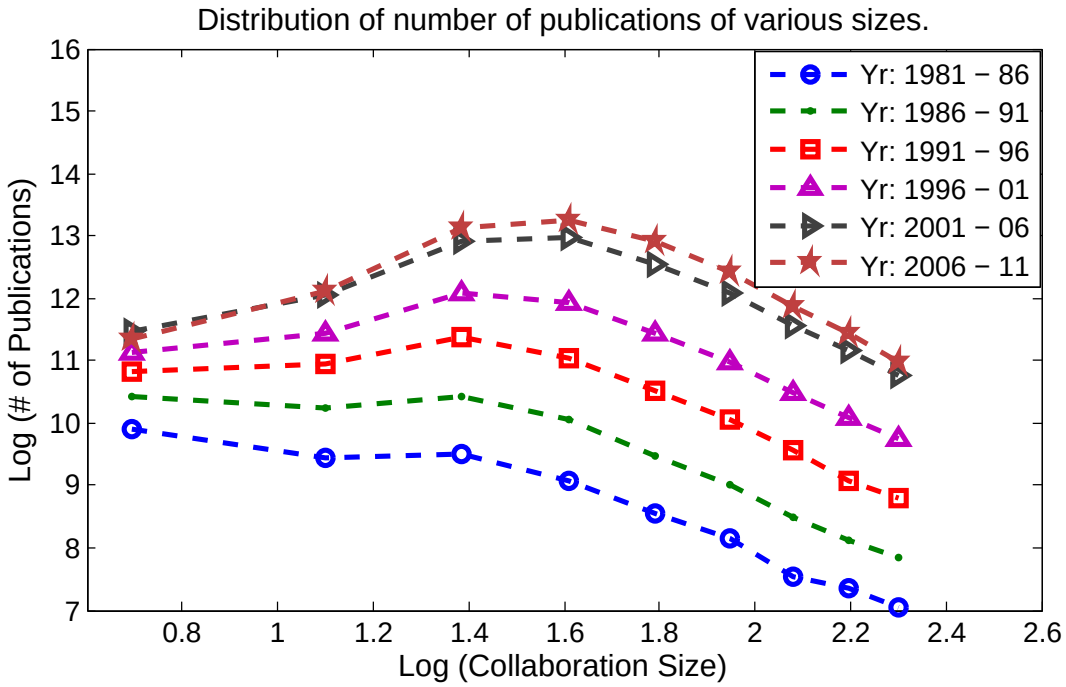

Figure 5.5: Log-Log Plot depicting No. of publications over different sizes of collaboration

Hypergraph Case (Algorithm 3):

If the reassurance of i^{th} hyperedge reoccurs and also contain j^{th} vertex is an event. Assuming that all these events for a particular i^{th} hyperedge for each of the vertices are independent the probability of i^{th} hyperedge reoccurs is defined as:

$$\mathbf{p_h}(i) = \prod_{p \in h_k} \mathbf{S}_h(i,p) \tag{5.4}$$

Graph Case (Algorithm 4):

Similarly, in case of graphs the probability of i^{th} edge reoccurring in future is:

$$\mathbf{q_g}(i) = \prod_{p \in g_k} \mathbf{S}_g(i,p) \tag{5.5}$$

The probability of i^{th} hyperedge reoccurring using the dyadic edge probabilities is:

$$\mathbf{p_g}(i) = \prod_{q \in D} \mathbf{q_g}(q) = \prod_{q \in D} \prod_{p \in g_k} \mathbf{S}_g(q,p) \tag{5.6}$$

where D is the set of dyadic edges that are contained in the clique representation of the i^{th} hyperedge. Another thing to mention is that apart from multiplying the $\mathbf{S}_g(q,p)$ in equation 5, other operators like max, min and addition were tried, but resulted in trivially bad results for graphs. Therefore, we avoid these operators in this study.

The outcome of this whole process (Section 4) is these two vectors: $\mathbf{p_g}$ and $\mathbf{p_h}$. The i^{th} values of $\mathbf{p_g}$ and $\mathbf{p_h}$ are the likelihood of collaboration represented by the i^{th} hyperedge occurring in future as outputted by graph and hyperegraph models respectively. These vectors are used to generate the top-N list as detailed in the Section 5.

5.6 Experimental Analysis

In this section we discuss the experimental setup used to evaluate the performance of the proposed approach. First section describes the dataset, data preprocessing and experimental setup. In the second section, we discuss the various experiments conducted and their analysis.

5.6.1 Dataset and Experimental Setup

We have evaluated the performance of the proposed approach using the popular DBLP dataset (publicly available at (Tang et al., 2007a)) containing publications from years 1930-2011. For the experiments the dataset is divided into training and test periods (*splits*) as shown in the Table 1 and Table 2. As shown in Table 1 the *splits* are designed

Table 1: Training-Testing Splits for variable Testing periods

Split No.	Training Size = 5 yrs	Test Size = 3 yrs	Test Size = 4 yrs	Test Size = 5 yrs
A.1	1981-85	1986-88	1986-89	1986-90
A.2	1986-90	1991-93	1991-94	1991-95
A.3	1991-95	1996-98	1996-99	1996-2000
A.4	1996-2000	2001-03	2001-04	2001-05
A.5	2001-05	2006-08	2006-09	2006-10

Table 2: Training-Testing Splits for variable Testing periods

Split No.	Training Size = 3 yrs	Training Size = 4 yrs	Training Size = 5 yrs	Test Size = 3 yrs
B.1	1983-85	1982-85	1981-85	1986-88
B.2	1988-90	1987-90	1986-90	1991-93
B.3	1993-95	1992-95	1991-95	1996-98
B.4	1998-2000	1997-2000	1996-2000	2001-03
B.5	2003-05	2002-05	2001-05	2006-08

Table 3: Total Edges for all the splits for different sizes and variable testing or training periods

	Split B.1-5				Split A.1-5			
Group Size	Training Size (Years)	No. of Training Edges	No. of Old Test Edges	No. of New Test Edges	Test Size (years)	No. of Training Edges	No. of Old Test Edges	No. of New Test Edges
2	3	192652	49537	210719	3	281203	52887	207369
3	3	129719	21242	176313	3	182562	22269	175286
4	3	60847	6121	93987	3	83421	6397	93711
5	3	23386	1502	39044	3	31907	1567	38979
6	3	9625	442	15990	3	13199	456	15976
7	3	4239	153	6851	3	5733	159	6845
2	4	239635	51746	208510	4	281203	61190	301689
3	4	158564	21964	175591	4	182562	24726	253978
4	4	73281	6319	93789	4	83421	6929	136111
5	4	28150	1546	39000	4	31907	1666	56502
6	4	11620	451	15981	4	13199	494	23266
7	4	5023	155	6849	4	5733	168	9921
2	5	281203	52887	207369	5	281203	65816	385474
3	5	182562	22269	175286	5	182562	25853	320284
4	5	83421	6397	93711	5	83421	7178	170603
5	5	31907	1567	38979	5	31907	1702	70431
6	5	13199	456	15976	5	13199	501	29005
7	5	5733	159	6845	5	5733	170	12339

with constant training period but variable testing periods. Table 2 contains *splits* with variable training periods and fixed length testing periods. Table 3 provides the statistics of the training and test set. It provides the total sum of edge counts across all the splits in two different ranges of splits: Split A.1 to A.5 and Split B.1 to B.5 as mentioned. However, only the No. of training and No. of old edges are useful statistics about the data for the proposed experiments.

The distribution Figure 5.5 is a *log-log* plot showing the distribution of publication counts of various collaboration sizes for different 5 year time periods of DBLP dataset. We observe that the distribution (Figure 5.5) across the different intervals follow a similar pattern. This shows that *splits* that were designed are equivalent as far as conducting experiments is concerned and no bias is involved.

As a preprocessing step, all the single author papers were removed since they do not capture relationships between authors.

For the CP Decomposition (CP-ALS) (that is required for Algorithm 1) Tensor Toolbox (Bader & Kolda, 2007) is used. To find the parameter K for the CP-ALS algorithm we use the ensemble method approach proposed by Dunlavy et al Dunlavy et al. (2011) with $K = \{20, 40, ...200\}$. Also the parameter $T_{buf} = 3$ years is taken (Dunlavy et al., 2011). We have used the term graph and dyadic graph interchangeably.

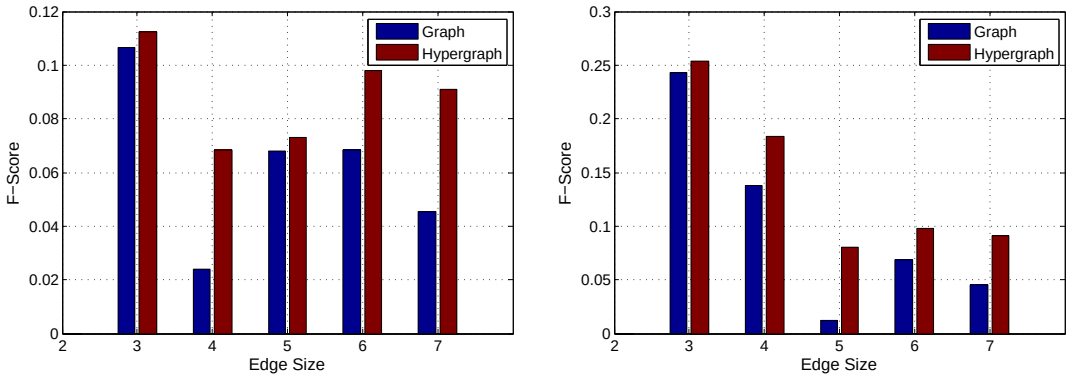

Figure 5.6: Experiment A: (a) AvgF-Score@100 (b) AvgF-Score@1000

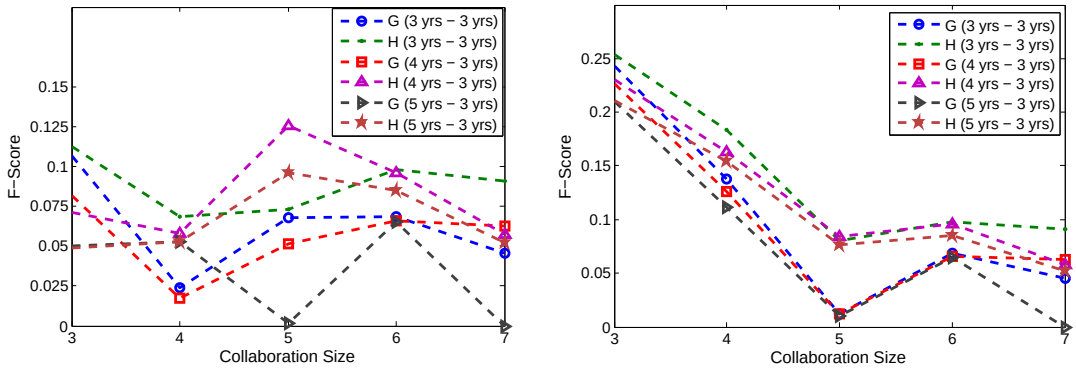

Figure 5.7: Experiment B (Variable Training Size): (a) AvgF-Score@100 (b) AvgF-Score@1000

5.6.2 Evaluation

In this section four experiments are described that evaluate our proposed approach and provide comparative analysis between dyadic and hypergraphs models. Each of the experiment below is conducted using some of the *splits*. The training period of each *split* is used to train the dyadic Graph or Hypergraph models using Algorithm 1. Also each snapshot is of size one year (i.e. one time slice of both tensor contains collaborations from 1 year). The algorithm is run for both graph and hypergraph case to return the edge ($\mathbf{P_g}$) and hyperedge probability ($\mathbf{P_h}$) vectors. These probability vector contains likelihood values for collaborations of different sizes. Each vector is sorted in descending order and the list of top-N elements for each size is extracted. Out of these top-N elements the performance for each size collaboration over test set (old test edges in Table 3) is compared using the following metrics:

$$\text{Precision@N (Size-}h) = \frac{\begin{array}{c}\text{\# of size 'h' collaborations}\\ \text{correctly predicted}\\ \text{from size 'h' top-}N\text{ list}\end{array}}{N} \tag{5.7}$$

$$\text{Recall@N (Size-}h) = \frac{\begin{array}{c}\text{\# of size 'h' collaborations}\\ \text{correctly predicted}\\ \text{from size 'h' top-}N\text{ list}\end{array}}{\begin{array}{c}\text{\# of actual size 'h'}\\ \text{collaborations}\end{array}} \tag{5.8}$$

$$\text{AvgPrecision@N (Size-}h) = \frac{\begin{array}{c}\text{Sum of Precision@N (Size-}h\text{)}\\ \text{for all splits}\end{array}}{\text{Total \# of splits}} \tag{5.9}$$

$$\text{AvgRecall@N (Size-}h) = \frac{\begin{array}{c}\text{Sum of Recall@N (Size-}h\text{)}\\ \text{for all splits}\end{array}}{\text{Total \# of splits}} \tag{5.10}$$

$$\text{AvgF-Score@N (Size-}h) = \frac{\begin{array}{c}\text{2 * AvgPrecision@N (Size-}h\text{)}\\ \text{* AvgRecall@N (Size-}h\text{)}\end{array}}{\begin{array}{c}\text{AvgPrecision@N (Size-}h\text{)}\\ \text{+ AvgRecall@N (Size-}h\text{)}\end{array}} \tag{5.11}$$

This study considers collaborations of size $= \{2, 3, 4, 5, 6, 7\}$ as we are interested only in higher order collaborations (size=2 is used in the analysis as the trivial dyadic

case). AverageF-Score@N and AverageF-Score@N are used in the experiments only for collaborations of size = $\{3, 4, 5\}$ across all the *splits* (over which the experiment is conducted). Collaboration of size = $\{6, 7\}$ the number of predictions are much less as compared to size = $\{3, 4, 5\}$ case. Therefore, for these cases all the predictions (rather than top-N) are used using the following metrics:

$$\text{Precision (Size-}h) = \frac{\text{\# of size 'h' collaborations correctly predicted}}{\text{Total \# of size 'h' predicted}} \tag{5.12}$$

$$\text{Recall (Size-}h) = \frac{\text{\# of size 'h' collaborations correctly predicted}}{\text{\# of actual size 'h' collaborations}} \tag{5.13}$$

$$\text{AvgPrecision (Size-}h) = \frac{\text{Sum of Precision (Size-}h\text{) for all splits}}{\text{Total \# of splits}} \tag{5.14}$$

$$\text{AvgRecall (Size-}h) = \frac{\text{Sum of Recall (Size-}h\text{) for all splits}}{\text{Total \# of splits}} \tag{5.15}$$

$$\text{AvgF-Score (Size-}h) = \frac{\text{2 * AvgPrecision (Size-}h\text{) * AvgRecall (Size-}h\text{)}}{\text{AvgPrecision (Size-}h\text{) + AvgRecall (Size-}h\text{)}} \tag{5.16}$$

In the expriments below AverageF-Score (Size-h) is used as a metric to evaluate the collaboration of size = $\{6, 7\}$ across all the *splits* (over which the experiment is conducted).

Experiment A

This experiment is conducted over the splits A.1 to A.5 for a fixed test period of 3 years (i.e. from Table 1 column 2 are the training periods and column 3 are the corresponding testing periods). AvgF-Score@100 and AvgF-Score@1000 are shown in the Figure 6 (a),(b) for size = $\{3, 4, 5\}$. As shown in Figure 6(a),(b) for size= 3, graphs perform comparably with hypergraphs however for size= 4 prediction using hypergraphs show approx. 150% and 40% increase in F-Score for @100 and @1000 cases respectively. For size>5 Figure 6(a) and Figure 6(b) are identical showing the AvgF-Score. For

size>5 graphs show similar trends as for size= 4 with performance degrading as size increases. As shown in figure 6(a) the F-Score for hypergraph perform better with an increase ranging from 25% for size= 5 to almost 100% for size= 7. This indicates that Hypergraphs maintain the higher order group information intact. However, owing to the limited training set for higher order (size> 6) collaborations hypergraph performance is reduced.

Experiment B

This experiment compares the prediction power of the two models: graph and hypergraphs, when trained over variable size training periods. The time period splits used in this case are B.1 to B.5 (which has fixed test period of 3 years) over training period size from 3 to 5 years as show in the Table 2. For size= $\{3, 4, 5\}$ AvgF-Score@100/1000 and AveragrF-Score for size> 5 curves for different training periods are shown in Figure 7(a),(b). As shown in Fig 7(a),(b) the F-Score curves for graph model are always lower than hypergraph curves for all size collaborations. Another interesting thing to note is that green curves of hypergraph are above pink and pink is above maroon for most sizes in both Figure 7(a),(b). Similar case is there for graphs (blue above red and red above grey). Thus, increasing the training period in several cases results in decrease in prediction power for both graphs and hypergraphs. This shows that the information about past can act as a noise and thus, decrease prediction accuracy.

Experiment C

To get further confidence in the prediction power of hypergraphs we ran experiments with predictions over variable testing periods from three to five years using the splits A.1 to A.5 (Table 1) and fixed training period size= 5 years. For size = $\{3, 4, 5\}$ the AvgF-Score@100 and AvgF-Scorel@1000 curves for different testing periods are shown in Figure 8(a),(b). As shown in Figure 8(a),(b), for size = $\{3, 4\}$ the graph model (curves colored blue, red and gray) is comparable to the green, pink and maroon curves (which represent hypergraph). However at higher order collaborations hypergraph outperform graphs (as inferred from the AvgF-Scores for size >= 5 shown in Figure 8(a),(b)).

Experiment D

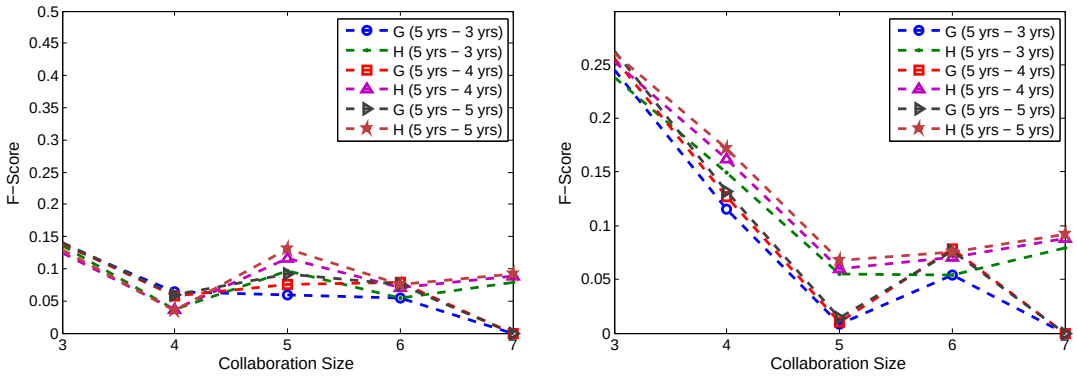

Figure 5.8: Experiment C (Variable Test Size): (a) AvgF-Score@100 and (b) AvgF-Score@1000

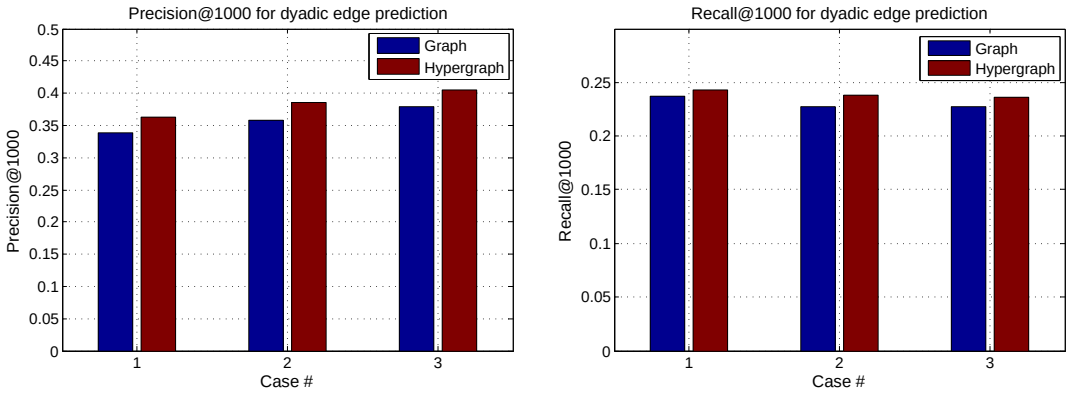

Figure 5.9: Experiment D (Dyadic Link Prediction): (a) AvgPrecision@1000 (b) AvgRecall@1000

This experiment analyzes the trivial case of predicting dyadic links. This experiment consists of three sub-experiments with the following testing and training combinations: A.1-A.5 with training of size = 5 years and test period size= 4 years (Case 1); B.1-B.4 with training period size = 4 years and test period size = 3 years (Case 2); and last, training using B.1-B.4 with training period of 3 years and testing using A.1-A.4 with test period size = 4 years (Case 3). These cases evaluate the dyadic link prediction under various combination of test and training periods. Results of this experiment are shown in the Figure 9(a),(b). It is clearly visible that the maroon bars (hypergraph) for different sub-experiments (Case 1 to 3) are always aslightly higher than blue bars (graphs). This shows that the performance of graphs and hypergraphs is comparable. Although, graphs are themselves sufficiently capture the information needed to predict dyadic links, the proposed tensor model for hypergraph is robust even to predict dyadic links.

5.7 Conclusion and Future Work

In this work we highlight the increasing need to model higher order structures such as groups in online social networks. Hypergraphs are proposed as a natural and highly generalized tool for capturing higher order groups. We show that hypergraphs preserve group information which dyadic graph representation is likely to loose. Therefore, problems that require the group information intact are inclined to be better captured using hypergraphs. We have taken one such problem of higher order collaboration activity prediction and formulated it as a hyperedge activity prediction problem. Further, we propose a novel approach using hyperincidence temporal tensors and graph incidence temporal tensors to effectively capture hypergraphs and graphs respectively. We show that tensors are an excellent way to capture temporal hypergraphs since they perform much better in predicting collaborations of size greater than three (in general higher order hyperedges) in comparison to the dyadic graph representation. Hypergraphs in most cases provide better results with an average increase of approx. 45% in F-Score for differ-

ent sizes $\in \{3, 4, 5, 6, 7\}$ (Figure 6). Moreover, it also turns out that the hyper-incidence tensor model is robust for dyadic edge prediction as well. Furthermore, space/time complexity of hypergraph approach is much lower than graphs, both in theory and practice.

Chapter 6

Hypergraph Embeddings using Symmetric Tensor Decomposition

This chapter serves as the entry point to the second part of this thesis. We dedicated the previous three chapters to develop various hypergraph inference mechanisms. In this and the next chapter, we shift gears and study hypergraph compression techniques.

We start by again motivating that data structured in the form of overlapping or non-overlapping sets is found in a variety of domains, sometimes explicitly but often subtly. For example, teams, which are of prime importance in social science studies, are "sets of individuals"; "item sets" in pattern mining are sets, and for various types of analysis in language studies a sentence can be considered as a "set or bag of words". Although building models and inference algorithms for structured data has been an essential task in the fields of machine learning and statistics, research on "set-like" data remains less explored. Relationships between pairs of elements can be modeled as edges in a graph. However, for modeling relationships that involve all members of a set, a hyperedge is a more natural representation. In this part of the thesis, we focus on the problem of embedding hyperedges in a hypergraph (a network of overlapping sets) to a low dimensional vector space. We propose a tensor-based algebraic model that captures the hypergraph structure in a principled manner (without losing set-level information)

and, more importantly, is for general non-uniform hypergraphs. Our central focus is to: (a) highlight the connection between hypergraphs (topology) and tensors (algebra) and (b) squarely compare graphs versus hypergraphs. Our hypergraph tensor decomposition method outperforms graph (or graph proxies for hypergraph) based spectral baselines on real-world as well as synthetic datasets. We, therefore, argue that the proposed methods are more generic methods suitable for hypergraphs (and therefore also for graphs) that preserve accuracy and efficiency.

Next Section 6.1 introduces and motivates the problem, followed by preliminaries in Section 6.2. We discuss the problem formulation and the tensor based approach in Section 6.3. Section 6.4 is dedicated to the various experimental evaluations conducted and describes the various datasets employed. Finally, we have Section 6.5, which providing literature review, followed by the the conclusion.

6.1 Introduction

In group structured data we have multiple entities related by some form of group relationships. Such data is more abundantly found in the real world than has been usually studied (Estrada & Rodriguez-Velazquez, 2005). In social networks domain, team data from massive online multi-player games (Ahmed et al., 2011) such as World of Warcraft, group communication tools such as Skype and Google Docs and research collaborations (pub; Subbian et al., 2013). There are other fields in which structural relationships between entities is important as well, and large datasets capturing them exist. Examples include Natural Language Processing (Bengio & Bengio, 2000), Biology (Hwang et al., 2008; Klamt et al., 2009a), e-commerce (Christakopoulou & Karypis, 2014; Deshpande & Karypis, 2004) and Chemistry (Bartholomay, 1960). Figure 6.1 shows such examples for networks of groups, sentences and item-sets.

Hypergraph (Berge, 1984), which is a generalization of graphs, is a popular model to naturally capture higher-order relationships between sets of objects (Figure 6.1) (Estrada & Rodriguez-Velazquez, 2005). Within machine learning, algorithms guided by the

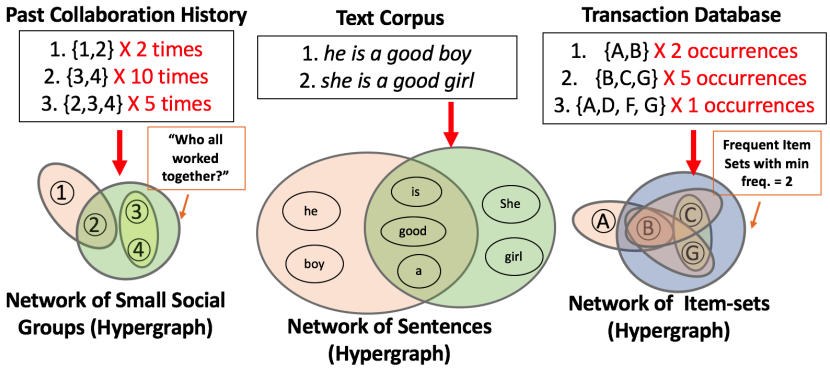

Figure 6.1: "Set-like" hypergraph structures from various domains

structure of such higher order networks (Zhou et al., 2006b) have found applications in a variety of domains (Gao et al., 2013; Li & Li, 2013; Sharma et al., 2015; 2017; Tian et al., 2009).

Often is it useful to embed nodes of graph to low dimensional vector space (by a process referred to as graph embedding) as these embeddings can employed predictive tasks pertaining to nodes (like node classification). In this chapter we focus on learning *hypergraph embeddings*, which involves not just learning node embeddings but also hyperedge embeddings for a given hypergraph. However, unlike graphs (see (Cai et al., 2017)), learning node embeddings for hypergraph have been less explored and we are unaware of any work that considers hyperedge embeddings. In this chapter, we propose method which (1) learn hypergraph embeddings directly, (2) leverage the hypergraph topology and (3) not loose the hyperedge-level joint information. These learned embeddings can then be employed by a supervised or semi-supervised algorithm to perform various predictive tasks pertaining to nodes as well as hyperedges. Example of hyperedge tasks can be for example performance prediction of a team (set of individuals) engaged in a collaborative task or classification of individuals in a cohort for some psychological evaluation by embedding their response vectors (set of "positive" responses).

Most of the past methods learn *node* embeddings for hypergraphs by extending traditional graph embedding methods for hypergraph setting (Hwang et al., 2008; Zhou et al., 2006b). However, as debated by Agarwal et al. (2006a), such representations

can be learned by constructing graphs, which are proxies for the hypergraph structure. However, these proxy graphs for a given hypergraph are not without merit as observed in some recently theoretical studies (Ghoshdastidar & Dukkipati, 2015; 2016). In this work, we highlight that a k-way tensor represents a k-uniform hypergraph (Qi, 2005) and in order to capture hypergraphs in a principled manner any statistical algorithm should work at the level of tensors and not matrices which only model dyadic affinities. This fact was first leveraged in computer vision community by Shashua et. al. (Shashua et al., 2006) where they perform image segmentation using higher-order affinity tensor decomposition and point to its connection with uniform hypergraphs. In this work we leverage these observations to design node embeddings for *general (non-uniform)* hypergraphs based on joint decomposition of various cardinality hypergraph tensors. Further, we also introduce the concept of *dual tensors* for obtaining the hyperedge embeddings directly. As one of the central focus of this work is to squarely compare graphs versus hypergraphs, we compare our method with graph (or graph proxies for hypergraph) based spectral baselines. Proposed hypergraph tensor decomposition outperforms baselines on several real-world as well as synthetic datasets. The main contributions of this work are as follows:

- We propose a general *hypergraph tensor decomposition* method designed for general hypergraphs (containing different cardinality hyperedges) unlike simple uniform hypergraph tensor decomposition, which is restricted to fixed cardinality hyperedges (i.e. uniform hypergraph). We are unaware of any such works or applications employing this approach.

- We propose the novel concept of a *dual tensor*, corresponding to the hypergraph dual that allows us to get a hyperedge embedding directly.

- We provide an empirical comparison between graphs and general hypergraphs in a principled manner using the statistical algorithms proposed.

6.2 Preliminaries

Although we had defined models in detain in Chapter 2, we again provide preliminaries for convenient reading, and also, the notations employed in this chapter might be more simplified for the discussion within this chapter. We consider the scenario where we have a collection of *elements*. These elements can represent individual actors in case of social groups or words in sentences or items in item-sets within a transaction database. In other words a social group or a sentence or an item-set are *sets* which contain these *elements*. Let $V = \{v_1, v_2, ..., v_n\}$ represents n elements and we have m different sets defined over these elements, denoted by $G = \{g_1, g_2, ..., g_m\}$, where $g_i \subseteq V$ represents the i^{th} set. The cardinality $|g_i|$ represents the number of elements in the set. Also each set $g_i \in G$ has an occurrence number $R(g_i)$, which denotes the number of times it has occurred. Such overlapping or non-overlapping sets can be modeled as a *hypergraph* (Berge, 1984), where the nodes and hyperedges, represent the elements and sets, respectively. This hypergraph is represented as $N_g = (V, G)$ with G as the collection of hyperedges over the nodes V. The incidence matrix $\mathbf{H} \in \{0, 1\}^{|G| \times |V|}$ for N_g represents the presence of nodes in different hyperedges with $\mathbf{H}(g_i, v) = 1$ if $v \in g_i$ else 0. We also define degree $d(v)$ of a vertex v as the number of hyperedges incident on this vertex i.e. $d(v) = \sum_{g_i \in G} \mathbf{H}(g_i, v)$.

Problem Statement: Given this setting, our goal is to learn the mapping $\phi : G \rightarrow \mathbb{R}^d$ from hyperedges to feature representations (i.e., embeddings) that can be used to build predictive models involving sets. Here d is a parameter specifying the number of dimensions of the embedding vector. Equivalently, ϕ can be thought of as a look-up matrix of size $|G| \times d$, where $|G|$ is the total number of sets or hyperedges.

6.3 Methodology

In this section we develop tensor (higher-order matrix) based linear algebraic methods that learn node as well as hyperedge embedding by taking into account the joint probability over a hyperedge. The idea behind using tensors is that they retain the set-level

information contained in a hypergraph, unlike the proxy graphs (corresponding to hypergraphs) based techniques (used as baselines in our experiments), which approximate hyperedge or set-level information with dyadic edge-level information. As noted before this idea of tensor based higher-order affinity retention was first highlighted in a statistical setting by Shashua et al. (2006). After which, we argue it has largely remained unnoticed, more specifically in context of hypergraphs. The following proposition from combinatorial probability puts this argument more formally.

Proposition 1 *Given a set of random variables* $X_1, ...X_c$ *(*$c \geq 2$*), and* $H(.)$ *as the information entropy, we have* (Chung et al., 1986, p. 34)*:*

$$H(X_1, ..., X_c) \leq \big(\frac{1}{c-1}\big) \sum_{(i,j)\subseteq 2^{[c]}} H(X_i, X_j) \tag{6.1}$$

Therefore, the joint probability distribution over c cardinality hyperedges is more informative (lower entropy) than the sum total of information attained from probability distributions over each of the $\binom{c}{2}$ dyadic edges.

Although tensors can retain higher order information, most research to date has focused on uniform hypergraphs using symmetric tensors. We propose an approach which is principally suited for *general* hypergraph structured data using higher-order tensors. For a given hypergraph we can extract a sub-hypergraph that only consists of the hyperedges with cardinality k. This sub-hypergraph is a k-uniform hypergraph or k-graph (Cooper & Dutle, 2012). Corresponding to this k-uniform hypergraph, we can define a k^{th} order n-dimensional symmetric tensor (Qi, 2005) $\mathcal{A}^k_{\mathbf{hyp}} = (a_{p_1,p_2,..,p_k}) \in \mathbb{R}^{[k,n]}$ whose elements are as follows:

$$a_{p_1,p_2,..,p_k} = R(g_i) \tag{6.2}$$

where $\{v_{p_1}, v_{p_2}, ..., v_{p_k}\} \in g_i$ and $|g_i| = k, \forall i \in \{1, ..., m\}$. Note that symmetry here implies that the value of element $a_{p_1,p_2,..,p_k}$ is invariant under any permutation of its indices $(p_1, p_2, .., p_k)$. Rest of the elements in the tensor are zeros. We also define the

lexicographically ordered index set for hyperedges:

$$\begin{aligned}\mathcal{P}^k = \big\{\mathbf{p}|\mathbf{p} = (p_1, p_2, .., p_k) \text{ where } \{v_{p_1}, v_{p_2}, ..., v_{p_k}\} \in g_i, \\ \forall g_i \in G \text{ s.t. } |g_i| = k \text{ and } p_1 < p_2 < ... < p_k\big\},\end{aligned} \tag{6.3}$$

and we have different sets $\{\mathcal{P}^k\}\forall k \in \{c_{min}, .., c_{max}\}$ (c_{min} and c_{max} are the maximum and the minimum hyperedge cardinality in the given hypergraph). Notice that $\mathcal{P}^k$ contains unique (non-repetitive) indexes as there is only a single $\mathbf{p}$ corresponding to each of the different hyperedges $g_i \in G$. Consequently, we have $|\mathcal{P}^k| = |\{g_i : |g_i| = k\}|$.

In a similar manner we can also define a *dual tensor*, corresponding to *hypergraph dual* where the roles of nodes and hyperedges are interchanged. We consider all the hyperedges in the hypergraph dual that are of cardinality k. This basically corresponds to all the vertices in the original hypergraph which have a degree of k, i.e., they are part of exactly k hyperedges in the original hypergraph. Corresponding to this k-uniform hypergraph dual, we can define a k^{th} order m-dimensional symmetric dual tensor $\mathcal{A}^k_{\mathbf{dual}} = (a_{q_1,q_2,..,q_k}) \in \mathbb{R}^{[k,m]}$ whose elements are initialized as follows:

$$a_{q_1,q_2,..,q_k} = 1 \tag{6.4}$$

where $\{g_{q_1}, g_{q_2}, ..., g_{q_k}\} \ni v_j$ and $d(v_j) = k, \forall j \in \{1, ..., n\}$. Note that this tensor is also symmetric and rest all the elements in the tensor are zeros. Again, we define the lexicographically ordered index set for *dual* hyperedges (vertices in the *original* hypegraph):

$$\begin{aligned}\mathcal{Q}^k = \big\{\mathbf{q}|\mathbf{q} = (q_1, q_2, .., q_k) \text{ where } v_j \in \{g_{q_1}, g_{q_2}, ..., g_{q_k}\}, \\ \forall v_j \in V \text{ s.t. } |d(v_j)| = k \text{ and } q_1 < q_2 < ... < q_k\big\},\end{aligned} \tag{6.5}$$

and we have different sets $\{\mathcal{Q}^k\}\forall k \in \{d_{min}, .., d_{max}\}$ (d_{min} and d_{max} are the maximum and the minimum vertex degree in the original hypergraph). Again, notice that $\mathcal{Q}^k$ contains unique (non-repetitive) indexes as there is only a single $\mathbf{q}$ corresponding to

each of the different dual hyperedge (vertex in the original hypergraph) i.e. $v_i \in V$. Consequently, we have $|\mathcal{Q}^k| = |\{v_i : d(v_i) = k\}|$.

To realize our aim of learning node or hyperedge embeddings we perform Symmetric Tensor Decomposition (Comon et al., 2008), in a manner similar to Kolda (2015), but jointly across different cardinality hypergraph tensors (for node embeddings) or dual tensors (for hyperedge embeddings). Specifically, for the hyperedge embeddings we consider the following optimization formulation:

$$f(\lambda, \mathbf{Z}) = \sum_{k=\alpha_1}^{\alpha_2} \left\| \mathcal{A}^k_{\mathbf{dual}} - \mathcal{M}^k \right\|^2 = \sum_{k=\alpha_1}^{\alpha_2} k! \left(\sum_{\mathbf{q} \in \mathcal{Q}^k} \left(a^k_{\mathbf{q}} - \mathcal{M}^k_{\mathbf{q}} \right)^2 \right) \tag{6.6}$$

where,

$$\mathcal{M}^k = \sum_{r=1}^{d} \lambda_r \underbrace{(\mathbf{z}_r \otimes \mathbf{z}_r \otimes \ldots \otimes \mathbf{z}_r)}_{k \text{ times}} \equiv \sum_{r=1}^{d} \lambda_r \mathbf{z}_r^{\otimes k} \tag{6.7}$$

with $\otimes$ is the Kronecker product (generalized outer product, ref. Bader & Kolda (2007)), $\mathbf{z}_r \in \mathbb{R}^m$, $\lambda_r \in \mathbb{R}$, $\mathbf{Z} \in \mathbb{R}^{m \times d}$ with $\mathbf{Z}(:, r) = \mathbf{z}_r$ and $a^k_{\mathbf{q}}$ is the $a_{q_1, q_2, .., q_k}$ entry of $\mathcal{A}^k_{\mathbf{dual}}$. Given the dual tensors $\mathcal{A}^k_{\mathbf{dual}}, \forall k \in \{\alpha_1, .., \alpha_2\}$ (α_1 and α_2 are the minimum and maximum cardinalities considered for the dual hyperedges), Equation 6.6 aims to learn symmetric decomposition $\mathcal{M}^k$ with $\mathbf{Z}$ as the matrix containing d-dimensional embeddings for the m hyperedges. Notice that the embeddings in $\mathbf{Z}$ are shared across the different order (k) decompositions $\mathcal{M}^k$. Furthermore, although there are $k!$ entries for each k cardinality hyperedge in the dual tensor $\mathcal{A}^k_{\mathbf{dual}}$, however, the summation in Equation 6.6 is performed over only the unique set of indexes in $\mathcal{Q}^k$. This helps us escape the actual construction of the complete dual tensors ($\mathcal{A}^k_{\mathbf{dual}}$) which otherwise may result in exponential space explosion. This idea of using only unique index sets ($\mathcal{Q}^k$) has been leveraged by both Kolda et. al. (Ballard et al., 2011; Kolda, 2015) as well as by Shashua et al. (2006) in form of semi-norms.

We highlight that $(\mathbf{z}_r^{\otimes k})_{\mathbf{q}} = \mathbf{z}_{q_1 r}\mathbf{z}_{q_2 r}\cdots\mathbf{z}_{q_k r}$ with $\mathbf{z}_{q_p r} = \mathbf{Z}(q_p, r) \in \mathbb{R}$. Also let:

$$\delta_{\mathbf{q}} = \left(a_{\mathbf{q}}^k - \mathcal{M}_{\mathbf{q}}^k\right) = \left(a_{\mathbf{q}}^k - \sum_{r=1}^{d} \lambda_r (\mathbf{z}_r^{\otimes k})_{\mathbf{q}}\right) \tag{6.8}$$

Then the gradients for Equation 6.6 are given by:

$$\frac{\partial f(\lambda, \mathbf{Z})}{\partial \lambda_r} = -2 \sum_{k=\alpha_1}^{\alpha_2} k! \sum_{\mathbf{q}\in\mathcal{Q}^k} \delta_{\mathbf{q}} (\mathbf{z}_r^{\otimes k})_{\mathbf{q}} \tag{6.9}$$

$$\frac{\partial f(\lambda, \mathbf{Z})}{\partial \mathbf{z}_{jr}} = -2 \sum_{k=\alpha_1}^{\alpha_2} k!\, \lambda_r \sum_{\mathbf{q}\in\mathcal{Q}^k} \delta_{\mathbf{q}} \left(\mathbf{z}_{q_1 r}\cdots\mathbf{z}_{q_{s-1} r}\mathbf{z}_{q_{s+1} r}\cdots\mathbf{z}_{q_k r}\right) , \tag{6.10}$$

where, $j = q_s \in \{q_1, \cdot\cdot, q_k\}$. Furthermore, following Kolda (2015) we add the following penalty to address the scaling ambiguity by enforcing the following penalty and gradient:

$$p_\gamma(\mathbf{Z}) = \gamma \sum_{r=1}^{d} (\mathbf{z_r}^\intercal \mathbf{z_r} - 1)^2 \quad , \quad \frac{\partial p_\gamma}{\partial \mathbf{z_r}} = 4\gamma(\mathbf{z_r}^\intercal \mathbf{z_r} - 1)\mathbf{z_r} \tag{6.11}$$

Another important issue that requires attention is that the optimization function $f(\lambda, \mathbf{Z})$ only considers dual hyperedges of cardinalities within $\{\alpha_1, .., \alpha_2\}$. As we will discuss in section **??**, due to scalability reasons most likely $\{\alpha_1, .., \alpha_2\} \subset \{c_{min}, ..., c_{max}\}$. Given this setting, it is therefore, possible that some hyperedges (in the original hypergraph) might not contain any vertex which has degree in the range $\{\alpha_1, .., \alpha_2\}$. Such hyperedges will then suffer from the cold start issue and to remedy that we leverage the auxiliary information in the form of the hypergraph structure. We therefore, introduce the following penalty and gradient:

$$p_\eta(\mathbf{Z}) = \eta \sum_{r=1}^{d} \mathbf{z}_r^\intercal \mathbf{L_{dual}} \mathbf{z}_r \quad , \quad \frac{\partial p_\eta}{\partial \mathbf{z_r}} = 2\eta \mathbf{L_{dual}} \mathbf{z}_r \tag{6.12}$$

where $\mathbf{L_{dual}}$ is the dual hypergraph Laplacian (see detail in Chapter 2). This topology

smoothing constraint allows the diffusion of latent embeddings from the *not* cold-start vertices to nearby cold-start vertices, in order for the later to achieve meaningful non-zero embeddings. Notice that the above Laplacian contains the entire hypergraph dual structure, therefore, allowing information exchange (1) across different cardinality hyperedges and (2) also between hyperedges that are not suffering from cold start. It would be interesting to develop other Laplacians that (a) only affect the embeddings of the cold-start hyperedges, (b) only allow hyperedges of same cardinality to affect each others embeddings and (c) employ hasse diagram based Laplacian Sharma et al. (2017) which explicitly models the cardinality hierarchy.

Putting it all together, the complete optimization objective function is:

$$f_{tot}(\lambda, \mathbf{Z}) = f(\lambda, \mathbf{Z}) + p_\gamma(\mathbf{Z}) + p_\eta(\mathbf{Z}) \tag{6.13}$$

where the choice of γ is the weight of the penalty on the norm of the columns of $\mathbf{Z}$ and the choice of η determines the penalty on the extent of smoothness (similarity) between the embeddings of hyperedges within neighborhood of each other. We call the algorithm that optimizes the above objective (Equation 6.13) as **Hypergraph-Tensor-Decomposition (HTD)** (Algorithm 7).

Algorithm 7 HTD $(d, m, \alpha_1, \alpha_2, \gamma, \eta, \mathcal{A}^k_{\mathbf{dual}} \forall k \in \{\alpha_1, .., \alpha_2\}, \mathbf{L_{dual}})$

```
randomly initialize Z ∈ R^{m×d} and λ ∈ R^d
repeat
  for r = 1 to d do
    Update λ_r using Equation 6.9
    for j = 1 to m do
      Update z_{jr} using Equation 6.10
    end for
    Update z_r using Equation 6.11
    Update z_r using Equation 6.12
  end for
until fit criteria achieved or max. # of iterations exceeded
return Z
```

Notice that till now we have described the process of learning hyperedge embeddings.

The same process can also be used for vertex embedding same algorithm 7 can be used to get the vertex embeddings, by calling:

$$\mathbf{HTD}(d, n, \beta_1, \beta_2, \gamma, \eta, \mathcal{A}^k_{\mathbf{hyp}} \forall k \in \{\beta_1, .., \beta_2\}, \mathbf{L_{hyp}}) \ . \tag{6.14}$$

Parameters β_1, β_2 are the limits for the hyperedge cardinalities considered in Equation 6.6 and $\mathbf{L_{hyp}}$ is the hypergraph laplacian (see Chapter 2). We shall jointly refer to the embeddings achieved for nodes and hyperedges via the above tensor decomposition techniques as **t2v**. Lastly, we would like to highlight that the tensors that we have employed are super-symmetric and hence able to capture distribution over sets rather than sequence. But in general we can employ a k-way tensor which is not symmetric to even capture sequence. In this sense tensors are more general purpose.

6.4 Experiments

6.4.1 Dataset Description

In this chapter, we consider both real-world as well as synthetic datasets. For the former, we make use of five popular real-world datasets from the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml). The five selected datasets have data points whose feature vectors contain mostly boolean-valued features. (From each of the datasets, we removed the very few non-boolean valued features.) We then consider each data point (sample) as a hyperedge with features as vertices. All the features (vertices) which have value one for a given sample (hyperedge) are considered vertices of this sample hyperedge. In short, we treat the data matrix (sample-feature mapping) as the hypergraph incidence matrix (hyperedge-vertex mapping). Below we describe the five datasets:

1. **zoo**: In this dataset, there are several animals each described with a set of boolean attributes like, for example, does it have a feather, or is it airborne. There are several classes of animals, and the aim is to classify animals correctly into its class.

Data	Hyperedges (m)	Vertices (n)	Max. Cardinality	Avg. Cardinality
zoo	101	15	10	6.53
voter	432	16	13	7.91
autism-child	291	14	13	7.46
autism-adolo	103	14	12	7.59
autism-adult	681	14	13	5.81
synthetic	7020	80	6	3.3

Table 6.1: Hypergraph Statistics for various Datasets

2. **voter**: In this, the aim is to classify congressman as democrat versus republican based on 16 key votes, where each vote is boolean (yea or nay). Each congressmen's hyperedge contains only "yay" vertices.

3. **autism-child, autism-adolo, autism-adult**: These three datasets contain psychological evaluation on a cohort of children, adolescents, and adults, respectively, for classification into having ASD disorder or not. Attributes are boolean item responses to behavioral questions. We treat each item (psychological evaluation question) as a vertex, and with each positively responded item (vertex) becomes part of the corresponding individual's hyperedge.

For generating **synthetic** hypergraphs, we employ the recently proposed hypergraph stochastic block-model (**hSBM**) (Ghoshdastidar & Dukkipati, 2015) which are generalization of the traditional stochastic graph model to a hypergraph setting. This model is fairly straightforward. Here we describe a slightly augmented process as we need labels for hyperedges (rather than partition labels for vertices in hSBM). We start with a set of vertices and divide them into two equal sets (we restrict ourselves to only two vertex partitions but can be easily extended for more). We then randomly generate hyperedges between any vertices with probability p_{inter} and within vertices in the same partition with probability p_{intra}. In order to generate clusters or communities one chooses $p_{inter} < p_{intra}$, resulting in more dense connections within each partition rather than across partitions. For a given cardinality c, we set our probability vector

$\mathbf{p}(c) = [p_{inter}, p_{intra}] = [5, 40] \cdot (\log n / n^{(c-1)})$, where n is the number of vertices and order $O(K \cdot \log n / n^{(c-1))}$ is recommended to realize sparse regime i.e. $O(n \log n)$ hyperedges. The particular value of constant K is chosen to realize not too sparse hypergraphs and a sufficiently high number of higher-order hyperedges. We realize 25 hypergraphs using the above process with roughly $15n \log n$ to $25n \log n$ hyperedges. For our experiments we restrict to $[2, 6]$ cardinality hyperedges (average statistics shown in Table 6.1). We then label hyperedges into three different classes: those consisting of only vertices from first vertex partition, those from an only second partition, and those with a mix of vertices from both partitions. The aim, in this case, is to solve this 3-class classification problem.

6.4.2 Evaluation Methodology and Experimental Setup

Methods Compared

We compare hyperedge embeddings obtained from our proposed method with several baselines. Both the baseline, as well as proposed approaches, can output hyperedge embeddings in two different ways:

1. Each method can directly output hyperedge embeddings. Our proposed approach gives direct hyperedge embedding using the dual tensor as the input. We refer to these embeddings as **t2v-dual**. Similarly, we have two baselines that give hyperedge embeddings directly via eigenvalue decomposition of line graph Laplacian (2.47) (herein **e2v-line**) and that of proxy dual hypergraph Laplacian (2.44) (herein **e2v-dual**). Refer to Chapter 2 for details with regards to these Laplacians.

2. We can use these methods to output vertex embeddings first and then combine the vertex embeddings of the vertices within a hyperedge. We evaluate only two kinds of combinations - summing the vertex embeddings (**sum**) or taking the mean of vertex embeddings (**mean**). (Note the later is hyperedge cardinality dependent.) We refer hyperedge embeddings obtained from our proposed approach in this

manner by first obtaining vertex embeddings via decomposing hypergraph tensor, as **t2v**. Similarly, for the two baselines, we can first obtain vertex embeddings via eigenvalue decomposition of graph Laplacian (*Gibson* WCE graph (2.22)) and proxy hypergraph Laplacian (2.42), which can then be combined to get hyperedge embeddings referred to as **e2v** and **e2v-hyp**, respectively. Refer Chapter 2 for details with regards to these Laplacians.

We therefore, have two different comparisons. First is between embeddings obtained from **e2v-line** and **e2v-dual** with **t2v-dual** embeddings. Second, is between **e2v** and **e2v-hyp** with **t2v**. The later comparison happens further separately for both **sum** and **mean** combining strategies.

Evaluation Tasks and Setup

As described in Section 6.4.1, each dataset has a hypergraph and a corresponding classification task associated with it. We first obtain the hyperedge embeddings for the various datasets. The hyper-parameters specific to the proposed tensor method (γ and η) were determined using grid searches over the search spaces: $\gamma = [0.01, 0.5, 5]$ and $\eta = [0.01, 0.5, 5, 10]$. Also the cardinality inputs $\alpha_1 = \beta_1 = 2$ and $\alpha_2 = \beta_2 = 10$. Further, the hyper-parameter of dimension (d) which is common to all the methods is determined by grid search over: $d = [8, 16, 32, 64]$. 5-fold cross-validation was used to determine all the best hyperparameters.

The obtained hyperedge embeddings for a given dataset are utilized for the hyperedge classification task associated with the dataset. For each classification task, we perform several evaluation runs. In each run, we randomly choose 30% of hyperedges as the test set and train logistic regression classifier using the remaining 70% training hyperedges. We chose the Area Under Curve (AUC) as the evaluation metric (the higher, the better). We take average AUC score across five runs as the final AUC. We performed Logistic regression with l_2-norm regularization whose hyper-parameter was chosen by 5-fold cross-validation using a grid search.

Dataset	Embed. Combination	Eigen Decomp.		Tensor Decomp.
		graph (e2v)	proxy hyp. (e2v-hyp)	hyp. tensor (t2v)
Zoo	Node Embed Sum	0.42	0.40	**0.83**
	Node Embed Average	0.60	0.40	**0.83**
		line graph (e2v-line)	proxy dual (e2v-dual)	dual tensor (t2v-dual)
	Only Hyperedge Embed	0.52	0.60	**0.80**

Dataset	Embed. Combination	Eigen Decomp.		Tensor Decomp.
		graph (e2v)	proxy hyp. (e2v-hyp)	hyp. tensor (t2v)
Voter	Node Embed Sum	0.94	0.94	**0.96**
	Node Embed Average	0.94	0.94	**0.96**
		line graph (e2v-line)	proxy dual (e2v-dual)	dual tensor (t2v-dual)
	Only Hyperedge Embed	0.94	0.93	**0.96**

Dataset	Embed. Combination	Eigen Decomp.		Tensor Decomp.
		graph (e2v)	proxy hyp. (e2v-hyp)	hyp. tensor (t2v)
Autism Child	Node Embed Sum	0.98	0.98	**0.99**
	Node Embed Average	0.73	0.71	0.72
		line graph (e2v-line)	proxy dual (e2v-dual)	dual tensor (t2v-dual)
	Only Hyperedge Embed	0.97	0.96	0.95

Dataset	Embed. Combination	Eigen Decomp.		Tensor Decomp.
		graph (e2v)	proxy hyp. (e2v-hyp)	hyp. tensor (t2v)
Autism Adolescent	Node Embed Sum	0.89	0.91	**0.96**
	Node Embed Average	0.65	0.65	**0.76**
		line graph (e2v-line)	proxy dual (e2v-dual)	dual tensor (t2v-dual)
	Only Hyperedge Embed	0.86	0.88	**0.89**

Dataset	Embed. Combination	Eigen Decomp.		Tensor Decomp.
		graph (e2v)	proxy hyp. (e2v-hyp)	hyp. tensor (t2v)
Autism Adult	Node Embed Sum	0.99	**1.00**	**1.00**
	Node Embed Average	0.75	0.76	**0.80**
		line graph (e2v-line)	proxy dual (e2v-dual)	dual tensor (t2v-dual)
	Only Hyperedge Embed	0.98	0.98	0.96

Dataset	Embed. Combination	Eigen Decomp.		Tensor Decomp.
		graph (e2v)	proxy hyp. (e2v-hyp)	hyp. tensor (t2v)
Synthetic	Node Embed Sum	0.94	0.94	**0.99**
	Node Embed Average	0.52	0.56	**0.99**
		line graph (e2v-line)	proxy dual (e2v-dual)	dual tensor (t2v-dual)
	Only Hyperedge Embed	**0.94**	**0.94**	**0.94**

Table 6.2: Classification AUC Scores of tensor methods compared to baselines

6.4.3 Results and Discussion

As detailed in prior section 6.4.1, for each dataset we have two different sets of comparison based on two different ways of obtaining hyperedge embedding: (a) directly obtaining hyperedge embeddings or (b) obtained by aggregating (sum or mean) vertex embeddings. For case (a) we compare **t2v-dual** versus **e2v-line**/**e2v-dual** and for case (b) compare **t2v** versus **e2v**/**e2v-hyp**. Results are reported in Table 6.2 for various datasets. We refer to methods with **t2v-** prefix jointly as "**t2v-** methods". Similarly, "**e2v-** methods" for all methods with **e2v-** prefix.

One of our central hypothesis is that embeddings obtained from methods which try to retain the higher-order information within hypergraphs are better than those methods which do not. We observe from Table 6.2 (the best scores for each row are highlighted in bold) that tensor-based methods (**t2v-** methods) consistently outperform graph-based methods (**e2v-** methods), if not worse. This observation supports our hypothesis that tensor-based models, which preserve the joint information within a hyperedge, indeed, are better models for hypergraph representations.

Also while comparing between the two vertex-embedding aggregation functions: sum and average, we observe (see "sum" and "average" rows in Table 6.2 across **e2v**, **e2v-hyp** and **t2v**) that in most of the cases hyperedge-embeddings obtained via summation of vertex-embeddings, perform better and in some cases those obtained via averaging are performing poorly, especially for **e2v** (like synthetic and autism datasets). Although, summation based aggregation function is performing better in several cases, in a couple of cases, like voter and zoo datasets, its not the case. This observation highlights the fact that the choice of aggregation function is itself a hyper-parameter, which requires tuning in the case of vertex-embedding based methods and a critical drawback. Direct hyperedge-embedding methods, on the other hand, are not subject to any such choices.

Lastly, we note that summation based **t2v** is consistently at least as good as **t2v-dual**, except for autism-adolo and synthetic datasets where it outperforms. Notice that **t2v** employs hypergraph tensors, and the number of non-zeros in them is proportional

to the number of hyperedges and **t2v-dual** uses dual-tensor whose non-zeros are proportional to the number of vertices. Given that in all of our datasets, the number of hyperedges is much more than the number of vertices, the dual-tensor becomes very sparse and possibly explains the relatively weaker performance of **t2v-dual**. (*Remark*: But we still might choose to prefer **t2v-dual** because of the criticism we mentioned in the last paragraph.)

We also note that the difference between **t2v-** methods and the **e2v-** methods is not always similar. For example, this difference is much higher in the zoo dataset as compared to others. Observations such as this and the one described in the last paragraph, points us to several intuitive questions like what kinds of datasets and their hypergraph properties (avg. degree, size, and others) affect the choice of method? Or when would we prefer working using graph methods, and tensor methods might be overkill? Or employ dual-tensor versus hypergraph-tensor? These questions require both theoretical understanding as well as extensive experimentation using a larger variety of data, and hence, we would like to take this as a separate future work. The focus of this work is to introduce higher-order information retaining methodology.

6.5 Related Works

Hypergraphs were studied rigorously by Berge (1976; 1984) as a generalization of graphs and *directed* hypergraphs have been introduced by Bretto (2013). Hypergraph were argued for the first time as a model to naturally capture higher-order relationships between sets of objects across variety of domains by Estrada & Rodriguez-Velazquez (2005). Hypergraphs have been used to model complex networks in different fields including biology (Klamt et al., 2009a), databases (Fagin, 1983a) and data mining (Han et al., 1997; Zhou et al., 2007). Within machine learning, algorithms guided by the structure of hypergraph were introduced by Zhou et al. (2006b) and have found applications in a variety of domains (Gao et al., 2013; Li & Li, 2013; Sharma et al., 2015; 2017; Tian et al., 2009). Simplicial complex (Munkres, 1984) based view of hypergraph

using *hasse lattice* (Skiena, 1990) within machine learning has recently been proposed by Sharma et al. (2017).

Representation learning (RL) (Bengio et al., 2013) focuses on learning features (geometry) of the data (topology). Machine learning algorithms make use of these features for prediction. Traditionally these features are readily available within the data-set or are engineered manually. However, this is often a tedious and human labor-intensive process. RL addresses this issue by learning features (or representations) automatically in a task-dependent supervised or a task-independent unsupervised manner.

Node Representations in Graph: Traditionally unsupervised node embedding learning has been dong using latent models like matrix factorization (Ahmed et al., 2013) or by community detection (Tang & Liu, 2011) based techniques for networks (Belkin & Niyogi, 2001; Cox & Cox, 2000; Roweis & Saul, 2000; Tenenbaum et al., 2000). In each case there is a vector of features learned for a node, each of whose entries reflects node's association with some latent dimension or a network community. More recently, there has been a revived interest in graph embedding in form of *context* oriented techniques (Grover & Leskovec, 2016a; Perozzi et al., 2014; Tang et al., 2015b). These techniques are inspired by recent unsupervised RL methods in NLP (Le & Mikolov, 2014; Mikolov et al., 2013) where word embeddings are learned that are similar to words in a given neighborhood or *context*. These techniques differ in the manner they generate this context as well as in the objective which they optimize. Also there are supervised algorithms learn embeddings which are optimal for the specific task at hand. This results in high accuracy but incurs significant computational cost for training. Recently, several supervised learning algorithms have been proposed for network analysis (Tian et al., 2014; Xiaoyi et al., 2013) and for text networks in a semi-supervised setting (Tang et al., 2015a). Finally, we refer readers to a very recent and comprehensive survey on graph embedding methods by Cai et al. (2017).

Node Representations in Hypergraph: Learning embeddings for nodes within a hypergraph while incorporating the hypergraph topology using *proxy graphs* is introduced by Zhou et al. (2006b). Using graph proxy destroys the hyperedge-level joint informa-

tion and thus, incur loss of information. Also, Agarwal et al. (2006a) squarely criticize that such representations can be learned by constructing graphs, which are proxies for the hypergraph structure. However, these proxy graphs for a given hypergraph are not without merit as observed in some recently theoretical studies (Ghoshdastidar & Dukkipati, 2015; 2016).

Set Representations: RL for sets using neural networks has been proposed recently (Vinyals et al., 2016), where a memory network is used to compose features sequentially but in an order invariant manner. In their very recent paper, Rezatofighi et al. (2016) have tried to answer this set ordering issue by the use of *random set theory*. However, they do not consider embedding but focus on learning set-level probabilities. More importantly, both of these works, do not consider the hypergraph structure of overlapping sets which is the main focus of this paper.

Tensors For comprehensive view of tensors, tensor decomposition as well as applications we refer to the survey by Kolda & Bader (2009b). The connection between k-way tensor and k-uniform hypergraph eigen values was established by Qi (2005). The use of k-way symmetric tensor and their non-negative decomposition for uniform hypergraph partitioning was first introduced by Shashua et al. (2006). But they are again restricted to uniform hypergraphs.

6.6 Conclusion

In this chapter we have proposed a tensor-based algebraic method to generate higher-order representations for both hyperedges (representing sets of nodes) and hypergraph nodes (that also take into account the hypergraph structure). While introducing a new idea of a dual tensors corresponding to the hypergraph dual, we develop a novel approach of using factors from joint decomposition of k-way tensors corresponding to k-uniform sub-hypergraphs, as generic node & hyperedge representations. We show that that our method outperforms several graph based baselines in terms of accuracy. We therefore, argue that our proposed tensor methods are principally suited for hypergraphs (and therefore also for graphs) while maintaining accuracy and efficiency.

Chapter 7

Hyperedge2vec: Hyperedge Representations using Deep Learning

In this chapter, we revisit the problem of hyperedge embedding, but unlike the batch-learning based model in the previous chapter, here we consider an online setting. Inspired by image embeddings methods from computer vision, we treat *hyperedge* as the prime entity to be encoded. We develop deep neural networks based auto-encoders that are trained in an online fashion. Regularization is a crucial ingredient for the successful training of auto-encoders. We use de-noising based regularization where we propose an interesting noise generation schemes which take into account the hypergraph structure. Learned hyperedge representations are then evaluated using various datasets and tasks.

Next Section 7.1 introduces and motivates the problem, followed by preliminaries in Section 7.2. We discuss the problem formulation and the neural network based approach in Section 7.3. Section 7.4 is dedicated to the various experimental evaluations conducted and describes the various datasets employed. Finally, we have Section 7.5, which providing literature review, followed by the the conclusion.

7.1 Introduction

In the last chapter, we had introduced the problem of hyperedge embedding, but unlike the batch-learning based model in the previous chapter, here we consider an online setting. As we have stated several times previously that one of the main objects while modeling hypergraph is to preserve the higher-order information as much as possible. To restate, by higher-order, we mean the hyperedge-level contextual information. For the purpose of preserving this information, in the previous chapter, we had employed higher-order tensors which represent the hypergraph structure mathematically and principally capture the hyperedge context. In this chapter, we seek to capture this hyperedge-level information by leveraging the property of neural networks to approximate arbitrarily complex functions. But unlike the intuitive and interpretative notion of higher-order adjacency tensor, in the case of neural networks, we have a black-box model built using a series of non-linear layers. Although black-box models, neural network approaches in the past few years, they have shown tremendous performance on a wide variety of tasks. Further, neural network-based models are inherently trained in an online fashion, which makes them suitable for application where we have new data coming in real-time. Hence, we choose neural networks for our task of learning ***deep hyperedge embeddings***.

In general, the literature on graph embeddings is significantly more than on hypergraph embeddings. Also, until recently, the models for embedding graph structures have been focused more on techniques other than neural networks (Grover & Leskovec, 2016a; Perozzi et al., 2014; Tang et al., 2015b). Those that are based on neural networks are of mainly two kinds. One is based on the generalization of the recurrent neural network (RNN) for graph data (Gori et al.; Li et al., 2015; Scarselli et al., 2009). Second, line of research generalizes of convolutional neural networks (CNN) to a graph setting (Atwood & Towsley, 2015; Bruna et al., 2013; Defferrard et al., 2016). However, all of these works on graph-based neural networks, which we described so far, are designed for graphs and aim to either learn node embeddings or designed for tasks that require the entire graph's embedding. In fact, we are unaware of any work within large-scale network analysis that

considers embeddings at the hyperedge-level. In contrast, in the natural language processing (NLP) literature, methods for learning embeddings for higher-order "sequential" structures like sentences and paragraphs have been proposed (Kalchbrenner et al., 2014; Le & Mikolov, 2014). Also recently, there has been an interest in modeling "set-like" structures within the deep-learning community (Rezatofighi et al., 2016; Vinyals et al., 2016). However, they do not consider the hypergraph structure between the sets, and therefore, do not model hypergraphs in a principled manner. Here we attempt to permeate this gap by developing a neural network supported hypergraph embedding methods.

We emphasize that in regular graphs, edges are not as impressive as nodes, since they always connect only two nodes, and therefore a focus on nodes is justified. However, given that hyperedges can have vastly varying degrees, they are as interesting and vital as nodes, and thus should be treated firsts class objects. We aim to fill this research gap by directly learning representations for hyperedges while taking into account the topological connectivity between the various hyperedges in a hypergraph.

Inspired by the image auto-encoders from computer vision, we consider the hyperedge as the primary entity to be encoded and develop ***hyperedge auto-encoders***. But unlike the images, where we use Gaussian noise per pixel to generate noisy image samples, hyperedges are discrete structures and require a discrete noise. Random discrete noise like salt-pepper noise might result in entirely unrelated noisy hyperedge samples, which may result in a non-meaningful training. Instead, we harness the hypergraph structure and use it to sample noisy hyperedges for a given original hyperedge. We devise a variety of random walk schemes on the hypergraph structure and generating meaningful noisy hyperedges. During the evaluation of these embeddings, our focus is three folds. We aim to examine how the embeddings that use graph topology perform in comparison to those leveraging the hypergraph structure. Secondly, we aim to study how shallow architectures perform in the relation of deep auto-encoders. Lastly, we intend to contrast the different noise generation schemes that we have designed. Hence, we design several experiments to carry out these evaluations using both real as well as synthetic datasets.

7.2 Preliminaries

Although we had defined models in detain in Chapter 2, we again provide preliminaries for convenient reading, and also, the notations employed in this chapter might be more simplified for the discussion within this chapter. Here we consider the scenario where we have a collection of *elements*. These elements can represent individual actors in case of social groups or words in sentences or items in item-sets within a transaction database. In other words a social group or a sentence or an item-set are *sets* which contain these *elements*. Let $V = \{v_1, v_2, ..., v_n\}$ represents n elements and we have m different sets defined over these elements, denoted by $G = \{g_1, g_2, ..., g_m\}$, where $g_i \subseteq V$ represents the i^{th} set. The cardinality $|g_i|$ represents the number of elements in the set. Also each set $g_i \in G$ has an occurrence number $R(g_i)$, which denotes the number of times it has occurred. Such overlapping or non-overlapping sets can be modeled as a *hypergraph* (Berge, 1984), where the nodes and hyperedges, represent the elements and sets, respectively. This hypergraph is represented as $N_g = (V, G)$ with G as the collection of hyperedges over the nodes V. The incidence matrix $\mathbf{H} \in \{0, 1\}^{|G| \times |V|}$ for N_g represents the presence of nodes in different hyperedges with $\mathbf{H}(g_i, v) = 1$ if $v \in g_i$ else 0. We also define degree $d(v)$ of a vertex v as the number of hyperedges incident on this vertex i.e. $d(v) = \sum_{g_i \in G} \mathbf{H}(g_i, v)$.

Problem Statement: Given this setting, our goal is to learn the mapping $\phi : G \rightarrow \mathbb{R}^d$ from hyperedges to feature representations (i.e., embeddings) that can be used to build predictive models involving sets. Here d is a parameter specifying the number of dimensions of the embedding vector. Equivalently, ϕ can be thought of as a look-up matrix of size $|G| \times d$, where $|G|$ is the total number of sets or hyperedges.

7.3 Methodology

In this section, we describe the overall methodology and approach to learn hyperedge embeddings using neural networks. We start, in Subsection 7.3.1, by describing the

neural network-based autoencoders – specifically the denoising autoencoders which rely on "denoising" regularization scheme. Next, in Subsection 7.3.2, we describe the formulation of the hyperedge compression problem as training autoencoders and develop meaningful noise generation schemes for hypergraphs.

7.3.1 Denoising Autoencoders

An autoencoder (Bengio et al., 2009) takes an input vector $\mathbf{x} \in [0,1]^n$ and maps it to a latent representation $\mathbf{z} \in [0,1]^d$. This is typically done using an affine mapping followed by a non-linearity (more so when the input, like in our case, is binary (Vincent et al., 2010)): $\mathbf{z} = f_\theta(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$, with parameters $\theta = \{\mathbf{W}, \mathbf{b}\}$. Here, σ is a sigmoid function defined as $\sigma(x) = 1/(1+e^{-x})$, $\mathbf{W}$ is a $\mathbb{R}^{n \times d}$ weight matrix and $\mathbf{b}$ is the offset. This latent representation is then used to reconstruct a vector $\mathbf{y} = g_{\theta'}(\mathbf{z}) = \sigma(\mathbf{W}'\mathbf{z} + \mathbf{b}')$, in the input space, $\mathbf{y} \in [0,1]^n$ with parameters $\theta' = \{\mathbf{W}', \mathbf{b}'\}$. The mappings f_θ and $g_{\theta'}$ are referred to as the **encoder** and **decoder**, respectively. Figure 7.1 displays this process diagrammatically. The representation $\mathbf{y}$ is learned by minimizing the following reconstruction error:

$$\theta^*, \theta'^* = \arg\min_{\theta^*, \theta'^*} \frac{1}{m} \sum_{i=1}^{m} L(\mathbf{x}_i, \mathbf{z}_i) = \arg\min_{\theta^*, \theta'^*} \frac{1}{m} \sum_{i=1}^{m} L(\mathbf{x}_i, g_{\theta'}(f_\theta(\mathbf{x}_i)) \tag{7.1}$$

where L is a loss function, which in case of binary or bit probabilities is often chosen as the *cross-entropy loss*:

$$L(\mathbf{x}, \mathbf{z}) = \sum_{j=1}^{n} [\mathbf{x}(j) \log \mathbf{z}(j) + (1 - \mathbf{x}(j)) \log(1 - \mathbf{z}(j))] \tag{7.2}$$

In their paper, Vincent et al. Vincent et al. (2010) have shown that minimizing reconstruction amounts to maximizing the lower bound on the mutual information between input $\mathbf{x}$ and the representation $\mathbf{y}$. However, they have further argued (Vincent et al. (2008)) that $\mathbf{y}$ retaining information about input $\mathbf{x}$ is insufficient. They further, propose the idea that the learned representation should be able to recover (*denoising*)

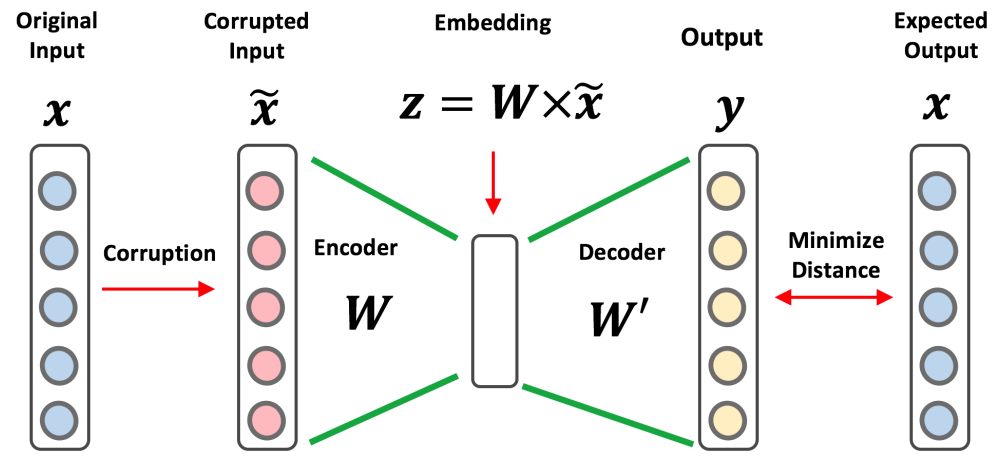

Figure 7.1: Example depicting a neural network based Auto-encoder with single hidden layer (which outputs the embedding).

the original input even after being trained with corrupted input (*adding noise*). They generate the corrupted input ($\tilde{\mathbf{x}}$), using a stochastic mapping $q(\tilde{\mathbf{x}}|\mathbf{x})$. Choice of noise is usually either *Gaussian* for real inputs and *Salt-and-pepper noise* for discrete inputs. The *denoising autoencoder* then learns the representation for each input, $\mathbf{x}$, same as Equation 7.1, but with the following modified loss function: $L(\mathbf{x}(i), g_{\theta'}(f_\theta(\tilde{\mathbf{x}}(i))))$.

7.3.2 Hasse Denoising Autoencoder

We leverage the denoising autoencoder for learning representation for hyperedges by treating each hyperedge as the object to be encoded. We shall be using the terms hyperedge representation or hyperedge embedding interchangeably, to refer to the hyperedge encodings obtained from the autoencoder. Training data for a given hypergraph consists of pairs of noisy and original hyperedges. We consider an i^{th} hyperedge as its binary vector representation: $\mathbf{e_i} = \mathbf{H}(i,:)$ (where $\mathbf{e_i} \in [0,1]^n, \mathbf{e_i}(j) = 1$ if $v_j \in e_i$ else 0). For a given hyperedge, $\mathbf{e_i}$, which we refer to as the *original hyperedge*, we can generate a *noisy hyperedge*, $\tilde{\mathbf{e}}_\mathbf{i}$. The pair, $(\tilde{\mathbf{e}}_\mathbf{i}, \mathbf{e_i})$, therefore, constitutes a single training data point. The *denoising autoencoder* then learns its weights, $(g_{\theta'}, f_\theta)$, while minimizing the following loss for a given training data point $(\tilde{\mathbf{e}}_\mathbf{i}, \mathbf{e_i})$: $L(\mathbf{e_i}, g_{\theta'}(f_\theta(\tilde{\mathbf{e}}_\mathbf{i})))$.

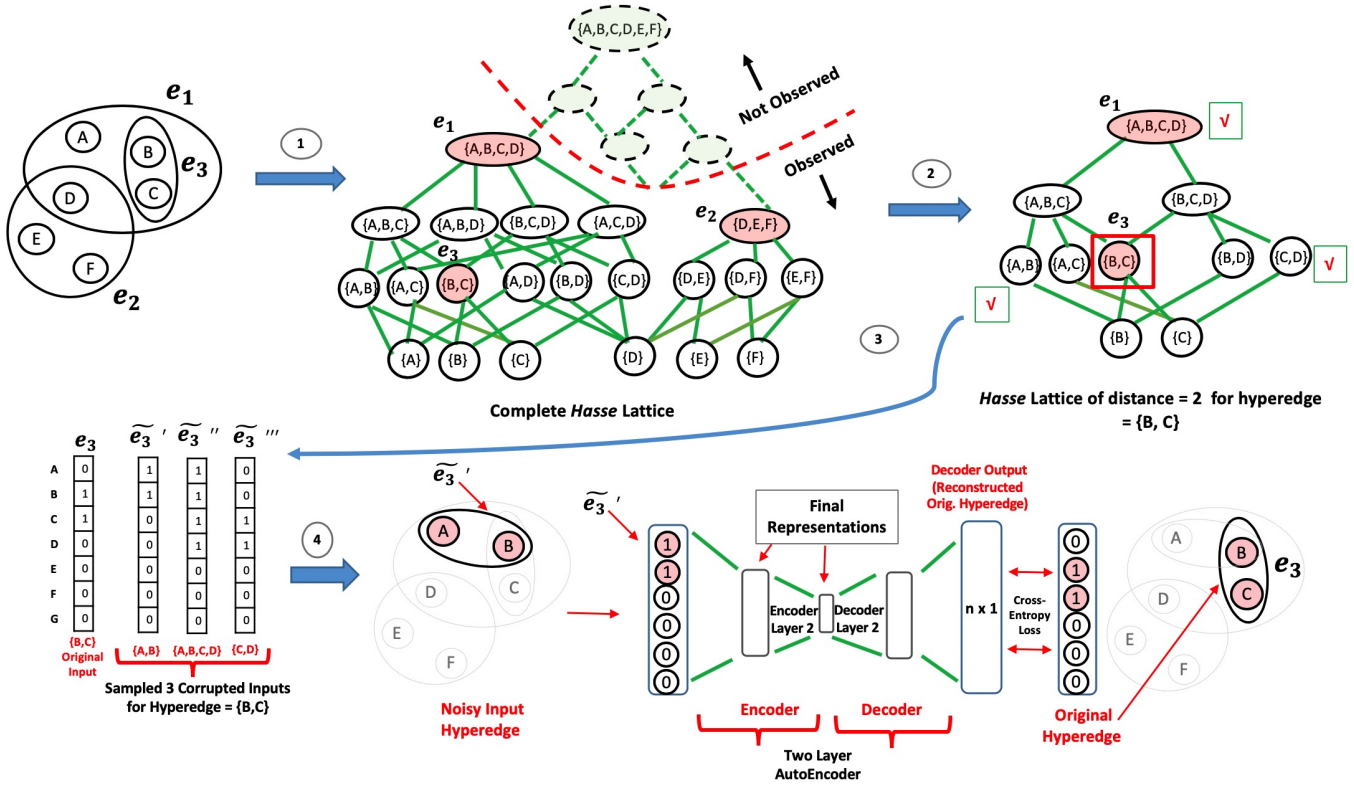

Figure 7.2: For the given hypergraph between six nodes, $V = \{A, B, C, D, E, F\}$ (top-left), we have *Hasse Diagram* (top-middle) with shaded sets, as the hyperedges observed, and not shaded with bold outline sets, are the subset of the observed hyperedges. We also show not shaded with dotted outline on top, which are not observed in the hypergraph, but are possible subsets of V. Note only the nodes lying in the observed region constitute the Hasse diagram in corresponding to the hypergraph in top-left but we consider the *complete Hasse lattice*. For a given hyperedge {B,C} (square box) we then construct the sub-lattice made of hyperedges with distance $h = 2$ from {B,C}. We perform random walk starting from the node corresponding to hyperedge {B,C} and sample $p = 3$ hyperedges (nodes visited by the random walk; shown with a check-mark). Finally, we train the autoencoder to reconstruct the original hyperedge from these p noisy hyperedges.

So far, so good; however, we still have to figure out an effective strategy for generating noisy hyperedges. The hyperedge $\mathbf{e_i}$) is a discreet data and therefore, Gaussian noise, which is the usual choice for real inputs, is not an option. We require some form of discrete noise. One way is to flip (change 0 to 1 and vice versa) a few randomly chosen components of the $\mathbf{e_i}$) binary vector to generate a noisy vector $\tilde{\mathbf{e}}_\mathbf{i}$. Notice the size of $\mathbf{e_i}$) vector for each hyperedge is n, which is the number of vertices in the hypergraph. In most natural hypergraphs, especially social networks, n can be quiet high ranging from thousands to millions or even billions (like Facebook, for example). Therefore, randomly using a discrete noise like *salt-and-pepper*, might not be reasonable, as there is a large number of possible permutations (as size n is large) and not all of them are related. Random addition of 1s or deletion of existing 1s from $\mathbf{e_i}$, amounts to randomly adding or deleting vertices to the hyperedge $\mathbf{e_i}$. This might end up in noisy hyperedges that are completely unrelated to the given original hyperedge ($\mathbf{e_i}$). For example, users (nodes) in a social network from completely different regions of the network suddenly form a group (hyperedge). Such anomalous scenarios rarely happen in practice, and social groups evolve gradually via simple processes (Sharma et al., 2015; 2017).

Rather, we take advantage of the hypergraph structure to guide us in generating this noise systematically. A hypergraph can be defined by its corresponding *hasse diagram or lattice* (see Def. 44; (Sharma et al., 2017)). Remember, in a Hasse diagram is a graph whose nodes are either hyperedges (which have been observed in the past) or their subsets, and the connections are based on common node membership while adhering to cardinality hierarchy. See Figure 7.2) for an example hypergraph (top-left) and corresponding Hasse lattice (top-middle). Hasse diagram, therefore, offers an opportunity to use graph sampling techniques like random walk, to sample noisy hyperedges in a meaningful manner. For an original hyperedge ($\mathbf{e_i}$), we consider the sub-lattice consisting of only those hyperedges that are at a distance h from it in the complete lattice. On this sub-lattice, we sample r hyperedges (nodes in sub-lattice) by performing random walk starting at the given hyperedge's node. For example (see Figure 7.2), for the node corresponding to the hyperedge $e_3 = (A, B)$, we obtain the sub-lattice

(top-right corner) which only contains nodes at a distance $h = 2$, from e_3 node. Our stochastic mapping $q(\tilde{\mathbf{e}}_\mathbf{i}|\mathbf{e}_\mathbf{i})$ is therefore, a random walk on the sub-lattice of hyperedge ($\mathbf{e}_\mathbf{i}$) containing hyperedges at distance h from it. Intuitively, the hyperedges coming within a reasonable distance should affect each other's representations and should have more similar representations.

Note that in the toy example of Figure 7.2, we only extract the sub-lattice only from the *observed region* of the *Complete Hasse Lattice* (as indicated in the top-middle diagram). For example a hyperedge $e_4 = \{B, C, D, E\}$ is in the *not observed* region, yet seems like a hyperedge which is related to all the three hyperedges (e_1, e_2, e_3) and therefore, can perfectly serve as a noisy hyperedge for all three original hyperedges. Unfortunately, the Hasse diagram only represents the observed region of the complete Hasse lattice, and therefore, a random walk based sampling on it misses hyperedges like e_4, which, although a good candidate for noise, but occurs in the unobserved region of the complete Hasse lattice. We, therefore, design a strategy that is capable of sampling both observed as well as the unobserved region of the complete Hasse lattice. To do so, we perform random walk on the weighted clique expanded (WCE) graph, $\mathbf{A_{wc}}$ (see Def. 30), of the hypergraph and generate noise hyperedges from the random walk sampled nodes. Although the random walk is performed on the WCE graph, the noisy hyperedges are sampled from the complete Hasse lattice. We, therefore, refer to the noise generation process as *Hasse Noise Generator*, and it is described in the Algorithm 8.

We start by initializing an empty list in which we will add the training data points (see line 1). Line 2-21, is the main loop where we iterate over each hyperedge $e \in E(N_g)$ in the given hypergraph, N_g, in order to generate noisy hyperedges $\tilde{e}$. Starting from the given hyperedge, e, we perform p_1 random walks in lines 4-11 and aggregate all the nodes visited across the walks in the list Q. For each random walk we pick a start vertex, $v \in e$ from the given hyperedge, e, either randomly or in proportion to the degree, $d(v)$ of the vertex v (see lines 5-9), depending on the parameter o. Starting with this initial vertex we perform a random walk on the WCE graph, $\mathbf{A_{wc}}$. Sitting on some vertex the random-walker chooses a neighbor vertex in proportion to the edge weights

given by the weighted adjacency matrix $\mathbf{A_{wc}}$. Walker then hops to the chosen vertex if total hops in the current walk are less than the parameter p_2 and the chosen vertex is not greater than the maximum distance, p_2, from the initial vertex v. After hopping to the new vertex the walker applies the same logic recursively and eventually returning a list of vertices visited by it. This simple procedure is executed by the $randomWalk()$ function in line 10. Using the list of vertices, Q, sampled via random walks, we generate r noisy hyperedges for the hyperedge e (see lines 12-20). We first decide a cardinality c from the set of input cardinalities $\mathbf{c}$ and their distribution vector $\mathbf{c_p}$. We sample a cardinality $c \in \mathbf{c}$ in proportion to $\mathbf{c_p}(c)$. We then randomly pick c different vertices from the list Q as the noisy hyperedge $\tilde{e}$. Note that Q can have repeated occurrences of the same vertex as it might have been visited multiple times during the p_1 random walks. Therefore, Q carries the information about the topology and those vertices that are in vicinity of hyperedge e are more frequent in Q. We might want to ignore this frequency information by taking the set of unique vertices, $unique(Q)$ in list Q. Parameter q dictates this choice (see lines 14-18). In the end we have r training data points $(\tilde{e}, e)$ for a given hyperedge e. All these data points are added into the training data list Tr (see line 19). Note that the same training data point $(\tilde{e}, e)$ might occur multiple times in Tr and this frequency shall also represent in topological relatedness of the hyperedges, $\tilde{e}$ and e.

7.4 Experiments

7.4.1 Dataset Description

Similar to Chapter 6, in this chapter, we consider the same set of real-world as well as synthetic datasets. For the ease of reading, we are reiterating the dataset description. We make use of five popular real-world datasets from the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml). The five selected datasets have data points whose feature vectors contain mostly boolean-valued features. (From each of the datasets, we removed the very few non-boolean valued features.) We then consider each data point

Algorithm 8 Hasse Noise Generator $(\mathbf{A_{wc}}, N_g, \mathbf{c}, \mathbf{c_p}, o, p_1, p_2, p_3, q, r)$

$Tr \leftarrow emptyList()$
for $e \in E(N_g)$ **do**
  $Q \leftarrow emptyList()$
  for $j = 1$ to p_1 **do**
    if $o =$ {randomly} **then**
      Pick a vertex $v \in e$ randomly
    else if $o =$ {degree} **then**
      Pick a vertex $v \in e$ in proportion to degree
    end if
    $Q \leftarrow Q + randomWalk(\mathbf{A_{wc}}, p_2, p_3)$
  end for
  for $k = 1$ to r **do**
    Sample a cardinality $c \in \mathbf{c}$ using the cardinality distribution vector $\mathbf{c_p}$
    if $q =$ {unique} **then**
      $\tilde{e} \leftarrow$ Choose c vertices from $unique(Q)$ without repetition
    else if $q =$ {frequency} **then**
      $\tilde{e} \leftarrow$ Choose c vertices from Q without repetition
    end if
    $Tr \leftarrow Tr \cup (\tilde{e}, e)$
  end for
end for
return Tr

Data	Hyperedges (m)	Vertices (n)	Max. Cardinality	Avg. Cardinality
zoo	101	15	10	6.53
voter	432	16	13	7.91
autism-child	291	14	13	7.46
autism-adolo	103	14	12	7.59
autism-adult	681	14	13	5.81

Table 7.1: Hypergraph Statistics for various Datasets

(sample) as a hyperedge with features as vertices. All the features (vertices) which have value one for a given sample (hyperedge) are considered vertices of this sample hyperedge. In short, we treat the data matrix (sample-feature mapping) as the hypergraph incidence matrix (hyperedge-vertex mapping). Below we describe the five datasets:

1. **zoo**: In this dataset, there are several animals each described with a set of boolean attributes like, for example, does it have a feather, or is it airborne. There are several classes of animals, and the aim is to classify animals correctly into its class.

2. **voter**: In this, the aim is to classify congressman as democrat versus republican based on 16 key votes, where each vote is boolean (yea or nay). Each congressmen's hyperedge contains only "yay" vertices.

3. **autism-child, autism-adolo, autism-adult**: These three datasets contain psychological evaluation on a cohort of children, adolescents, and adults, respectively, for classification into having ASD disorder or not. Attributes are boolean item responses to behavioral questions. We treat each item (psychological evaluation question) as a vertex, and with each positively responded item (vertex) becomes part of the corresponding individual's hyperedge.

7.4.2 Evaluation Methodology and Experimental Setup

Methods Compared

We compare hyperedge embeddings obtained from our proposed method with several baselines. Both the baseline, as well as proposed approaches, can output hyperedge embeddings in two different ways:

1. Each method can directly output hyperedge embeddings. Our proposed approach gives direct hyperedge embedding when trained using training data generated from the original hypergraph. We refer to these embeddings as **hyperedge2vec** or in short **h2v**. Similarly, we have two baselines that give hyperedge embeddings directly via eigenvalue decomposition of line graph Laplacian (2.47) (herein **e2v-line**) and that of proxy dual hypergraph Laplacian (2.44) (herein **e2v-dual**). Refer to Chapter 2 for details with regards to these Laplacians. We also employ a recently popular node embedding method based on random walks and *skip-gram* model called *node2vec* (Grover & Leskovec, 2016b). We use *node2vec* on the line graph to obtain hyperedge embeddings directly, which we refer herein **n2v-line**.

2. We can use these methods to output vertex embeddings first and then combine the vertex embeddings of the vertices within a hyperedge. We evaluate only two kinds of combinations - summing the vertex embeddings (**sum**) or taking the mean of vertex embeddings (**mean**). (Note the later is hyperedge cardinality dependent.) We refer hyperedge embeddings obtained from our proposed approach in this manner by first obtaining vertex embeddings via training using training data generated via dual hypergraph, as **h2v-dual**. Similarly, for the two baselines, we can first obtain vertex embeddings via eigenvalue decomposition of graph Laplacian (*Gibson* WCE graph (2.22)) and proxy hypergraph Laplacian (2.42), which can then be combined to get hyperedge embeddings referred to as **e2v** and **e2v-hyp**, respectively. Refer Chapter 2 for details with regards to these Laplacians. Also we use *node2vec* on the *Gibson* WCE adjacency matrix (2.21) to obtain vertex

embeddings, which we aggregate to obtain hyperedge embeddings, herein referred to as **n2v**.

We therefore, have two different comparisons. First is between embeddings obtained from **e2v-line**, **e2v-dual** and **n2v-line** with **h2v** embeddings. Second, is between **e2v**, **e2v-hyp** and **n2v** with **h2v-dual**. The later comparison happens further separately for both **sum** and **mean** combining strategies.

Evaluation Tasks and Setup

As described in Section 7.4.1, each dataset has a hypergraph and a corresponding classification task associated with it. We first obtain the hyperedge embeddings for the various datasets. There are two sets of hyper-parameters, one associated with the noise generation, and the other set is the autoencoder hyper-parameters.

Noise Setup: The hyper-parameters specific to noise generation (see Algorithm 8) are the following along with their search spaces– (1) number of noisy hyperedges for a given original hyperedge: $r = [5, 10, 15, 25, 50]$, (2) initial node is picked randomly or in proportion to its degree: $o = \{randomly, degree\}$, (3) total number number of walks: $p_1 = [5, 10, 15, 20]$, (4) walk length (including the initial node): $p_2 = [2, 3, 4, 5, 6, 7]$ and lastly, (5) whether to use node visited frequency to sample it for noisy hyperedge: $q = \{unique, frequency\}$. Grid search was used to find the best parameters. Noise generation also requires two more hyper-parameters: set of cardinalities to sample for noisy hyperedge ($\mathbf{c}$) and the corresponding sampling probabilities ($\mathbf{c_p}$). These can be either made fixed or can be derived from data. We use the cardinality distribution of each dataset to ascertain $\mathbf{c}, \mathbf{c_p}$, while constraining minimum cardinality of at least two ($c \geq 2, \forall c \in \mathbf{c}$).

Autoencoder Setup: The hyper-parameters specific to the proposed autoencoder method includes the learning rate (δ), batch size (bs), number of training epochs (ep) and regularization parameter (β) for l_2-regularization on neural networks weights. These parameters were tuned via grid search over following search space: $\delta = [0.01, 0.001, 0.0001]$,

$bs = [10, 100]$, $ep = [20, 50, 100]$ and $\beta = [0.01, 0.5, 1, 5, 10]$. Again, grid search was performed to fine tune these parameters. The proposed neural network method was implemented using TensorFlow API (Abadi et al., 2015), and the optimizer used for training the neural-net was Adam (Kingma & Ba, 2014).

Embedding Dimensions: Further, the hyper-parameter of embedding dimension (d) which is common to all the graph-based baselines is determined by grid search over: $d = [8, 16, 32, 64]$. For the auto-encoder method (**h2v**) we consider three scenarios: (1) single hidden layer ($L1$) of dimension $d = d_1$; (2) two hidden layers ($L1$ & $L2$) of dimensions d_1, d_2 and we concatenate these embedding to get a single $d = d_1 + d_2$ size embedding; and (3) two hidden layers ($L1$ & $L2$) of dimensions d_1, d_2 and we use the output of $L2$ ($d = d_2$ dimension) as the embedding. We found the best settings with the grid search over: $d_1 = [8, 16, 32, 64]$ and $d_2 = [8, 16, 32, 64]$.

Evaluation Task: The obtained hyperedge embeddings for a given dataset are utilized for the hyperedge classification task associated with the dataset. For each classification task, we perform several evaluation runs. In each run, we randomly choose 30% of hyperedges as the test set and train logistic regression classifier using the remaining 70% training hyperedges. We chose the Area Under Curve (AUC) as the evaluation metric (the higher, the better). We take the average AUC score across five runs as the final AUC. We performed Logistic regression with l_2-norm regularization whose hyper-parameter was chosen by 5-fold cross-validation using a grid search.

7.4.3 Results and Discussion

As detailed in prior section 7.4.2, for each dataset we have two different sets of comparison based on two different ways of obtaining hyperedge embedding: (a) directly obtaining hyperedge embeddings or (b) obtained by aggregating (sum or mean) vertex embeddings. For case (a) we compare **hyperedge2vec** or **h2v** versus **e2v-line**/**e2v-dual**/**n2v-line** and for case (b) compare **hyperedge2vec-dual** or **h2v-dual** versus **e2v**/**e2v-hyp**/**n2v**. Results are reported in Table 7.2 for various datasets. We refer to methods with "**h2v-**" prefix jointly as "**h2v-** methods". Similarly, "**e2v-** methods"

Dataset	Embed. Combination	Eigen Decomp.		Node2vec	Autoencoder
		graph (e2v)	proxy hyp. (e2v-hyp)	graph (n2v)	dual (h2v)
Zoo	Node Embed Sum	0.42	0.40	0.71	0.63
	Node Embed Average	0.60	0.40	0.65	**0.76**
		line graph (e2v-line)	proxy dual (e2v-dual)	line graph (n2v-line)	hypergraph (h2v)
	Only Hyperedge Embed	0.52	0.60	0.43	**0.73**

Dataset	Embed. Combination	Eigen Decomp.		Node2vec	Autoencoder
		graph (e2v)	proxy hyp. (e2v-hyp)	graph (n2v)	dual (h2v)
Voter	Node Embed Sum	0.94	0.94	0.95	**0.96**
	Node Embed Average	0.94	0.94	0.93	**0.96**
		line graph (e2v-line)	proxy dual (e2v-dual)	line graph (n2v-line)	hypergraph (h2v)
	Only Hyperedge Embed	0.94	0.93	0.87	**0.96**

Dataset	Embed. Combination	Eigen Decomp.		Node2vec	Autoencoder
		graph (e2v)	proxy hyp. (e2v-hyp)	graph (n2v)	dual (h2v)
Autism Child	Node Embed Sum	0.98	0.98	0.97	**1.00**
	Node Embed Average	0.73	0.71	0.66	0.72
		line graph (e2v-line)	proxy dual (e2v-dual)	line graph (n2v-line)	hypergraph (h2v)
	Only Hyperedge Embed	0.97	0.96	0.47	**0.97**

Dataset	Embed. Combination	Eigen Decomp.		Node2vec	Autoencoder
		graph (e2v)	proxy hyp. (e2v-hyp)	graph (n2v)	dual (h2v)
Autism Adoloscent	Node Embed Sum	0.89	0.91	0.85	**0.95**
	Node Embed Average	0.65	0.65	0.67	**0.77**
		line graph (e2v-line)	proxy dual (e2v-dual)	line graph (n2v-line)	hypergraph (h2v)
	Only Hyperedge Embed	0.86	0.88	0.52	**0.89**

Dataset	Embed. Combination	Eigen Decomp.		Node2vec	Autoencoder
		graph (e2v)	proxy hyp. (e2v-hyp)	graph (n2v)	dual (h2v)
Autism Adult	Node Embed Sum	0.99	1.00	0.99	**1.00**
	Node Embed Average	0.75	0.76	0.77	**0.80**
		line graph (e2v-line)	proxy dual (e2v-dual)	line graph (n2v-line)	hypergraph (h2v)
	Only Hyperedge Embed	0.98	0.98	0.74	**1.00**

Table 7.2: Classification AUC Scores of Hyperedge2vec compared to baselines

and "**n2v-** methods" for all methods with "**e2v-**" and "**n2v-**" prefix, respectively.

One of our central hypothesis is that embeddings obtained from methods which try to retain the higher-order information within hypergraphs are better than those methods which do not. We observe from Table 7.2 (the best scores for each row are highlighted in bold) that in case of direct hyperedge embeddings as well as node embedding summation cases, autoencoder-based methods (**h2v-** methods) consistently outperform graph-based methods (**e2v-** methods), if not worse. This shows that the non-linearity in autoencoder-based models is, indeed, successfully able to encode and preserve the hyperedge-level contextual information.

Also while comparing between the two vertex-embedding aggregation functions: sum and average, we observe (see "sum" and "average" rows in Table 7.2 across **e2v**, **e2v-hyp** and **h2v**) that in most of the cases hyperedge-embeddings obtained via summation of vertex-embeddings, perform better and in some cases those obtained via averaging are performing poorly, especially for **e2v** as well as autoencoder methods (like autism datasets). This observation again highlights the fact, a critical drawback which we had also encountered in Chapter 6, that the choice of aggregation function in the case of vertex-embedding is not clear. Direct hyperedge-embedding methods, on the other hand, are not subject to any such choices.

Lastly, we note that the tensor-based methods in the previous chapter, suffer from sparsity issues like when the number of vertices is much less than hyperedges, and also allow for a limited width of cardinality spectrum. This is not the case with the autoencoder methods proposed in this chapter. Autoencoders are robust and can handle any cardinality hyperedges in the original hypergraph, as well as it's dual. Therefore, it's free from degree and cardinality distributions.

However, there is no free lunch. The main challenge in the case of autoencoder methods is the design of an effective noisy hyperedge sampling strategy. Although, the strategy proposed in Algorithm 8, is relatively general and tunes parameters like cardinality distribution $\mathbf{c_p}$ according to the dataset. But there are several other random walk parameters like the length of the walk, choice of the initial node in a hyperedge, and

the maximum distance of the walk, that needs knowledge of the topology of hypergraph data. This leads to several intuitive questions like what kinds of datasets and their hypergraph properties (avg. degree, size, and others) affect the choice of noise strategy? Also, in several cases, we find the graph-based methods performing close to autoencoder methods. Therefore, a natural question would we prefer working using graph methods, and autoencoder methods might be overkill? It's worth mentioning that training neural networks can often be cumbersome, especially with several hyper-parameters to tune and various architectural choices.

These questions require both theoretical understanding as well as extensive experimentation using an enormous variety of data, and hence, we would like to take this as a separate future work. The focus of this work is to introduce higher-order information retaining methodology by leveraging the non-linearity of neural networks to model the higher-order information within hypergraphs.

7.5 Related Works

Traditional unsupervised graph embedding methods such as spectral clustering for graphs (Belkin et al., 2006; Zhou et al., 2004; Zhu et al., 2003) have been extended to hypergraphs (Zhou et al., 2006b). However, these embeddings focus on learning representations for nodes, and not for hyperedges. Agarwal et al. (2006a) has criticized that such representations can be learned by constructing graphs, which are proxies for the hypergraph structure. Recently, the focus has shifted to learn node-embeddings via methods inspired by skip-gram model Le & Mikolov (2014). Attempts along these lines (Grover & Leskovec, 2016a; Perozzi et al., 2014; Tang et al., 2015b) argues that language models have a ready-made context in the form of sentences or paragraphs to train the model, which are not available in networks, and therefore, they propose different ways to generate this context via random-walks. By contrast, we focus on networks where the context is already present, e.g., collaboration networks where collaborative teams are hyperedges or language hyper-networks where sentences are hyperedges.

Another famous line of work is based on neural networks for graph-structured data. Within this one stream is based on recurrent neural network (RNN) generalization for graph data (Gori et al.; Scarselli et al., 2009) which was simplified recently Li et al. (2015). Other stream is based on generalization of convolutional neural networks (CNN) to a graph setting (Bruna et al., 2013) which has recently been extend to Defferrard et al. (2016) fast localized convolution. Atwood & Towsley (2015) proposed a diffusion based CNN but their approach is $O(n^2)$ in complexity. Another interesting work is that of Niepert et al. (2016), where they are inspired by the original image convolutions itself and convert the graph into sequences for 1-D convolutions. Until recently these CNN-based approaches were not scalable for a large-scale network setting, which some recent has alleviated by Kipf & Welling (2016) where they apply it to a semi-supervised setting on large scale graphs.

However, all these works on graph-based neural networks, which we described so far, are designed for graphs and aim to either learn node embeddings or designed for tasks that require the entire graph's embedding. We are unaware of any work within large-scale network analysis that considers embeddings at the hyperedge-level. In contrast, in the natural language processing (NLP) literature, methods for learning embeddings for higher-order "sequential" structures like sentences and paragraphs have been proposed (Kalchbrenner et al., 2014; Le & Mikolov, 2014). Also recently, there has been an interest in modeling "set-like" structures within the deep-learning community (Rezatofighi et al., 2016; Vinyals et al., 2016). However, they do not consider the hypergraph structure between the sets, and therefore, do not model hypergraphs in a principled manner.

7.6 Conclusion

This chapter revisits the problem of hyperedge embedding, but unlike the tensor-based interpretable model in the previous chapter, here we tried to capture this hyperedge-level information by leveraging the property of neural networks to approximate arbitrarily complex functions. Inspired by the image auto-encoders from computer vision,

we consider the hyperedge as the primary entity to be encoded and develop de-noising hyperedge auto-encoders. We harness the hypergraph structure and use it to sample meaningful noisy hyperedges for a given original hyperedge. We show that our method outperforms several graph-based baselines in terms of accuracy. This shows that the combination of hypergraph based noise and non-linearity in autoencoder-based models is, indeed, successfully able to encode and preserve the hyperedge-level contextual information.

Chapter 8

Summary

This thesis proposed the **Hypergraph Analytics Framework**, under which we have modeled group structured data as well as studied group phenomena from the lens of Hypergraphs. In Chapter 1, we started by providing a taxonomy for group-structured data and develop a set of group abstractions for them, which clearly defined the various group relationship objects we aim to study and model. We then moved to the important task of analyzing as well as proposing various categories of hypergraph models for these group abstractions. A detailed overview of these categories, along with their data structures, was provided in Chapter 2. Different models primarily vary by the amount of higher-order information they can retain. Various problems studied in this thesis have employed these models, where the choice of model has been dictated by the type of group data as well as the problem at hand. In particular, we dealt with two core classes of problems pertaining to hypergraphs. First, is that of spatial analysis: conducting inference or learning probabilities over hypergraph structures. This was taken up in the first part of the thesis consisting of Chapters 3, 4 and 5, which are dedicated to various kind of **inference mechanisms on hypergraph** structures. Second is that of spectral analysis: compressing these higher-order hypergraph structures to low dimension embedding spaces. This was the second part of thesis consisting of Chapters 6 and 7, which are concerned with the **hypergraph compression techniques**.

In the first half, we performed the *Spatial Analysis* of the hypergraph by dividing the problem of inference on the hypergraph structure into sub-problems. First was that of predicting the likelihood of hyperedges in the observed hypergraph structures – ***old hyperedge prediction*** – given the observed hypergraph. This problem was addressed in Chapter 3 where the observed hypergraph structures is modelled using a *simplicial complex* and corresponding *Hasse lattice* structure. Second was that of assigning probabilities to hyperedges in the unobserved hypergraph structure – ***new hyperedge prediction*** – again given the observed hypergraph. Chapter 4 addressed this second problem by employing a simple and elegant approach to incremental sampling of new hyperedges. Various graph, as well as hypergraph model based accretion prediction methods, are proposed, which output probability of the sampled hyperedges. Overall these two chapters address the hypergraph evolution problem by addressing sub-problems of hyperedge stability, as well as increase or decrease in cardinality prediction. However, both these chapters are limited to static group data only and, therefore, are performing hypergraph evolution prediction by *cross-sectional analysis*. Chapter 5, therefore, extends our study of hypergraph evolution to temporal group data and performed *longitudinal analysis*. Here, temporal groups were modeled using higher-order matrices, also called *tensors*, which can model time as another dimension. The task of ***temporal hyperedge prediction*** was then carried out using tensor decomposition techniques. Overall, interlacing the Chapters 3, 4 and 5, we reach a complete recipe of modeling and inference over the hypergraph structure for both cross-sectional and longitudinal analysis.

The second part of the thesis, while complimenting the first half, dealt with the *Spectral Analysis* of hypergraph by developing techniques to compress the hypergraph topology to lower-dimensional latent space. Considering hyperedges as first-class citizens, we have chiefly considered hyperedge compression or *hyperedge embeddings*. We examined two different embedding approaches in Chapters 6 and 7. In Chapter 6 we developed an algebraic model for hyperedge embeddings for general (non-uniform) hypergraphs. Higher-order symmetric tensors are used in order to retain the hyperedge-

level information directly, and the idea of joint decomposition of various cardinality tensors is introduced to learn embeddings. The notion of *dual tensors* is also proposed in order to obtain hyperedge embeddings directly. Chapter 7 revisited the problem of hyperedge embedding, but unlike the interpretable tensor models in the Chapter 6, we considered neural network-based black-box models with the promise of non-linearity in neural networks to capture the hyperedge context information. Inspired by the image auto-encoders from computer vision, we developed hyperedge denoising auto-encoders, which harness the Hasse diagram of the hypergraph to sample meaningful noisy hyperedges. The proposed hyperedge embeddings, in both Chapters 6 and 7, are observed to be performing better than those obtained from various graph-based baselines.

Chapter 9

Future Directions

Several exciting ideas have naturally emerged from the research conducted within this thesis. Some are extensions of the work presented in the thesis to special cases, of which some research is under progression. Also, some are newer ideas that have appeared and require a separate investigation. In this chapter, we would take a moment to point out some of these valuable future perspectives.

Hyperedge Prediction

In the first half of the thesis, we focused on the problem of hyperedge prediction. We proposed various hypergraph based topological or algebraic methods for addressing this problem. Some of these methods, especially those we encountered in Chapter 3, were based on a learning hyperedge probability scores based on semi-supervised learning. The algorithms developed to diffuse the scores from the observed (sub)hyperedges nodes in the Hasse diagram to the unobserved (sub)hyperedge nodes. We observe an overwhelming similarity in this setup to that of language models based on smoothing techniques (Chen & Goodman, 1999). Each c-cardinality (sub)hyperedge correspond to c-gram in text data, and the aim is to ascertain the probabilities of c-grams gave the frequency of occurrence. The difference is that the c-grams are sequences, unlike (sub)hyperedges, which are sets. Therefore, we have already started working on ideas

for generalizing the ideas of sequence-based language models to the sets in the hypergraph. We refer to this generalization as *Hasse smoothing* which where we leverage the smoothing techniques and apply them to the hierarchy dictated by the Hasse diagram. Here we would be interested in how the interpretable smoothing approaches contrast with diffusion-based semi-supervised approaches, which we have developed, with regards to performance.

Methods we have developed so far in the first half of the thesis Chapters 3-5 can be used to predict hyperedges (and their subhyperedges) present in the input hypergraph. The biggest hurdle in predicting the general hyperedges is enumerating them and including them in the input hypergraph of the hyperedge prediction algorithm. The more groups we enumerate, the larger the input hypergraph, which requires more memory and also affects the run-time.

One way to address this is by developing a clever hyperedge sampling technique that enumerated the most relevant hyperedges, which are, in some sense, the best candidates worthy of assigning probability or, in other terms, have a strong potential of becoming a future group. We have already seen a sampling strategy that we had devised in Chapter 7 in the form of the Hasse based noisy hyperedge generation. However, we think that hyperedge sampling is an essential direction with much potential to be pursued as a research project in its own right.

Once we have the enumerated hyperedges, we do not necessarily have to work using algorithms that use the entire input hypergraph in memory. We can think of two different ways to go about this. First is working with vertex embeddings, and developing functions that aggregate the vertex embeddings to output hyperedge probability. This was the primary motivation for the hypergraph compression techniques developed in the second half of the thesis. In this work, we have employed some basic vertex embedding aggregation functions like averaging, summation, or entropy. However, we think that exploring more functions and possibly developing algorithms that learn such functions automatically, is a critical analysis direction which requires an independent effort. The second approach can be combining use functions that combine the edge level probabili-

ties to output hyperedge level probabilities. When it comes to distribution and exciting options, explore the use of the rich *copula theory* (Nelsen, 2007) where the joint distribution is achieved from marginal distributions via copulas. In our setting, it would amount to treating the edge probabilities as marginals and the hyperedge probabilities as the higher-order joint probabilities achieved via gelling the marginals via copula functions.

Although explicit enumeration via sampling is one way, we think a more holistic approach is where the algorithm inherently prunes the search space and restricts the search to the most promising regions of the lattice. Search space pruning is prevalent in machine learning and statistics, where feature selection is performed using sparsity inducing norms that prune the high dimensional feature space (Hastie et al., 2015). Specifically, we are interested in structured norms that induce sparsity concerning any previous structural information available via domain knowledge or present in data (Jenatton et al., 2011). For example, *group lasso* norm selects either all or none of the features from a given predefined set of overlapping or non-overlapping groups (Yuan & Lin, 2006). In our group evolution problem, we have the rich hypergraph structure, which can be leveraged for inducing sparsity and selecting the most informative regions in the *hasse lattice*. Notice that unlike feature selection, which is the primary objective of most sparse techniques, we would aim to select groups/clusters of vertices among which future groups are most likely to be formed. We have to perform this constrained search while learning the probabilities for the group. For this, we treat that learning probability of a k size group as the task of *k-way tensor* completion (Qi, 2005). Note that these sparsity norms, we just mentioned, have been applied as constraints (regularization) to the various supervised learning cost objectives like matrix completion (Kim et al., 2012) and recently to tensor completion tasks (Tomioka & Suzuki, 2013). We would, therefore, like to develop algorithms that perform k-way tensor completion regularized by novel sparsity norms guided by the hypergraph/hasse lattice structure, to perform the constrained search.

Although the hyperedge prediction models proposed in the first part of this thesis are primarily topology-based or algebraic, there can be other kinds of models that

do not directly use the topology or use it in part along with other techniques. In group data from domains where the entities themselves are intelligent agents like social groups, one can build models where the hypergraph topology is an outcome of the group interactions of individual agents. In such a scenario, one can leverage advanced techniques like reinforcement learning (Sutton & Barto, 2018). It would be interesting to model the social group formation as an outcome of a joint decision between a set of agents and also based on the group-level as well as agent-level rewards. In the case of groups, the games are that of group-level cooperation between agents and ideas of cooperative game theory (Chalkiadakis et al., 2011) start coming into the picture. But the most exciting aspect for us would be to explore the dynamics of the properties of the emerging hypergraph topology as well as if/how to leverage the new hypergraph topology to design the actions or rewards.

In this thesis, we have restricted ourselves to undirected hypergraphs. A self-evident and significant direction would be to generalize the methodology developed in this work to *directed hypergraphs*. We think that at the theoretical level, some of the methods for undirected hypergraphs might generalize to directed hypergraphs, but the kind of applications that motivate the use of directed hypergraphs are very different from those for the undirected case. Therefore, building machine learning models for directed hypergraphs necessitates an independent research route. Here, we present a research problem involving directed hypergraphs that we are currently investigating. The problem of finding dominant content flow trends and predicting flow in networks is vital in the context of online social, communication (Mori et al., 2005), and multimedia networks. Typically such kinds of flows are studied in terms of node-to-node information flow. We want to generalize this to a setting where information flow is happening at the level of social groups or teams. Examples include email communication where the sender sends an email to multiple participants; a research collaboration happening between a group of researchers such that part of the group is learning or getting influenced on a topic or idea which the other part of the group who might be more specialized in that topic or worked on that idea before. Such kind of influence spread within groups can be modeled

as a directed hyperedge. A directed hyper-path then represents a flow of information between groups over the path. We aim to leverage this model to address two subproblems. First, modeling group information flow in terms of directed hypergraphs offers the opportunity to build hypergraph mining algorithms to mine such group information flow hyperpaths, studying their properties, and characterizing them. Another problem is to predict the evolution of this information flow hyperpaths. For this, we aim to leverage the fact that hyper-path evolution is similar to a sequence prediction problem given past sub-sequence and use Recurrent Neural Networks (RNN) (Elman, 1990) to model variable-length sequential information.

Hypergraph Emebeddings

In the second part of the thesis, we performed the spectral analysis of the hypergraphs where our focus was to develop methods to learn embeddings for both vertices as well as hyperedges while preserving both local hyperedge-level information as well as global hypergraph topology information. In Chapter 6, we developed two separate algorithms of hypergraph tensor decomposition and dual tensor decomposition that output vertex and hyperedge embeddings, respectively. However, we learn each embedding separately. It would be worthwhile to perform simultaneous dual and hypergraph tensor decomposition and possibly also to learn the mapping between the hyperedge and vertex embeddings.

The methods developed in Chapter 6 leverage the connection between hypergraph and symmetric tensors and build tensor decomposition based embeddings techniques that perform successfully on real data. Spectral hypergraph theory (Cooper & Dutle, 2012) was the primary motivation for the leveraged connection, so, it would be worthwhile to perform a more in-depth analysis via this connection by studying the relationship of the obtained embeddings to spectral properties of adjacency tensors or topological properties of the hypergraph. It would also be of interest to understand the relationship between the embeddings and those obtained via another object that we

have not yet explored: the higher-order tensor Laplacians (Qi, 2014).

In Chapter 7, inspired by image autoencoders, we developed autoencoder for hyperedges, providing a principled treatment to hyperedges. The denoising autoencoders that we have proposed have fully-connected dense layers. However, if the data has a spatial structure, then convolutional neural networks (CNN) are suggested which exploit spatially local correlation by enforcing a sparse local connectivity pattern between neurons of adjacent layers: each neuron is connected to only a small region of the input volume. CNNs have also been successfully used within image autoencoders (Makhzani & Frey, 2015), deconvolutional networks (Zeiler et al., 2010) and convolutional predictive sparse decomposition (PSD) (Kavukcuoglu et al., 2010; Sermanet et al., 2013). Inspired from these image-based convolutional autoencoders and leveraging the recently proposed graph convolutional networks (Bruna et al., 2013; Defferrard et al., 2016; Kipf & Welling, 2016) we propose the idea of *hyperedge convolutional autoencoder*. It would be interesting to see the performance of these hyperedge convolutional autoencoders, to those of the Hasse denoising autoencoder we have proposed in this thesis. We are currently pursuing this investigation.

Also, we notice that noisy hyperedge generation is a critical component of the autoencoder proposed in Chapter 7. It would also be worthwhile to develop other kinds of noise generation schemes, possibly with larger sampled context and studying properties of noise across different kinds of datasets and their impact on performance.

The hyperedge autoencoder proposed in Chapter 7 has a strong resemblance to the emerging area of metric learning (Kulis et al., 2013) as both techniques aim at learning embeddings and also develop a variety of schemes to prepare training and testing samples, which in our case correspond to the noisy and original hyperedges. However, metric learning uses cost functions based on the embeddings vectors, unlike the denoising loss function. Nonetheless, the resemblance is encouraging enough to perform a proper investigation about this connection.

Theoretical Investigations

The central message of this thesis is that the importance of hypergraph resides in their ability to represent the hyperedge-level functions. If the task requires modeling such functions, then hypergraphs can be leveraged. Also, a good model for hypergraph is the one that maximizes the retention of the hyperedge-level function. One of the core outcomes of this thesis is the intuitive realization that although the hyperedge-level functions are of prime importance, their importance is so far as to their relation to vertex-level functions. One of the invaluable to pursue direction would then be to see how the mapping of these functions are inter-related across the variety of data and tasks. An incessant intuition is that the variation in hyperedge-level function is similar or in accordance with the vertex-level function. In such cases, hypergraphs might not be a suitable model, versus a case where the hyperedge-level function is very different than the vertex-level function. All this analysis involving the study of vertex and hyperedge level functions, thus, demands a strong and independent research investigation.

Intuitively, the above discussion also seems to lead to the idea of generalizing the signal processing over graphs (Hammond et al., 2011; Shuman et al., 2013) to hypergraphs and designing of wavelets over hypergraphs rather than graph wavelets (Rustamov & Guibas, 2013). There is a dearth of literature in either of these directions and are promising for future research. For example, theoretically, convolution for graph (Hammond et al., 2011) is based on the spectrum of graph Laplacian, which operates on function over vertices. Operators that measures variations on functions defined on p-sized subsets (hyperedges) of the vertex set are the higher-order Laplacians (Forman, 2003) which operate on what is called in discrete differential geometry as p-forms (Rosenberg, 1997). Unlike the proxy graphs pointed in Chapter 2, these higher-order Laplacians model the hypergraph in a true sense. Following the lines of Hammond et al. (2011), we would, therefore, like to take the endeavor of modeling wavelets on hypergraphs based on spectral hypergraph theory (which has gotten reasonably advanced in past decade or so, see (Cooper & Dutle, 2012; Qi, 2005)). Further, build convolutions that convolve

measures at hyperedge-level and employ them to develop more robust algorithms for hyperedge prediction.

Another useful direction would also be to develop synthetic labeled data where the hyperedge and vertex labels are related in various ways. Such datasets would be crucial for analyzing functions or signals over hypergraph. A good starting point would be the ideas such as hypergraph stochastic block models (Ghoshdastidar & Dukkipati, 2015), which generalize the graph stochastic block models to hypergraph. However, they still focus on vertex functions in a hypergraph setting as their focus is vertex partitioning. It would be intriguing to develop and analyze synthetic data based on both hyperedge labels and vertex labels.

Bibliography

Alibaba record sale, 2017. https://www.techinasia.com/china-singles-day-2017-record-spending.

Google knowledge graph. https://developers.google.com/knowledge-graph/.

Pubmed data. https://www.ncbi.nlm.nih.gov/pubmed/.

Guild play in mmogs: Rethinking common group dynamics models. In *Social Informatics: Third International Conference, SocInfo 2011, Singapore, October 6-8, 2011. Proceedings*, pp. 145–152, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-24704-0. doi: 10.1007/978-3-642-24704-0_19.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. 2015.

Sameer Agarwal, Kristin Branson, and Serge Belongie. Higher order learning with graphs. In *Proceedings of the 23rd international conference on Machine learning*, pp. 17–24. ACM, 2006a.

Sameer Agarwal, Kristin Branson, and Serge Belongie. Higher order learning with graphs. In *Proceedings of the 23rd international conference on Machine learning*, pp. 17–24. ACM, 2006b.

Charu Aggarwal and Karthik Subbian. Evolutionary network analysis: A survey. *ACM Computing Surveys (CSUR)*, 47(1):10, 2014.

Charu C Aggarwal and Jiawei Han. *Frequent Pattern Mining*. Springer, 2014.

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Acm sigmod record*, volume 22, pp. 207–216. ACM, 1993.

Muhammad Aurangzeb Ahmad, Zoheb Borbora, Cuihua Shen, Jaideep Srivastava, and Dmitri Williams. *Guild play in mMOGs: Rethinking common group dynamics models*. Springer, 2011.

Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*, pp. 37–48. ACM, 2013.

Iftekhar Ahmed, Channing Brown, Andrew Pilny, Dora Cai, Yannick Atouba Ada, and Marshall Scott Poole. Identification of groups in online environments: The twist and turns of grouping groups. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pp. 629–632. IEEE, 2011.

Iftekhar Ahmed, Amogh Mahapatra, Marshall Scott Poole, Jaideep Srivastava, and Channing Brown. Identifying a typology of players based on longitudinal game data. In *Predicting Real World Behaviors from Virtual World Data*, pp. 103–115. Springer, 2014.

Mohammad Al Hasan and Mohammed J Zaki. A survey of link prediction in social networks. In *Social network data analytics*, pp. 243–275. Springer, 2011.

Hamidreza Alvari, Alireza Hajibagheri, and Gita Sukthankar. Community detection in dynamic social networks: A game-theoretic approach.

Hamidreza Alvari, Sattar Hashemi, and Ali Hamzeh. Detecting overlapping communities in social networks by game theory and structural equivalence concept. In *Artificial Intelligence and Computational Intelligence*, pp. 620–630. Springer, 2011.

Deborah G Ancona and David F Caldwell. Bridging the boundary: External activity and performance in organizational teams. *Administrative science quarterly*, pp. 634–665, 1992.

Katsuhiko Ariga, Jonathan P Hill, Michael V Lee, Ajayan Vinu, Richard Charvet, and Somobrata Acharya. Challenges and breakthroughs in recent research on self-assembly. *Science and technology of advanced materials*, 9(1):014109, 2008.

Armen S Asratian. *Bipartite graphs and their applications*. Number 131. Cambridge University Press, 1998.

James Atwood and Don Towsley. Search-convolutional neural networks. *arXiv preprint arXiv:1511.02136*, 2015.

John C Avise. *Evolutionary pathways in nature: a phylogenetic approach*. Cambridge University Press, 2006.

Ivo Baar, Lukas Hübner, Peter Oettig, Adrian Zapletal, Sebastian Schlag, Alexandros Stamatakis, and Benoit Morel. Data distribution for phylogenetic inference with site repeats via judicious hypergraph partitioning. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 175–184. IEEE, 2019.

Brett W Bader and Tamara G Kolda. Efficient matlab computations with sparse and factored tensors. *SIAM Journal on Scientific Computing*, 30(1):205–231, 2007.

Grey Ballard, Tamara Kolda, and Todd Plantenga. Efficiently computing tensor eigenvalues on a gpu. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pp. 1340–1348. IEEE, 2011.

Albert-Laszlo Barabâsi, Hawoong Jeong, Zoltan Néda, Erzsebet Ravasz, Andras Schubert, and Tamas Vicsek. Evolution of the social network of scientific collaborations. *Physica A: Statistical Mechanics and its Applications*, 311(3):590–614, 2002.

Anthony F Bartholomay. Molecular set theory: A mathematical representation for chemical reaction mechanisms. *Bulletin of Mathematical Biology*, 22(3):285–307, 1960.

Stephen A Beebe and John T Masterson. Communication in small groups: principles and practice, 2009.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, volume 14, pp. 585–591, 2001.

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(Nov):2399–2434, 2006.

Yoshua Bengio and Samy Bengio. Modeling high-dimensional discrete data with multilayer neural networks. In *Advances in Neural Information Processing Systems*, pp. 400–406, 2000.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

C. Berge. *Graphs and hypergraphs*, volume 6. Elsevier, 1976.

Claude Berge. *Hypergraphs: combinatorics of finite sets*, volume 45. Elsevier, 1984.

Claude Berge and Edward Minieka. *Graphs and hypergraphs*. North-Holland publishing company Amsterdam, 1973.

Wai Fong Boh, Yuqing Ren, Sara Kiesler, and Robert Bussjaeger. Expertise and collaboration in the geographically dispersed organization. *Organization Science*, 18(4): 595–612, 2007.

Marianna Bolla. Spectra, euclidean representations and clusterings of hypergraphs. *Discrete Mathematics*, 117(1-3):19–39, 1993.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pp. 1247–1250, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-102-6. doi: 10.1145/1376616.1376746. URL http://doi.acm.org/10.1145/1376616.1376746.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pp. 2787–2795, 2013.

Alain Bretto. Hypergraph theory. *An introduction. Mathematical Engineering. Cham: Springer*, 2013.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

Jiajun Bu, Shulong Tan, Chun Chen, Can Wang, Hao Wu, Lijun Zhang, and Xiaofei He. Music recommendation by unified hypergraph: combining social media information and music content. In *Proceedings of the international conference on Multimedia*, pp. 391–400. ACM, 2010.

Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques and applications. *arXiv preprint arXiv:1709.07604*, 2017.

Georgios Chalkiadakis, Edith Elkind, and Michael Wooldridge. *Computational aspects of cooperative game theory*. Morgan & Claypool Publishers, 2011.

Chien-Hsun Chen, Chuen-Tsai Sun, and Jilung Hsieh. Player guild dynamics and evolution in massively multiplayer online games. *CyberPsychology & Behavior*, 11(3): 293–301, 2008.

Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394, 1999.

George Cheney, Lars Thøger Christensen, Theodore E Zorn Jr, and Shiv Ganesh. *Organizational communication in an age of globalization: Issues, reflections, practices*. Waveland Press, 2010.

Evangelia Christakopoulou and George Karypis. Hoslim: higher-order sparse linear method for top-n recommender systems. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 38–49. Springer, 2014.

Janara Christensen, Stephen Soderland, Oren Etzioni, et al. An analysis of open information extraction based on semantic role labeling. In *Proceedings of the sixth international conference on Knowledge capture*, pp. 113–120. ACM, 2011.

Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

Fan RK Chung, Ronald L Graham, Peter Frankl, and James B Shearer. Some intersection theorems for ordered sets and graphs. *Journal of Combinatorial Theory, Series A*, 43(1):23–37, 1986.

Taejoong Chung, Jinyoung Han, Daejin Choi, Taekyoung Ted Kwon, Huy Kang Kim, and Yanghee Choi. Unveiling group characteristics in online social games: A socioeconomic analysis. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14. International World Wide Web Conferences Steering Committee, 2014.

James S Coleman. Social capital in the creation of human capital. *American journal of sociology*, pp. S95–S120, 1988.

Pierre Comon, Gene Golub, Lek-Heng Lim, and Bernard Mourrain. Symmetric tensors and symmetric tensor rank. *SIAM Journal on Matrix Analysis and Applications*, 30 (3):1254–1279, 2008.

Noshir Contractor. Some assembly required: leveraging web science to understand and enable team assembly. *Phil. Trans. R. Soc. A*, 371(1987):20120385, 2013.

Joshua Cooper and Aaron Dutle. Spectra of uniform hypergraphs. *Linear Algebra and its Applications*, 436(9):3268–3292, 2012.

Trevor F Cox and Michael AA Cox. *Multidimensional scaling*. CRC press, 2000.

Josh Daspit, C Justice Tillman, Nancy G Boyd, and Victoria Mckee. Cross-functional team effectiveness: An examination of internal team environment, shared leadership, and cohesion influences. *Team Performance Management: An International Journal*, 19(1/2):34–56, 2013.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pp. 3844–3852, 2016.

Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

Inderjit S Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 269–274. ACM, 2001.

Daniel M Dunlavy, Tamara G Kolda, and Evrim Acar. Temporal link prediction using matrix and tensor factorizations. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(2):10, 2011.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Ernesto Estrada and Juan A Rodriguez-Velazquez. Complex networks as hypergraphs. *arXiv preprint physics/0505137*, 2005.

R. Fagin. Degrees of acyclicity for hypergraphs and relational database schemes. *Journal of the ACM (JACM)*, 30(3):514–550, 1983a.

Ronald Fagin. Degrees of acyclicity for hypergraphs and relational database schemes. *Journal of the ACM (JACM)*, 30(3):514–550, 1983b.

Katherine Faust. Centrality in affiliation networks. *Social networks*, 19(2):157–191, 1997.

Scott L Feld. The focused organization of social ties. *American journal of sociology*, 86 (5):1015–1035, 1981.

Valeria Fionda. Networks in biology. 2019.

Jason Flannick. *Algorithms for biological network alignment*. ProQuest, 2008.

Robin Forman. Bochner's method for cell complexes and combinatorial ricci curvature. *Discrete and Computational Geometry*, 29(3):323–374, 2003.

Julien Gagneur, Roland Krause, Tewis Bouwmeester, and Georg Casari. Modular decomposition of protein-protein interaction networks. *Genome biology*, 5(8):R57, 2004.

Shenghua Gao, Ivor Wai-Hung Tsang, and Liang-Tien Chia. Laplacian sparse coding, hypergraph laplacian sparse coding, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):92–104, 2013.

Anne-Claude Gavin, Markus Bösche, Roland Krause, Paola Grandi, Martina Marzioch, Andreas Bauer, Jörg Schultz, Jens M Rick, Anne-Marie Michon, Cristina-Maria Cruciat, et al. Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, 415(6868):141–147, 2002.

Gourab Ghoshal, Vinko Zlatić, Guido Caldarelli, and MEJ Newman. Random hypergraphs and their applications. *Physical Review E*, 79(6):066118, 2009.

Debarghya Ghoshdastidar and Ambedkar Dukkipati. A provable generalized tensor spectral method for uniform hypergraph partitioning. In *International Conference on Machine Learning*, pp. 400–409, 2015.

Debarghya Ghoshdastidar and Ambedkar Dukkipati. Uniform hypergraph partitioning: Provable tensor methods and sampling techniques. *arXiv preprint arXiv:1602.06516*, 2016.

David Gibson, Jon Kleinberg, and Prabhakar Raghavan. Clustering categorical data: An approach based on dynamical systems. *The VLDB Journal*, 8(3-4):222–236, 2000.

Gene H Golub and Christian Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, 1970.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 2, pp. 729–734. IEEE.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pp. 855–864, 2016a. doi: 10.1145/2939672.2939754.

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

Jean-Loup Guillaume and Matthieu Latapy. Bipartite structure of all complex networks. *Information processing letters*, 90(5):215–221, 2004.

Roger Guimerà, Brian Uzzi, Jarrett Spiro, and Luis A Nunes Amaral. Team assembly mechanisms determine collaboration network structure and team performance. *Science*, 308(5722):697–702, 2005.

Jungpil Hahn, Jae Yun Moon, and Chen Zhang. Emergence of new project teams from open source software developer networks: Impact of prior collaboration ties. *Information Systems Research*, 19(3):369–391, 2008.

David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2): 129–150, 2011.

E.H. Han, G. Karypis, V. Kumar, and B. Mobasher. *Clustering based on association rule hypergraphs*. University of Minnesota, Department of Computer Science, 1997.

Eui-Hong Han, George Karypis, Vipin Kumar, and Bamshad Mobasher. Hypergraph based clustering in high-dimensional data sets: A summary of results. *IEEE Data Eng. Bull.*, 21(1):15–22, 1998.

Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. CRC press, 2015.

Jonathan Hayes and Claudio Gutierrez. Bipartite graphs as intermediate model for rdf. In *In Proc. of the 3th Int. Semantic Web Conference (ISWC), number 3298 in LNCS*, pp. 47–61. Springer-Verlag, 2004.

Pamela J Hinds and Mark Mortensen. Understanding conflict in geographically distributed teams: The moderating effects of shared identity, shared context, and spontaneous communication. *Organization science*, 16(3):290–307, 2005.

TaeHyun Hwang, Ze Tian, Rui Kuangy, and Jean-Pierre Kocher. Learning on weighted hypergraphs to integrate protein interactions and gene expressions for cancer outcome prediction. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pp. 293–302. IEEE, 2008.

Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Tushar Chandra, and Craig Boutilier. Slateq: A tractable decomposition for reinforcement learning with recommendation sets. 2019a.

Eugene Ie, Vihan Jain, Jing Wang, Sanmit Navrekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Morgane Lustman, Vince Gatto, Paul Covington, et al. Reinforcement learning for slate-based recommender systems: A tractable decomposition and practical methodology. *arXiv preprint arXiv:1905.12767*, 2019b.

Rodolphe Jenatton, Jean-Yves Audibert, and Francis Bach. Structured variable selection with sparsity-inducing norms. *Journal of Machine Learning Research*, 12(Oct): 2777–2824, 2011.

Neil F Johnson, Chen Xu, Zhenyuan Zhao, Nicolas Ducheneaut, Nicholas Yee, George Tita, and Pak Ming Hui. Human group formation in online guilds and offline gangs driven by a common team dynamic. *Physical Review E*, 79(6):066117, 2009.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

Ah Reum Kang, Juyong Park, and Huy Kang Kim. Loyalty or profit? early evolutionary dynamics of online game groups. In *Network and Systems Support for Games (NetGames), 2013 12th Annual Workshop on*, pp. 1–6. IEEE, 2013.

Komal Kapoor, Dhruv Sharma, and Jaideep Srivastava. Weighted node degree centrality for hypergraphs. In *IEEE Network Science Workshop*, pp. 152–155. IEEE, 2013.

J Sylvan Katz and Ben R Martin. What is research collaboration? *Research policy*, 26 (1):1–18, 1997.

Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1): 39–43, 1953.

Koray Kavukcuoglu, Pierre Sermanet, Y-Lan Boureau, Karol Gregor, Michaël Mathieu, and Yann L Cun. Learning convolutional feature hierarchies for visual recognition. In *Advances in neural information processing systems*, pp. 1090–1098, 2010.

Jingu Kim, Renato DC Monteiro, and Haesun Park. Group sparsity in nonnegative matrix factorization. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pp. 851–862. SIAM, 2012.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

S. Klamt, U.U. Haus, and F. Theis. Hypergraphs and cellular networks. *PLoS computational biology*, 5(5):e1000385, 2009a.

Steffen Klamt, Utz-Uwe Haus, and Fabian Theis. Hypergraphs and cellular networks. *PLoS computational biology*, 5(5):e1000385, 2009b.

Bryan Klimt and Yiming Yang. The enron corpus: A new dataset for email classification research. In *European Conference on Machine Learning*, pp. 217–226. Springer, 2004.

Tamara G Kolda. Numerical optimization for symmetric tensor decomposition. *Mathematical Programming*, 151(1):225–248, 2015.

Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009a.

Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009b.

Brian Kulis et al. Metric learning: A survey. *Foundations and Trends® in Machine Learning*, 5(4):287–364, 2013.

David Lazer, Alex Sandy Pentland, Lada Adamic, Sinan Aral, Albert Laszlo Barabasi, Devon Brewer, Nicholas Christakis, Noshir Contractor, James Fowler, Myron Gutmann, et al. Life in the network: the coming age of computational social science. *Science (New York, NY)*, 323(5915):721, 2009.

Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, volume 14, pp. 1188–1196, 2014.

Jean-Marie Lehn. Perspectives in supramolecular chemistryfrom molecular recognition towards molecular information processing and self-organization. *Angewandte Chemie International Edition in English*, 29(11):1304–1319, 1990.

Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1), 2007. doi: 10.1145/1217299.1217301. URL https://doi.org/10.1145/1217299.1217301.

Emmanuel D Levy, Elisabetta Boeri Erba, Carol V Robinson, and Sarah A Teichmann. Assembly reflects evolution of protein complexes. *Nature*, 453(7199):1262–1265, 2008.

Lei Li and Tao Li. News recommendation via hypergraph learning: encapsulation of user behavior and news content. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pp. 305–314. ACM, 2013.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

Pedro G Lind, Marta C Gonzalez, and Hans J Herrmann. Cycles and clustering in bipartite networks. *Physical review E*, 72(5):056127, 2005.

Li Lu, Cuihua Shen, and Dmitri Williams. Friending your way up the ladder: Connecting massive multiplayer online game behaviors with offline leadership. *Computers in Human Behavior*, 35:54–60, 2014.

Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011.

R Duncan Luce and Albert D Perry. A method of matrix analysis of group structure. *Psychometrika*, 14(2):95–116, 1949.

Alina Lungeanu, Yun Huang, and Noshir S Contractor. Understanding the assembly of interdisciplinary teams and its impact on performance. *Journal of informetrics*, 8(1): 59–70, 2014.

Alireza Makhzani and Brendan J Frey. Winner-take-all autoencoders. In *Advances in Neural Information Processing Systems*, pp. 2791–2799, 2015.

Joseph A Marsh and Sarah A Teichmann. Structure, dynamics, assembly, and evolution of protein complexes. *Annual review of biochemistry*, 84:551–575, 2015.

Mausam Mausam. Open information extraction systems and downstream applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pp. 4074–4077. AAAI Press, 2016.

Terry A McKee and FR McMorris. Topics in intersection graph theory, 1999.

Radosław Michalski, Sebastian Palus, and Przemysław Kazienko. Matching organizational structure and social network extracted from email communication. In *International Conference on Business Information Systems*, pp. 197–206. Springer, 2011.

Tom Michoel and Bruno Nachtergaele. Alignment and integration of complex networks by hypergraph-based spectral clustering. *Physical Review E*, 86(5):056111, 2012.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the*

26th International Conference on Neural Information Processing Systems, NIPS'13, pp. 3111–3119, 2013.

Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and Analysis of Online Social Networks. In *Proceedings of the 5th ACM/Usenix Internet Measurement Conference (IMC'07)*, San Diego, CA, October 2007.

Peter R Monge and Noshir S Contractor. *Theories of communication networks*. Oxford University Press, 2003.

Terrence J Moore, Robert J Drost, Prithwish Basu, Ram Ramanathan, and Ananthram Swami. Analyzing collaboration networks using simplicial complexes: A case study. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pp. 238–243. IEEE, 2012.

T. Mori, M. Uchida, and S. Goto. Flow analysis of internet traffic: World wide web versus peer-to-peer. *Systems and Computers in Japan*, 36(11):70–81, 2005.

Mark Mortensen, Anita Woolley, and Michael OLeary. Conditions enabling effective multiple team membership. *Virtuality and virtualization*, pp. 215–228, 2007.

James R Munkres. *Elements of algebraic topology*, volume 2. Addison-Wesley Menlo Park, 1984.

Roger B Nelsen. *An introduction to copulas*. Springer Science & Business Media, 2007.

Mark EJ Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409, 2001.

Mark EJ Newman, Duncan J Watts, and Steven H Strogatz. Random graph models of social networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(Suppl 1):2566–2572, 2002.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pp. 2014–2023, 2016.

Hongseok Oh, Myung-Ho Chung, and Giuseppe Labianca. Group social capital and group effectiveness: The role of informal socializing ties. *Academy of Management Journal*, 47(6):860–875, 2004.

Akshay Patil, Juan Liu, Bob Price, Hossam Sharara, and Oliver Brdiczka. Modeling destructive group dynamics in online gaming communities. In *Proceedings of the 6th Int. AAAI Conf. on Weblogs and Social Media*, volume 2012, 2012.

Akshay Patil, Juan Liu, and Jie Gao. Predicting group stability in online social networks. In *Proceedings of the 22nd international conference on World Wide Web*, pp. 1021–1030. International World Wide Web Conferences Steering Committee, 2013a.

Akshay Patil, Juan Liu, Jianqiang Shen, Oliver Brdiczka, Jie Gao, and John Hanley. Modeling attrition in organizations from email communication. In *IEEE Int'l Conf. on Social Computing (SocialCom)*, pp. 331–338, 2013b.

Kelly J Pearson and Tan Zhang. On spectral hypergraph theory of the adjacency tensor. *arXiv preprint arXiv:1209.5614*, 2012.

Jose B Pereira-Leal, Emmanuel D Levy, Christel Kamp, and Sarah A Teichmann. Evolution of protein complexes by duplication of homomeric interactions. *Genome biology*, 8(4):R51, 2007.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710. ACM, 2014.

Scott Poole, Marshall Scott Poole, and Andrea B Hollingshead. *Theories of small groups: Interdisciplinary perspectives*. Sage Publications, 2004.

Li Pu and Boi Faltings. Hypergraph learning with hyperedge expansion. In *Machine Learning and Knowledge Discovery in Databases*, pp. 410–425. Springer, 2012.

Li Pu and Boi Faltings. Understanding and improving relational matrix factorization in recommender systems. In *Proceedings of the 7th ACM conference on Recommender systems*, pp. 41–48. ACM, 2013.

Linda L Putnam and Cynthia Stohl. Bona fide groups. *Research Methods for Studying Groups and Teams: A Guide to Approaches, Tools, and Technologies*, pp. 211, 2012.

LL Putnam and Cynthia Stohl. Bona fide groups. *Communication and group decision making*, pp. 147–178, 1996.

Liqun Qi. Eigenvalues of a real supersymmetric tensor. *Journal of Symbolic Computation*, 40(6):1302–1324, 2005.

LQ Qi. H+-eigenvalues of laplacian and signless laplacian tensors. *Communications in mathematical sciences*, 2014.

R Ramanathan, A Bar-Noy, P Basu, M Johnson, W Ren, A Swami, and Q Zhao. Beyond graphs: Capturing groups in networks. In *IEEE INFOCOM Workshops*, pp. 870–875, 2011.

Seyed Hamid Rezatofighi, Anton Milan, Ehsan Abbasnejad, Anthony Dick, Ian Reid, et al. Deepsetnet: Predicting sets with deep neural networks. *arXiv preprint arXiv:1611.08998*, 2016.

Juan A Rodriguez. On the laplacian eigenvalues and metric parameters of hypergraphs. *Linear and Multilinear Algebra*, 50(1):1–14, 2002.

Juan Alberto Rodriguez. On the laplacian spectrum and walk-regular hypergraphs. *Linear and Multilinear Algebra*, 51(3):285–297, 2003.

Steven Rosenberg. *The Laplacian on a Riemannian manifold: an introduction to analysis on manifolds*, volume 31. Cambridge University Press, 1997.

Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

Atanu Roy, Ayush Singhal, and Jaideep Srivastava. Formation and reciprocation of dyadic trust. *ACM Transactions on Internet Technology (TOIT)*, 17(2):15, 2017.

Raif Rustamov and Leonidas J Guibas. Wavelets on graphs via deep learning. In *Advances in neural information processing systems*, pp. 998–1006, 2013.

Nahid Safari-Alighiarloo, Mohammad Taghizadeh, Mostafa Rezaei-Tavirani, Bahram Goliaei, and Ali Asghar Peyvandi. Protein-protein interaction networks (ppi) and complex diseases. *Gastroenterology and Hepatology from bed to bench*, 7(1):17, 2014.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala, and Yann LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3626–3633, 2013.

Hossam Sharara, Lisa Singh, Lise Getoor, and Janet Mann. The dynamics of actor loyalty to groups in affiliation networks. In *Social Network Analysis and Mining, 2009. ASONAM'09. International Conference on Advances in*, pp. 101–106, 2009.

Hossam Sharara, Lisa Singh, Lise Getoor, and Janet Mann. Stability vs. diversity: Understanding the dynamics of actors in time-varying affiliation networks. In *IEEE Int'l Conf. on Social Informatics (SocialInformatics)*. IEEE, 2012.

Ankit Sharma and Jaideep Srivastava. Group analysis using machine learning techniques. In *Group Processes*, pp. 145–180. Springer, 2017.

Ankit Sharma, Jaideep Srivastava, and Abhishek Chandra. Predicting multi-actor collaborations using hypergraphs. *arXiv preprint arXiv:1401.6404*, 2014.

Ankit Sharma, Rui Kuang, Jaideep Srivastava, Xiaodong Feng, and Kartik Singhal. Predicting small group accretion in social networks: A topology based incremental approach. In *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 408–415. IEEE, 2015.

Ankit Sharma, Terrence J Moore, Ananthram Swami, and Jaideep Srivastava. Weighted simplicial complex: A novel approach for predicting small group evolution. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 511–523. Springer, 2017.

Amnon Shashua, Ron Zass, and Tamir Hazan. Multi-way clustering using super-symmetric non-negative tensor factorization. *Computer Vision–ECCV 2006*, pp. 595–608, 2006.

David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.

Michael Simmons, Ayush Singhal, and Zhiyong Lu. Text mining for precision medicine: bringing structure to ehrs and biomedical literature to understand genes and health. In *Translational Biomedical Informatics*, pp. 139–166. Springer, 2016.

Ayush Singhal and Jaideep Srivastava. Leveraging the web for automating tag expansion for low-content items. In *Information Reuse and Integration (IRI), 2014 IEEE 15th International Conference on*, pp. 545–552. IEEE, 2014.

Ayush Singhal, Ravindra Kasturi, and Jaideep Srivastava. Automating document annotation using open source knowledge. In *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2013 IEEE/WIC/ACM International Joint Conferences on*, volume 1, pp. 199–204. IEEE, 2013a.

Ayush Singhal, Karthik Subbian, Jaideep Srivastava, Tamara G Kolda, and Ali Pinar. Dynamics of trust reciprocation in multi-relational networks. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 661–665. ACM, 2013b.

Ayush Singhal, Ravindra Kasturi, and Jaideep Srivastava. Datagopher: Context-based search for research datasets. In *Information Reuse and Integration (IRI), 2014 IEEE 15th International Conference on*, pp. 749–756. IEEE, 2014a.

Ayush Singhal, Atanu Roy, and Jaideep Srivastava. Understanding co-evolution in large multi-relational social networks. In *Information Reuse and Integration (IRI), 2014 IEEE 15th International Conference on*, pp. 733–740. IEEE, 2014b.

Ayush Singhal, Michael Simmons, and Zhiyong Lu. Text mining for precision medicine: automating disease-mutation relationship extraction from biomedical literature. *Journal of the American Medical Informatics Association*, 23(4):766–772, 2016a.

Ayush Singhal, Michael Simmons, and Zhiyong Lu. Text mining genotype-phenotype relationships from biomedical literature for database curation and precision medicine. *PLoS computational biology*, 12(11):e1005017, 2016b.

Ayush Singhal, Ravindra Kasturi, Ankit Sharma, and Jaideep Srivastava. Leveraging web resources for keyword assignment to short text documents. *arXiv preprint arXiv:1706.05985*, 2017.

Steven Skiena. Hasse diagrams. *Implementing Discrete Mathematics: Combinatorics and Graph Theory With Mathematica*, pp. 163, 1990.

Robert Speer and Catherine Havasi. Representing general relational knowledge in conceptnet 5. In *LREC*, pp. 3679–3686, 2012.

Myra Spiliopoulou. Evolution in social networks: A survey. In Charu C. Aggarwal (ed.), *Social Network Data Analytics*, pp. 149–175, Boston, MA, 2011. Springer US. ISBN 978-1-4419-8462-3. doi: 10.1007/978-1-4419-8462-3_6.

Karthik Subbian, Charu Aggarwal, and Jaideep Srivastava. Content-centric flow mining for influence analysis in social streams. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pp. 841–846. ACM, 2013.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pp. 697–706. ACM, 2007.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Jian Tang, Meng Qu, and Qiaozhu Mei. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1165–1174. ACM, 2015a.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pp. 1067–1077. ACM, 2015b.

Jie Tang, Duo Zhang, and Limin Yao. Social network extraction of academic researchers. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pp. 292–301. IEEE, 2007a. URL http://www.informatik.uni-trier.de/~ley/db/.

Jie Tang, Duo Zhang, and Limin Yao. Social network extraction of academic researchers. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pp. 292–301. IEEE, 2007b.

Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: Extraction and mining of academic social networks. In *KDD'08*, pp. 990–998, 2008.

Lei Tang and Huan Liu. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery*, 23(3):447–478, 2011.

Carla Taramasco, Jean-Philippe Cointet, and Camille Roth. Academic team formation as evolving hypergraphs. *Scientometrics*, 85(3):721–740, 2010.

Alva Taylor and Henrich R Greve. Superman or the fantastic four? knowledge combination and experience in innovative teams. *Academy of Management Journal*, 49(4): 723–740, 2006.

Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. Learning deep representations for graph clustering. In *AAAI*, pp. 1293–1299, 2014.

Ze Tian, TaeHyun Hwang, and Rui Kuang. A hypergraph-based learning algorithm for classifying gene expression and arraycgh data with prior knowledge. *Bioinformatics*, 25(21):2831–2838, 2009.

Ryota Tomioka and Taiji Suzuki. Convex tensor decomposition via structured schatten norm regularization. In *Advances in neural information processing systems*, pp. 1331–1339, 2013.

Théo Trouillon, Christopher R Dance, Éric Gaussier, Johannes Welbl, Sebastian Riedel, and Guillaume Bouchard. Knowledge graph completion via complex tensor factorization. *The Journal of Machine Learning Research*, 18(1):4735–4772, 2017.

Harry L Van Trees. *Detection, Estimation and Modulation Theory*. Wiley, New York, 1968.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103. ACM, 2008.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep

network with a local denoising criterion. *Journal of Machine Learning Research*, 11 (Dec):3371–3408, 2010.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. In *ICLR*, 2016.

Caroline S Wagner, J David Roessner, Kamau Bobb, Julie Thompson Klein, Kevin W Boyack, Joann Keyton, Ismael Rafols, and Katy Börner. Approaches to understanding and measuring interdisciplinary scientific research (idr): A review of the literature. *Journal of Informetrics*, 5(1):14–26, 2011.

S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Structural Analysis in the Social Sciences. Cambridge University Press, 1994. ISBN 9780521387071. URL http://books.google.com/books?id=CAm2DpIqRUIC.

Anita Williams Woolley, Christopher F Chabris, Alex Pentland, Nada Hashmi, and Thomas W Malone. Evidence for a collective intelligence factor in the performance of human groups. *Science*, 330(6004):686–688, 2010.

Li Xiaoyi, Li Hui Du Nan, et al. A deep learning approach to link prediction in dynamic networks. In *Proceedings of the 2013 SIAM International Conference on Data Mining. Philadelphia, PA, USA: SIAM*, 2013.

Jinshan Xie and An Chang. H-eigenvalues of signless laplacian tensor for an even uniform hypergraph. *Frontiers of Mathematics in China*, 8(1):107–127, 2013.

Maoqiang Xie, Taehyun Hwang, and Rui Kuang. Prioritizing disease genes by bi-random walk. In *Advances in Knowledge Discovery and Data Mining*, pp. 292–303. Springer, 2012.

Ye Xu, Dan Rockmore, and Adam M Kleinbaum. Hyperlink prediction in hypernetworks using latent social features. In *Discovery Science*, pp. 324–339. Springer, 2013.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015.

Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

Rachel Elana Zax. *Simplifying Complicated Simplicial Complexes: Discrete Morse Theory and its Applications*. PhD thesis, Harvard University, 2012.

Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. 2010.

Hongyuan Zha, Xiaofeng He, Chris Ding, Horst Simon, and Ming Gu. Bipartite graph partitioning and data clustering. In *Proceedings of the tenth international conference on Information and knowledge management*, pp. 25–32. ACM, 2001.

D. Zhou, J. Huang, and B. Scholkopf. Learning with hypergraphs: Clustering, classification, and embedding. *Advances in Neural Information Processing Systems*, 19: 1601, 2007.

Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in Neural Information Processing Systems*, pp. 1601–1608, 2006a.

Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in neural information processing systems*, pp. 1601–1608, 2006b.

Denny Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in neural information processing systems*, pp. 321–328, 2004.

Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pp. 912–919, 2003.

Jason Y Zien, Martine DF Schlag, and Pak K Chan. Multilevel spectral hypergraph partitioning with arbitrary vertex sizes. *IEEE Transactions on computer-aided design of integrated circuits and systems*, 18(9):1389–1399, 1999.

Ezra W Zuckerman. Do firms and markets look different? repeat collaboration in the feature film industry, 1935–1995. *Unpublished*, 2004.

Alexander Aleksandrovich Zykov. Hypergraphs. *Russian Mathematical Surveys*, 29(6): 89–156, 1974.