University of California
Santa Barbara

Analyzing, Mining, and Predicting Networked Behaviors

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Computer Science

by

Minh Xuan Hoang

September 2017

The Dissertation of Minh Xuan Hoang is approved.

Professor Xifeng Yan

Professor Subhash Suri

Professor Ambuj K. Singh, Committee Chair

June 2017

Analyzing, Mining, and

Predicting Networked Behaviors

To mom Duyen and my grandparents.

Acknowledgements

First and foremost, I would like to thank my adopted mom Duyen, my grandparents, my mother Ha, and other people in my big extended family for bringing me up and giving me the best in life.

Infinite gratitude goes to my research advisor Prof. Ambuj K. Singh. Thanks to his guidance and support in the last six years, I have learned and grown tremendously as a researcher and as a person.

I also thank my other committee members Prof. Xifeng Yan and Prof. Subhash Suri for their help and feedback. Moreover, I am indebted to the efforts of my collaborators: Prof. Sayan Ranu, Dr. Ram Ramanathan, Dr. Xuan-hong Dang, Dr. Yu Zheng, Prof. Bruno Ribeiro, Dr. Terrence J. More, Dr. Ananthram Swami, Dr. Xiang Wu, Dr. William Yan, Dr. Misael Mongiovi, Dr. Bong Jun Ko, Dr. Murtaza Zafer, Nazli Dereli, and Furkan Kocayusufoglu. I am very grateful to my lab mates at the Data Mining and Bioinformatics Lab (DBL) for being the greatest people to walk through grad school with. I also take this opportunity to acknowledge the role of all my teachers at the University of California, Santa Barbara and Hanoi University of Science and Technology, Vietnam.

I also thank all my friends in the U.S. and Vietnam, and all over the world. I would not have found the inspiration to go through grad school if it was not for the inspiration of the wonderful and positive people they are.

Curriculum Vitæ
Minh Xuan Hoang

Education

2017	Doctor of Philosophy (Ph.D.) Dept. of Computer Science, University of California, Santa Barbara, California, USA.
2016	Master of Science (M.S.) Dept. of Computer Science, University of California, Santa Barbara, California, USA.
2010	Bachelor of Engineering (B.Eng.) Computer Science Department, Hanoi University of Science and Technology, Vietnam.

Experience

2011 - 2017	*Research/Teaching Assistant* Department of Computer Science, University of California, Santa Barbara, California, USA.
2015	*Research Intern* Microsoft Research Asia, Beijing, China.
2014	*Research Intern* Machine Learning and Data Science Department, Adobe Systems, California, USA.
2013	*Research Intern* Networking Department, IBM T.J. Watson Research Center Yorktown Heights, NY.
2012	*Research Intern* Network Research Department, Raytheon BBN Technologies Cambridge, MA.
2010 - 2011	*R&D Engineer* VNG Corporation, Hanoi, Vietnam.
2008 - 2010	*Security Intern* Bkav Corporation, Hanoi, Vietnam.

Publications

2017	**Minh X. Hoang**, Furkan Kocayusufoglu, Ambuj K. Singh, "Summarizing Network Processes with Network-constrained Binary Matrix Factorization", *Submitted for review*.
2017	**Minh X. Hoang**, Xuan-Hong Dang, Xiang Wu, Zhenyu Yan, Ambuj K. Singh, "GPOP: Group-based Popularity Prediction of Online Content", in WWW, 2017.
2016	**Minh X. Hoang**, Yu Zheng, Ambuj K. Singh, "FCCF: Forecasting City-wide Flows of Crowd using Big Data", in ACM SIGSPATIAL, 2016.
2015	Bruno Ribeiro, **Minh X. Hoang**, Ambuj K. Singh, "Beyond Models: Forecasting Complex Network Processes Directly from Data", in WWW, 2015.
2014	Sayan Ranu, **Minh X. Hoang**, Ambuj K. Singh, "Answering Top-k Representative Queries on Graph Databases", in ACM SIGMOD, 2014.
2014	Sayan Ranu, **Minh X. Hoang**, Ambuj K. Singh, "Applications of Top-k Representative Queries", in SWIM, 2014.
2014	**Minh X. Hoang**, Ram Ramanathan, Ambuj K. Singh, "Structure and evolution of missed collaborations in large networks", INFOCOM, 2014.
2013	Sayan Ranu, **Minh X. Hoang**, Ambuj K. Singh, "Mining Discriminative Subgraphs from Global-state Networks", ACM SIGMOD, 2013.
2013	**Minh X. Hoang**, Ram Ramanathan, Terrence J. Moore, Ananthram Swami, "Structural and collaborative properties of team science networks." ASONAM, 2013.

Awards

2016	SIGSPATIAL Student Travel Award
2014	SIGMOD Student Travel Award
2016	Second-place Team in UCSB Startup Weekend
2014	Microsoft Research Asia Excellent Award
2011	Vietnam Education Foundation (VEF) Fellowship
2010	Honda YES Award for Young Engineers and Scientists

Abstract

Analyzing, Mining, and

Predicting Networked Behaviors

by

Minh Xuan Hoang

Network structure exists in various types of data in the real world, such as online and offline social networks, traffic networks, computer networks, brain networks, and countless other cases where there are relationships between different entities in the data. What are the roles of network structures in these data? First, the network captures inherent characteristics of the data themselves. This is clear from the definition of the network, which represents the relationship between entities: e.g., the social links among people in a social network describe how they interact with each other; a road network summarizes how the roads are laid out geographically; a brain network obtained from fMRI images represents pairs of brain regions that are active at the same time; a computer network constrains the paths via which internet packages and thus information or viruses can spread. Second, the network structures affect the evolution of the data over time. For example, new friendship links in an online social network are frequently created between friends of friends. Similarly, the current road network structure is without a doubt taken into consideration when roads are added or temporarily closed. As we grow, our brains also grow, including the additions of useful links or the clean up of unnecessary links between brain regions. Third, the network structures act as guidance for many different processes happening in the data. For instance, the links between users on social network dictate how gossips can spread; the roads influence how traffic flows in a city; the links between brain regions affects the way we think and how effectively we do things; the connections between computers route the transfer of any information on the internet.

In this thesis, I studied the network effect in various networked behaviors, including analyzing such effect, finding its patterns, and predicting future networked behaviors. First, I gained insights into the data by analyzing the accompanied network structures as well as its evolution. Second, I proposed algorithms for mining different network patterns that help summarize the effect of the network structures on different networked behaviors. Finally, I proposed models to predict the evolution of networked behaviors over time. Toward these tasks, I explored a wide variety of network data, including protein-protein interaction networks, online social networks, collaboration networks, chemical compounds, and traffic networks. Overall, I tackled these network data in different aspects and developed a number of methods for effectively mining and forecasting networked behaviors in data.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Thanks to the advance in technologies, data in general and network-based data in particular are being collected with unprecedented speed and volume. In contrast to traditional relational databases, network-based data also come with rich information integrated in the nodes, edges, and various processes happening in the networks.

The first and most popular example of large-scale network data are online social networks, such as Facebook, Twitter, Youtube, LinkedIn, among others. These networks have reached the sizes of millions to billions nodes and edges, basically changing the ways human interact with each other, search, and share information. Facebook, Twitter, and Youtube, are no longer just a technology framework, but have become news outlets, where many people find breaking and relevant news instead of traditional media, such as website, newspapers, or televisions. As a result, social networks are now one important channel of advertising, spreading news, and even influence people's opinion in presidential elections. LinkedIn, on the other hand, has become the major tool that 94% of recruiters use to find job candidates, instead of the more traditional job boards or just word of mouth. Besides their huge sizes, these social networks also introduce a myriad of interactions and detailed information of users and their behaviors. These rich data poses exciting challenges as to how we can understand and make use of them in a scalable manner due to their large size.

Network-based data is also prevalent in bioinformatics and chemistry. For example, the protein-protein interaction networks summarize the way different proteins interact with each other, which dictates many biological processes in a living body. Due to these interactions, when a protein behaves abnormally, it affects other proteins in its neighborhood, leading to diseases such as cancer. Understanding how a disease progresses via these protein interactions is thus essential to finding its cures and designing drugs. In chemistry, each chemical compound can also be represented as a graph, where nodes are atoms, and edges are their chemical bonds. If two chemical compounds have similar graph structures, it's likely that they also have similar chemical and physical characteristics. Therefore, studying their structures is useful for managing and exploiting these compounds based on their characteristics.

Another example is traffic networks, where nodes can be road intersections, public transportation stations, or city regions, and edges indicate if two nodes are adjacent. Clearly, these traffic networks affect the efficiency of city planning by influencing the flows of traffic. Specifically, since the structures of these networks constrain the traffic, a traffic jam in a road intersection can cause traffic jams in a number of nearby roads. Analyzing these traffic networks to find patterns and predict traffic flows is thus of paramount importance to effective management of the bloodline of cities. For example, we can answer questions such as how to plan future roads to facilitate a certain goal, how to distribute resources such as fire stations, polices, shopping malls to meet the demand of different residential areas, and how to block or regulate the traffic in advance to avoid traffic jams or stampede in overcrowded areas.

Other examples of data with inherent network structures include, but not limited to, brain networks, computer networks, web networks, and other cases where there exist relationships among different entities in the data.

There are a number of challenges in dealing with network data. First and foremost, the inherent network structures require new methods for mining, modeling, and forecasting in stead of traditional methods for data without structures. In fact, most problems in graphs are NP-hard, leading to the need for approximate solutions, whose quality is not always guaranteed. The subgraph search space is also exponential in the size of a graph, posing another scalability challenge, especially with the scale of millions to billions nodes nowadays. In particular, this subgraph search space cannot be stored or computed in its entirety due to its size. As a consequence, enumerating this space to find a subgraph of interest is also infeasible, thus requiring either sampling for some heuristic to prune the search space.

1.1 Thesis statement and contributions

In this thesis, I show that: "*Exploiting the network structures allows more meaningful and accurate analyzing, mining and predicting of networked behaviors.*"

As network structures exist everywhere, it is important to develop effective methods to understand the network effect and employ it for other useful purposes. In this thesis, I tackle all three steps in dealing with network data: (i) analyzing networked behaviors, (ii) mining patterns of networked behaviors, and (iii) predicting networked behaviors. In particular, this thesis is organized into three main parts accordingly.

- **Analyzing networked behaviors.** In Part I, I analyze to gain insights into the evolution of dynamic collaboration networks. In Chapter 2 and Chapter 3, I represent a collaboration network as a *simplicial complex*, which captures the group dynamics instead of the pairwise relationships among people. Based on this new representation, I track the evolution of collaboration networks with qualitative metrics to discover missed collaboration in Chapter 2, and to evaluate the collaborative performance of team science networks in Chapter 3. In Chapter 4, I propose the novel concept of "rising stars" in a dynamic graph—nodes whose features change suddenly and significantly in time, compared to recent history of both themselves and other nodes. Next, I investigate these rising stars in three real-world datasets and uncover valuable insights into how these stars evolve and interact with each other.

- **Mining patterns in networked behaviors.** To understand and model complex networked behaviors, the first challenge is to discover patterns in such complex data. Thus, in Part II, I propose three novel mining problems for networked behaviors, which help extract insights and further facilitate other useful tasks such as modeling and predicting. In Chapter 5, I find the top-k network patterns to best summarize a set of network processes. Each network pattern represents a local community of connected nodes frequently participating in the same network processes. In Chapter 6, given a function that classifies a data object as relevant or irrelevant, I select top-k objects that best represent all relevant objects in the underlying database. Since both of these problems are NP-hard, I propose greedy solutions that are effective and scalable. In Chapter 7, I mine discriminative subgraphs from global-state networks. These are the influential sub-networks that have maximum impact on the global state and unearth the complex relationships between the local entities of a network and their collective behavior. To deal with the exponential subgraph search space in Chapter 5 and Chapter 7, I utilize Monte Carlo Markov Chain sampling.

- **Predicting networked behaviors.** After I have verified the network effect and found its patterns in the earlier two parts, in Part III, I aim at modeling and forecasting complex network phenomena – such as information cascades in online social networks or traffic flows in a city. In Chapter 8, I predict the movement of crowds in a city based on big data, including the road network, the region graph, and weather information. This problem is strategically important for traffic management, risk assessment, and public safety. I combine the network structure with other signals in the data to propose a scalable and accurate method. In Chapter 9 and Chapter 10, I forecast the spread of information in online social network. In particular, I predict the popularity of online content in social networks in Chapter 9, which is important in many applications, ranging from ad campaign design, web content caching and prefetching, to web-search result ranking. Finally, in Chapter 10, I propose a model-free approach to forecasting network processes based on relationships of statistical equivalence using two general axioms and historical data. To the best of our knowledge, SED is the first method that can perform axiomatic, model-free forecasts of complex stochastic processes.

In summary, in this thesis, I tackle three main problems in dealing with network-based data: analyzing, mining, and predicting. I propose solutions that are effective and scalable, using a wide range of techniques and properties such as Monte Carlo Markov Chain sampling, submodularity, tensor decomposition, statistical hypothesis testings, and other machine learning techniques. Extensive experiments on real-world datasets demonstrate excellent accuracy and running time of our proposed methods, as well as meaningful patterns and models.

Part I

Analyzing Networked Behaviors in Collaboration Networks

Before exploiting the network structures for mining and predicting tasks, we first need to verify if the network structures do impact different behaviors in the data. Therefore, in this section, I analyze different types of network data and gain insights into their characteristics and evolution. In particular, I work with collaboration networks, including the IMDB movie network, the DBLP bibliography network, the Enron email network, and two team science networks.

In Chapter 2, I study the nature of missed collaboration opportunities in evolving collaboration networks. I define a k-way missed collaboration as one in which every $(k-1)$-subset of the k persons has collaborated but the set of k has not. Representing a collaboration network as a *simplicial complex*, I model a missed collaboration as a *Minimal Non Face (MNF)*. Focusing on 2-dimensional and 3-dimensional MNFs, equivalent to 3-way and 4-way missed collaborations respectively, I analyze the DBLP publication network and the IMDB movie network. Our key findings are as follows. A large number of missed collaborations arise, but only a few persist for long. Specifically, the persistence time appears to be exponentially distributed for both 2-MNFs and 3-MNFs. Nodes with higher degree centrality are more likely to be part of 2-MNFs but little correlation was found with 3-MNFs. Considering the network of missed collaborations, the number of components as of today appears to be power law distributed across MNF types and data sets, but its evolution shows a divergence between DBLP and IMDB. I identify specific missed collaborations, and observe interesting patterns in the occurrence of names in pairs and triplets. Our results can help in developing random generative models of collaboration networks, cue researchers in on potential fruitful collaborations, and predict new collaborations.

In Chapter 3, I further apply the concept of simplex into team science. Team science is a collaborative approach to research, typically with researchers drawn from different disciplines. Team science networks have certain unique characteristics in their conception and intent that set them apart from other commonly studied social and collaboration networks. I study the structural properties, and present metrics for collaborative performance assessment in two real-world team science networks initiated by the Army Research Lab. I model a team using a higher-order generalization of an edge called a *simplex*. A simplex captures group relationships distinct from the union of pair-wise relationships. Our evaluation using a rigorous methodology reveals that the distributions of vertex and facet degrees (the number of maximal groups that a vertex belongs to) follow a power law, but with exponential cut-off at the tail in most cases. I propose metrics for quantitatively assessing the extent of intra-team and extra-team collaborations, and compare their effectiveness vis-a-vis our intuitive notions. Our work can be used as the basis for generative models, and for evaluating the collaborative performance of team science networks.

In Chapter 4, I look into dynamic graphs, which are widely used to model time-evolving interactions between entities in many types of networks. In such networks, each node can be characterized by the topological features extracted from its 1-hop neighborhood (its egonet) and the evolution of these features over time. I propose the novel concept of "rising stars" in a dynamic graph—nodes whose features change suddenly and significantly in time, compared to recent history of both themselves and other nodes. A unique aspect of our work is that even if a node has a sudden behavioral change with regards to its own history, it is not a rising star if the change in its evolutionary pattern is the same as a global trend affecting all nodes in a network. Using Chernoff bounds, I leverage the definition of rising stars to define "star creators"—nodes that contribute to the emergence of many neighboring rising stars. I investigate the relationship among rising stars as well as their relationship with the star creators and "super stars"—nodes that stand out with highest level of activity. Case studies on three real-world datasets show that our proposed concepts can be applied to find meaningful rising stars in their early stages, and pinpoint interesting nodes or events in dynamic networks. Further, our empirical analysis uncovers valuable insights into how these stars evolve and interact with each other.

Chapter 2

Structure and Evolution of Missed Collaborations in Large Networks

2.1 Introduction

A social collaboration network is a set of *actors* (e.g. researchers) who interact with each other by means of certain *collaborative acts* (e.g. co-authored publications) [8]. The value of strong collaborations in making an impact cannot be questioned [9, 10]. Many collaboration networks are formed largely autonomously, without any centralized control on collaboration. In such networks, do all fruitful collaborations come to bear? Or are there collaborations that appear natural and potentially fruitful, but do not come to pass even after a large number of years?

We investigate the nature of such missed collaborations in large collaboration networks. A missed collaboration is one where the participants are close enough in their skills that a k-collaborative act makes sense, but did not happen. We use a purely structural way of determining “close enough", based on the participants’ existing collaborations. Specifically, we say that there exists a k-way missed collaboration if every $(k-1)$-cardinality subset of the k

actors have collaborated, but the k actors as a set have not. For example, if (A,B), (B,C) and (C,A) have co-authored three different publications, but there is no publication where A, B, and C are joint co-authors, we say that there is a 3-way missed collaborations. In this example, the fact that A, B and C have pair-wise collaborated is considered sufficient evidence that their areas are close enough, yet they have missed collaborating on a paper together. The concept can similarly be extended to 4-way, 5-way and general k-way missed collaborations.

Collaboration networks are traditionally modeled by graphs (see [11] and references therein), but such a representation does not capture collaboration as a *group*. For example, consider a complete co-authorship graph on 4 vertices that represent four authors. Does this graph represent a single paper with four authors, four 3-author papers, or six 2-author papers? In particular, for the purposes of this paper, four 3-author papers is a missed 4-way collaboration, whereas a single 4-author paper is not – but a graph representation cannot distinguish between the two. While some researchers have suggested the use of bi-partite graphs with edges between actor vertices and collaboration vertices, the analysis itself almost always is done on a "one-mode projection" of these graphs [8]. What we really need is an abstraction where higher-order aggregations can be represented distinctly from the union of pair-wise collaborations.

In this paper, we use the *abstract simplicial complex* to represent and analyze collaboration networks. An abstract simplicial complex consists of a set V and a set of subsets of V closed under the subset operation. A simplicial complex is a generalization of a graph and therefore admits any analysis or metric based on graphs, but additionally provides analytical possibilities not possible with a graph-based representation. In section 2.2.1 we provide a brief background on simplicial complexes as necessary for understanding this paper. Prior works [12–14] have established the usefulness of simplicial complexes for analyzing collaboration networks[1]. We capture a missed collaboration as a well known feature of simplicial complexes called a *Mini-*

[1]The *hypergraph* [15] is another possible abstraction, but as argued in [13], a simplicial complex is a better fit as it is closed under subsets, capturing the subset closure property of the collaboration relationship.

mal Non Face (MNF). In particular, a k-way missed collaboration translates into a $(k-1)$-MNF in the associated collaboration complex. We focus on 2-MNFs (3-way missed collaborations) and 3-MNFs (4-way missed collaborations) as they are the only non-trivial MNFs occurring frequently enough for statistical analysis.

Our analysis has uncovered some key properties that cut across the two data sets. The persistence (number of years a missed collaboration lasts) is exponentially distributed for both MNF types. In other words, a vast majority of missed collaborations happen naturally in a few years after they form. The number of components of the MNF-induced network is power-law distributed for both MNF types. The number of MNFs a vertex is part of has a high correlation with the vertex and facet degrees for 2-MNFs, but not for 3-MNFs.

Some other features are remarkable in their differences across DBLP and IMDB. Whereas MNFs grow exponentially with time in DBLP, we can discern no clear pattern for IMDB, and in fact, the MNF increase in IMDB is surprisingly not even monotonic. Further, the growth of the number of components in the MNF-induced network slows in the latter years for DBLP (indicating an increase in connectivity growth), whereas it is just the opposite for IMDB. We discuss possible explanations for these phenomena in section 2.3.2. We have also created a list of all missed collaborations and their persistence. We present a sample, and discuss some informal observations. For instance, names that figure at the top of the 2-MNF list tend to combine as the most frequent pairs as well.

Our statistical observations can be useful in constructing new generative models of evolving collaboration networks, and our technique for identifying specific missed collaborations can be useful in recommending potentially fruitful new collaborations. The observation that most MNFs have low persistence can help in models that predict future collaborations.

2.2 Preliminaries

2.2.1 Simplicial Complex

An *abstract simplicial complex*[2] is denoted by $\Delta = (V, S)$, where V is a set of vertices, S is a non-empty set of subsets of V closed under the subset operation, i.e., for any $S_k \in S$, all subsets of S_k are also in S. A *simplex* or a *face* of a simplicial complex $\Delta = (V, S)$ is any subset $s \in S$. The *dimension* of a simplex is one less than the number of vertices in it. The dimension of a simplicial complex is the maximum dimension of the simplices in it. A graph is a special case of a simplicial complex, i.e., a simplicial complex of dimension 1. A *facet* of a complex is a maximal face, i.e., a face that is not a subset of any other face. The *facet degree* of a vertex is the number of facets that the vertex is a part of.

Figure. 3.1 shows a simple example simplicial complex. The facets are (0, 1, 2), (2, 3, 4), and (1, 4, 5, 6), and the faces (simplices) are the subsets of the facets, including the facets themselves. The dimension of this simplicial complex is 3. The vertex degree of vertex 1 is 5, whereas its facet degree is 2. The facet degree of node 3 is 1.

A *minimal non-face* (MNF) is a set of vertices M in a simplicial complex such that every subset of M except M itself is a simplex. If an MNF has $k+1$ vertices, it is called a *k-minimal non-face* (k-MNF), or an MNF of dimension k. For example, in Figure 3.1(a), (1, 2, 4) is a 2-MNF. A 1-MNF is a missing edge and represents an un-interesting trivial missed collaboration. In the complex {(1,2,3),(2,3,4),(1,2,4),(1,3,4)}, the set (1,2,3,4) is a 3-MNF, or a 4-way missed collaboration.

We have only given the minimum background required for understanding the rest of the paper. Readers interested in learning more about simplicial complexes and algebraic topology in general are referred to [16].

[2] We will henceforth drop the word “abstract" for brevity.

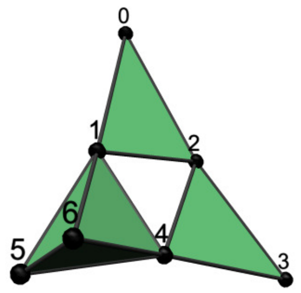


Figure 2.1: Example simplicial complex

2.2.2 Datasets

The DBLP Computer Science Bibliography

The DBLP Computer Science Bibliography is an online reference for bibliographic information on major computer science publications [17]. We extract all of the papers in this database from 1936 until September 2013 to create a dataset of 3,625,017 papers and 1,302,447 authors.

IMDB - The Internet Movie Databases

The Internet Movie Database (IMDB) is an online database of information related to films, television programs and other productions [18]. The database includes information regarding actors, actresses, directors, year of release, and other film-related information from year 1894 to 2013. We extract all films and the cast whose credits are less than or equal to 5 (the most important actors/actresses) to create a dataset with 488,238 cast members and 1,057,991 film titles.

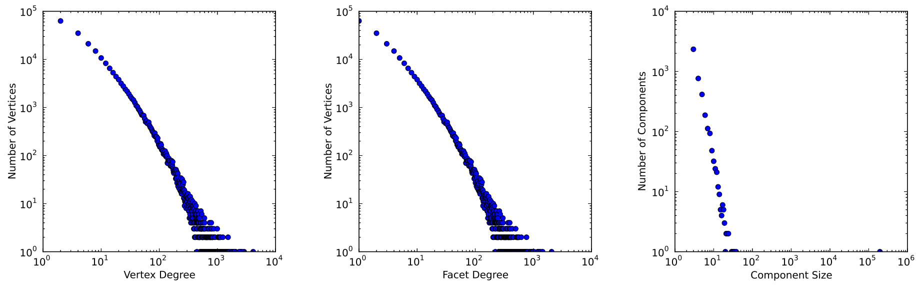
2.2.3 Representation as a simplicial complex

We represent each member in a data set as a vertex, and each collaborative act (movies, papers), as a facet of vertices comprising the collaborative act. Facets may share vertices. Thus, in the DBLP complex, each vertex represents a researcher and each simplex represents a collaboration relationship amongst the researchers (vertices) on one or more papers. Note that the number of facets may be less than the number of papers – for example, if there is a paper P1 by (A,B,C), and P2 by (A,B), we only have facet (A,B,C). The closure property of simplicial complexes means that (A,B) is automatically part of the complex. We note that the “collaboration" relationship is similarly closed under subsets, and therefore well-suited for modeling using simplicial complex. Similarly, in the IMDB simplicial complex, each vertex represents a cast member and each simplex represents a collaboration relationship amongst the cast members on one or more productions.

The evolution of collaboration is captured as a cumulative simplicial complex. Let SC(y) represent the simplicial complex using data from only year y, and SC(1) the first available year complex. Then, the evolving simplicial complex EvoSC(i) = $\cup_{y=1}^{i}$ SC(y). The *persistence* of an MNF is the number of consecutive years it is present in EvoSC.

The *persistence* of an MNF M that first appears in EvoSC(j) is the number of consecutive years Y such that M is in EvoSC($j + k$) for every $0 \leq k \leq Y$, and M is not present in EvoSC($j + Y + 1$). For example, if an MNF appears in the evolving 1943 complex and is present in 1944, 1945 and 1946, but not in 1947, then the persistence of the MNF is 3.

Some of our studies pertain to the network of MNFs. An *MNF network* or *MNF complex* of a simplicial complex has a vertex set equal to the union of vertices comprising the MNF and a facet corresponding to each MNF. In other words, it is the sub-complex induced by the MNFs of the complex.



(a) Vertex Degree (b) Facet Degree (c) Component count

Figure 2.2: Vertex degree (VD), facet degree (FD) and component size distributions of the 2-MNF-induced networks (DBLP 2013), log-log scale

2.3 Missed Collaborations: Structure and Evolution

We present a number of findings relating to the structure of MNFs in the DBLP and IMDB networks, and their evolution and persistence over time. Below, we use 2-MNF synonymously with a 3-way missed collaboration, and 3-MNF synonymously with a 4-way missed collaboration. The number of k-MNFs for k $>$ 3 are too few to draw conclusions from and therefore we do not consider them in our study. We thus have four combinations of {DBLP, IMDB} x {2-MNF, 3-MNF} studies in each section below. Note that, as stated in section 2.2.3, the simplicial complex referred to for year i for all of the below is the *evolving* complex which contains all collaborations up to that year i.

2.3.1 Structure as of 2013

We consider the MNF complex in the DBLP and IMDB, and study the question: *What is the distribution of the vertex and facet degrees, and number of components? In particular, given the preponderance of power law in network science, are these also power law?*

Figure 2.2 shows the distribution of vertex degree, facet degree and number of components for DBLP 2-MNF on a log-log scale. Visually, this appears to be power law distributed. However, visual analysis can be deceptive, hence, we analyze the distribution using Clauset's

Table 2.1: Power-law fit for distributions of degree and component size $P[x = k] \propto k^{-\alpha}$ for $x \geq x_{min}$

	Feature	2-MNF induced network			3-MNF induced network		
		x_{min}	α	p-value	x_{min}	α	p-value
DBLP	Facet deg.	72	2.83	0.17	11	3.3	1
	Vertex deg.	144	2.85	0.11	30	3.58	1
	#Comps	3	3.55	0.82	8	3.01	0.93
IMDB	Facet deg.	440	2.69	0.04	118	2.51	0.68
	Vertex deg.	880	2.69	0.04	354	2.51	0.66
	#Comps	4	2.62	1.0	14	2.01	1

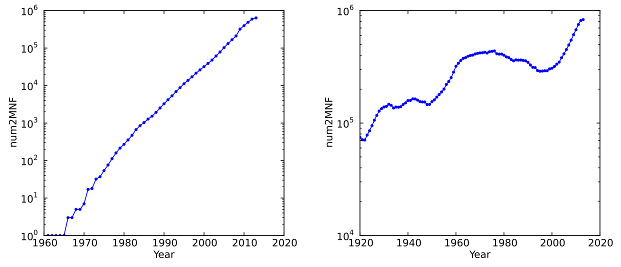
methodology [19]. The results are shown in Table 2.1 for DBLP 2-MNFs as well as the other three combinations. A p-value < 0.05 rejects the power-law hypothesis, and a higher $xmin$ value dilutes it. From the table, it appears that power law is indicated as a good fit for component count distributions for all 4 combinations, and for vertex and degrees of DBLP-3-MNF. Vertex and facet degrees of IMDB-2-MNF do not follow a power law, while the vertex and facet degrees of IMDB-3-MNF and DBLP-2-MNF show a somewhat weak fit to power law.

Thus, MNF network is structurally different in terms of degree distributions, with DBLP MNF networks having more of the well-known scale-free properties. However, surprisingly, from a global perspective of connectivity, they are similar, displaying strong power law properties.

Are higher degree vertices more likely to be part of more MNFs? Table 2.2 summarizes the Pearson correlation between the vertex/facet degree of a node and the number of MNFs it belongs to, for each of the four combinations. We observe that vertex and facet degrees for both IMDB and DBLP are correlated fairly strongly with number of 2-MNFs, but only very weakly with number of 3-MNFs. Thus, it appears that actors with high degree centrality are more at risk for missing 3-way collaborations. This is intuitive because the density around a node generates more collaborations overall and hence more missed, but the fact that this is not true for 4-way collaborations is somewhat surprising.

Table 2.2: Pearson correlation of vertex degree and the number of MNFs that vertex belongs to (Year 2013).

		2-MNF	3-MNF
IMDB	FD	0.7098	0.4828
	VD	0.7892	0.1946
DBLP	FD	0.7398	0.1100
	VD	0.7810	0.1494



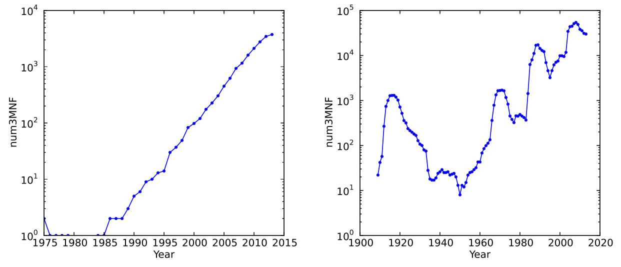
(a) DBLP (b) IMDB

Figure 2.3: Number of 2-MNFs over the years

2.3.2 Evolutionary Characteristics

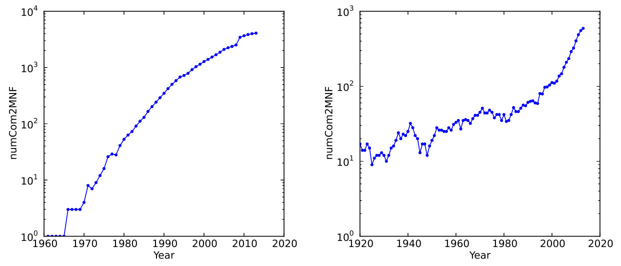
In this section, we study the following questions; *(a) How does the number of MNFs evolve over time?* and *(b) Do the MNFs get more connected or less connected over time?*

Figures 2.3 and 2.4 plot the number of MNFs as a function of years, in a semi-log plot. DBLP clearly shows exponential growth for both 2- and 3-MNFs. The 3-MNFs, not surpris-



(a) DBLP (b) IMDB

Figure 2.4: Number of 3-MNFs over the years



(a) DBLP (b) IMDB

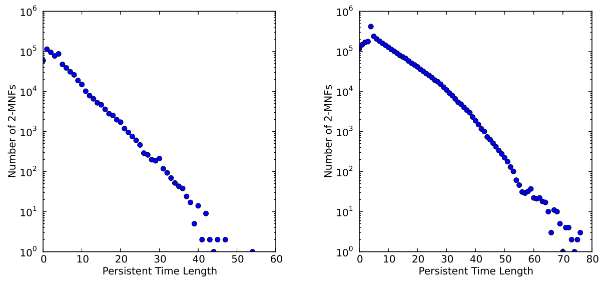
Figure 2.5: Number of connected components of the 2-MNF induced network over the years

ingly, start appearing at a much later date due to the required density and are fewer in number, but increase exponentially nonetheless.

On the other hand IMDB doesn't show a clear pattern, and in fact the number of 2- and 3-MNFs dips multiple times. This might have to do with the connectivity behavior of the original network (vice the MNF complex) as the MNFs can only form when there is good connectivity in a region.

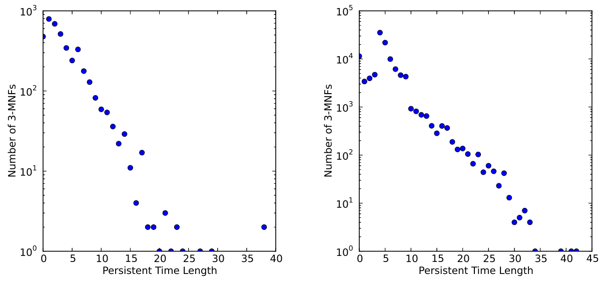
Figure 2.5 plots the number of connected components as a function of the years for 2-MNF network. We observe an interesting divergence between the behavior in DBLP and IMDB – whereas the growth in the number of components tapers off for DBLP during the latter years, it actually increases for IMDB toward the latter years. The behavior for 3-MNFs is a more muted version of the same behavior, and not shown here for space constraints. We believe this might also be a direct consequence of the increasing and decreasing connectivity of original network (vice the MNF complex) for DBLP and IMDB respectively. The increasing connectivity of DBLP is supported by the “densification" observations in [20]. Our own analysis of the average vertex degree evolution or the original network (not shown here due to space constraints), also shows an increase in growth rate for DBLP and a decrease for IMDB which further supports our hypothesis.

Why would the IMDB network get less dense with time? One reason for this could be that,



(a) DBLP (b) IMDB

Figure 2.6: Persistent time length of 2-MNFs over the years



(a) DBLP (b) IMDB

Figure 2.7: Persistent time length of 3-MNFs over the years

unlike DBLP, IMDB consists of not only movies, but also documentaries and other productions, and from a number of different countries. Since each genre of production and each country tends to have its own largely disjoint community, and the diversity of genre and nationality has increased in the last few decades, the network is likely to have more components in the latter decades. Using only movies, and only from Hollywood could show smooth trends paralleling DBLP, but is left for future work.

2.3.3 Persistence Properties

In this section, we study the question: *How long does a missed collaboration persist? What is the distribution of this persistence time?*

Based on the ubiquity of power law in network science, one might conjecture – as we did – that the distribution might be a power law. Figure 2.6 and 2.7 show, respectively, the distribution of the persistence of 2-MNFs and 3-MNFs plotted on a semi-log scale. The figures show that the persistence length of MNFs is not power law, but appear to be *exponentially distributed*. This means that the majority of MNFs that arise get closed quite soon, and only a few tend to last very long. For example, 72% of 2-MNFs and 84% of 3-MNFs have a persistence time ≤ 5 years. IMDB 2-MNF persistence is much longer compared to DBLP, in particular 9.12 years vs 4.37 years. This is likely due to the fact that IMDB history starts earlier. The fact that most MNFs tend to close soon means that we can use MNFs to predict future collaborations.

Unlike the two previous studies in sections 2.3.1 and 2.3.2, there is no significant difference in the behavior of MNF persistence between DBLP and IMDB. Thus it appears that insofar as individual MNF evolutionary features are concerned (Figures 2.6 and 2.7), there is more uniformity whereas if network-wide features are concerned (Figures 2.3, 2.4, and 2.5), there is a marked difference. This is probably related to the difference in the whole (vice MNF-induced) network. An interesting open question is if the exponential distribution of MNFs holds for other kinds of collaboration or other social networks, and if it might be the result of some natural process (similar to preferential attachment resulting in power law distribution).

2.3.4 Example Missed Collaborations

In section 2.3.3 we noted that a vast majority of MNFs closed in a few years. However, those that did last long are of particular interest, especially if they still remain open as of today. Enabling such collaborations by identifying them might be useful, especially since it is looking like they are not going to form naturally. In this section, we ask: *What are some actual long-persisting 3-way and 4-way missed collaborations? Are there any salient features of such examples?*

Table 2.3: Some example high-persistence 3-way missed collaborations

	Missed 3-way Collaborations	Yrs
DBLP	John W. Carr III, Walter F. Bauer, Alan J. Perlis	54
	John E. Hopcroft, Jeffrey D. Ullman, Richard M. Karp	42
	Robert E. Tarjan, Shimon Even, M. R. Garey	37
	David Peleg, Amotz Bar-Noy, Cynthia Dwork	26
	Alok N. Choudhary, Donald F. Towsley, David M. Nicol	19
IMDB	Judy Garland, Lucille Ball, Bert Lahr	68
	Angela Lansbury, Elizabeth Taylor, Janet Leigh	64
	Bing Crosby, Frank Sinatra, Debbie Reynolds	54
	Jane Fonda, Susannah York, Michael Caine	41
	Hema Malini, Amitabh Bachchan, Vinod Khanna	39

Table 2.3 and 2.4 show, respectively, some specific examples of 3-way and 4-way missed collaborations that are open as of 2013, and their persistence in number of years. The first line in each of the four combinations is the missed collaboration with the longest persistence. The other examples are chosen by the authors as a sample with an eye toward well known names. We note that some of the missed collaborations cannot be closed as the person no longer exists (e.g. Judy Garland[3])

In looking at the top several thousand 2-MNFs and 3-MNFs we have observed some interesting patterns. Names that figure at the top of the 2-MNF list tend to combine as the most frequent pairs. For example, in the DBLP 2-MNF list (Noga Alon (149)[4], Paul Erdos (123)) also combine as the most frequent edge in missed 2-MNFs. Similarly, names that figure at the top of the 3-MNF list tend to combine as the most frequent pairs and triplets. For example, in the DBLP 3-MNF list (Manish Gupta (99), Valentina Salapura (66)) are the most frequent edge in missed 3-MNFs, and also figure as part of the top triplets. We have also noted that there is very little intersection in the set of names in 2-MNFs and 3-MNFs in DBLP or IMDB. For example, Noga Alon figures just once and Paul Erdos never figures in the 3-MNF list. A

[3]Although we only considered an MNF if the person was active in the database over the past 5 years, Judy Garland is credited on certain recent films about her and therefore figures in the database!

[4]The number in parentheses is the number of times a person appeared in the top 10,000 MNFs.

Table 2.4: Some example high-persistence 4-way missed collaborations

	Missed 4-way Collaborations	Yrs
DBLP	H. J. Bowlden, B. J. Mailloux, J. E. L. Peck, C. H. A. Koster	38
	Y. Sagiv, J. D. Ullman, A. O. Mendelzon, D. Maier	27
	M. Charikar, A. Tomkins, J. M. Kleinberg, P. Raghavan	13
	S Ramaswami, F. Hurtado, E. D. Demaine, G. T. Toussaint	11
IMDB	T. Leblanc, A. Landa, L.Valenzuela, C. Velasco	42
	J. Belushi, P. Simon, G. Morris, J. Curtin	33
	J. Karlen, G. Hall, D. Selby, J. Bennett	26
	D. Malone, M. Farrow, E. Nelson, J. Douglas	25

Table 2.5: Summary of observations

	2-MNF	3-MNF
DBLP	**#Comps is strongly power law**	**#Comps is strongly power law**
	VD/FD weak power law	VD/FD power law
	High correlation w/ VD/FD	No/weak correlation w/ VD/FD
	#MNFs grow exponentially	*#MNFs grow exponentially*
	#Comps growth slows in tail	*#Comps slows in tail*
	Persistence exponentially distr	**Persistence exponentially distr.**
	Top singles also in top pairs	**Top singles in top pairs & triplets**
IMDB	**#Comps is strongly power law**	**#Comps is strongly power law**
	VD/FD not power law	VD/FD weak power law
	High correlation w/ VD/FD	No/weak correlation w/ VD/FD
	#MNFs growth non-monotonic	*#MNFs growth non-monotonic*
	#Comps growth faster in tail	*#Comps growth faster in tail*
	Persistence exponentially distr.	**Persistence exponentially distr.**
	Top singles also in top pairs	**Top singles in top pairs & triplets**

more rigorous analysis of these and other patterns is left for future work.

Our list of missed collaborations could form the basis of a recommendation system for new multi-way collaborations, especially in conjunction with a filtering system based on profiles such as in [21].

2.3.5 Discussion

Table 2.5 summarizes the observations from the previous subsections in the order they were presented. The observations that appear to hold across all four combinations of data sets and

MNF types are in bold. Those that differ markedly across DBLP and IMDB are shown in italics. Within each of DBLP and IMDB, other than the correlation of vertex and facet degrees to number of MNFs, observations largely hold across 2-MNFs and 3-MNFs.

The exponential distribution of persistence and the power law distribution of the number of MNF-network components appear to be interesting properties that merit further investigation to judge universality. Further, for DBLP, we also see other properties such as the exponential growth of MNFs, and a clear slowing of growth in the latter years, although that might be an artifact of the overall network evolution. The differences between DBLP and IMDB are interesting, but may be explained to some extent by the nature of the communities: DBLP, being more uniform and closer-knit in terms of acquaintance; IMDB, having not only movies but also documentaries and TV programs, and from diverse countries, is more disconnected and does not display the smooth trends seen in DBLP.

Some caveats are in order in interpreting our results. First, the identified missed collaborations may have happened outside of DBLP or IMDB. Second, names of the same person may differ, or different people may have the same name in the database. Finally, our trimming of the IMDB database by considering only the top credited actors/actresses may not be a representative sample of the whole.

2.4 Related Work

The last decade has seen a spurt in the study of collaboration networks [8, 11, 22–24]. In [11, 22], structural properties of publicly available scientific publication networks are analyzed, with the latter also investigating evolutionary aspects. Movie actor and the DBLP networks are analyzed in [24]. Self-organization and classification into different kinds of small-world networks appear in [8, 24]. The quest to discern power laws in the distribution of social networks has been the subject of much work in the literature [19, 25]. Visual and other simplistic methods, however, may be misleading [19, 26]. A rigorous method for verifying if a distribution follows power law is given in [19], which we apply in this paper.

Well established in mathematics, in particular algebraic topology [16], simplicial complexes have been used as a part of Q-analysis in the 1970s to analyze general structure [12], which have been applied into specific social network problems [27].

The application of simplicial concepts to collaboration networks appears to varying extents in [13,14]. In [14], 2-MNFs were briefly studied, but for much smaller collaboration networks. There has been some work on recommendation systems for collaboration (e.g. [21,28,29]), but these works are focused on recommending one other individual, utilize profile information to make a match, do not consider the evolutionary information, and do not study the statistical properties.

To our knowledge, ours is the first work that investigates the structure and evolution of multi-way missed collaborations in large, autonomous networks using a simplicial model.

2.5 Concluding Remarks

We have studied the structure and evolution of missed collaborations in DBLP and IMDB by modeling it as a Minimal Non Face (MNF) in the corresponding simplicial complex. We have discovered that some properties – e.g. distribution of MNF persistence – is exponential across 2- and 3-MNFs and DBLP/IMDB; and some properties – e.g. connectivity evolution – show some remarkable differences. Our statistical results can be used to create or validate random generative models of collaboration networks, and our techniques for identifying missed collaborations can be used as a recommender system for multi-way collaborations, or predict new collaborations.

There are a number of interesting avenues for further exploration: a more in-depth analysis of the results, and better support for our explanations; a similar study of other data sets (e.g.

ArXiv, PubMed, SourceForge); an efficient generalized algorithm for computing k-MNFs and persistence across time; and relationships with other features of simplicial complex, for e.g Betti numbers.

Chapter 3

Structural and Collaborative Properties of Team Science Networks

3.1 Introduction

A social collaboration network is a set of people who interact with each other by means of certain collaborative acts. Examples include publication co-authorship, movie collaborations and teams in organizations. The analysis of the structural and evolutionary properties of collaboration networks has been the subject of considerable research over the past decade [8, 11, 22–24, 30].

We consider a specific type of collaboration network called a *team science network*. According to [31], "Team science has been described as a collaborative and often cross-disciplinary approach to scientific inquiry that draws researchers who otherwise work independently or as co-investigators on smaller-scale projects into collaborative centers and groups". Increasingly, the hardest problems today require the combined effort of scientists from different fields. Thus, there has been a surge of interest and investment in team science programs by several public agencies such as NIH, NCI, NCCR, ARL, and other private foundations. It is therefore imperative that we understand the structural and collaborative aspects of team science networks.

Team science networks have certain unique features that set them apart from the more typical and well-studied collaboration networks such as co-authorship and movie networks. First, unlike the latter networks which are formed by independent decisions of the actors without an overarching management, team science networks are almost always created by an agency. Second, team science networks are often lightly managed to encourage collaboration between researchers of specific backgrounds, which constrains team formation, but is not as rigid as in business organizational teams. Finally, the structure of actual collaboration in terms of, say, papers co-written, may be different from the team structure.

We investigate two broad questions regarding team science networks: Do topological properties such as degree distribution carry over from other studied collaboration networks (such as co-authorship or movie networks) to team science networks? Given that the main reason for team science is the cross-disciplinary interaction, is there a quantitative way to measure whether such interaction is happening?

Collaboration networks are typically modeled by graphs [32]. However, as recently argued in [13], graphs can only capture pairwise relationships, but are ill-equipped to capture *group* aspects underlying teams and other other collaborations. For example, consider a complete graph on 4 vertices that represent four people. Does this graph represent a single team of four, four 3-member teams, or six 2-member teams? For such situations, we need an abstraction where higher-order (group) aggregation is a primitive.

In this paper, we represent each collaborator (team member) by a vertex. A set of members who are in the same team is called a *simplex*. A simplex represents the relation “is in the same team" and thus every team, as well as every possible sub-team, is a simplex. Thus, a team science network is a set of possibly overlapping simplexes (or *simplices*). A simplex generalizes the notion of an edge, which is a set of cardinality 2, to arbitrary cardinality sets. Similar to the way a weight can be attached to an edge in graphs to represent, say, the strength of a tie, a simplex may be *weighted* to denote some aspect of the larger aggregation. We provide more detailed and formal definitions in section 3.2.

Using traditional graph-theoretic metrics as well as those based on the higher-order simplices, we present an investigation along two broad lines: (i) structural properties of team science networks; and (ii) ways to measure the extent of collaboration within a team science network. Our analysis is based on two team science networks, both initiated by the Army Research Laboratory (ARL): the Network Science Collaborative Technology Alliance (NS-CTA), and the Communications and Networking Collaborative Technology Alliance (CN-CTA). Our contributions are as follows:

1. Using rigorous techniques developed in [19], we investigate the fit of *power law* to the distributions of vertex and facet degree (defined later) in the NS-CTA and CN-CTA, and show that a majority of them follow a power law with exponential cut-off.

2. We propose a metric called the *independence ratio* to measure the amount of simplex overlap, and compare the independence ratios of NS-CTA and CN-CTA.

3. We devise metrics to evaluate the *collaboration level*, both within a team, and between a team and other researchers. We apply these metrics to CN-CTA and NS-CTA to examine the effectiveness of these metrics, and provide an assessment of when and which metric to use.

The remainder of the paper is organized as follows. After introducing the terminologies and the datasets, in section 3.4, we study two structural aspects of the NS-CTA and CN-CTA. Section 3.5 presents metrics for assessing collaboration in team science networks and evaluates them on our two data sets. Related work is discussed in section 3.6, and we conclude in section 3.7.

3.2 Representing Groups

The use of graphs for representing collaboration networks in general and team science networks in particular does not capture the notion of groups, that is, a set of people working together. We therefore turn to using higher-order aggregations. A team science network is represented as *(V, S)*, where V is a set of vertices and S is a set of subsets s of V such that all elements of s collaborate in the same team. Each such subset s is called a *simplex*[1]. Any subset of a simplex is also a simplex. This captures nested sub-teams in a natural fashion, an aspect that will come in handy for assessing intra-team collaboration as a function of the output of sub-teams (section 3.5.1).

A non-empty subset of a simplex *s* is called *face* of *s*. The *dimension* of a simplex is one less than the number of vertices in it. Since these simplices can overlap or contain each other, we use the term *facet* to refer to a maximal simplex, i.e. a simplex that is not a subset of any other simplices. The *facet size* of a facet is the number of vertices comprising that facet. Similar to *vertex degree*, which is the number of edges incident to a vertex, the *facet degree* of a vertex is the number of facets that the vertex is a part of. For example, in a social network, the facet degree of a person is the number of maximal groups this person is in. In addition, a *minimal non-face* (MNF) is a set of vertices such that every subset except the set itself is a simplex. If an MNF has $k + 1$ vertices, it is called a k-minimal non-face (k-MNF). The use of these mathematical terminologies allows us to use other related concepts such as “face" and “MNF", eases the description in section 3.5.1, and facilitates future work building upon this paper that might leverage other metrics from the field of algebraic topology.

Figure 3.1 shows a simple collaboration network with three teams: {0, 1, 2}, {2, 3, 4}, and {1, 4, 5, 6}. The facet (team) sizes are 3, 3 and 4 respectively, and their dimensions are 2, 2, and 3 respectively. All subsets of the facets, including the facets themselves, are

[1]We borrow this and related terminologies later on from mathematics where it is used in a number of contexts to capture higher-order aggregations.

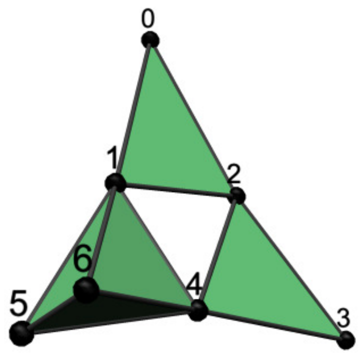


Figure 3.1: An example of collaboration network.

simplices and are shaded. The facet degree of vertex 4 (number of teams it belongs to) is 2 and that of vertex 5 is 1. Note that {1, 2, 4} is not a simplex even though {1, 2}, {1, 4} and {2, 4} are simplices. Instead, {1, 2, 4} is a 2-MNF. MNFs may be used to identify "missed" collaborations. Vertices 1, 2 and 4 are all collaborating pair-wise, which suggests a possible match in their interests. However, they miss an opportunity to combine their skills in a 3-way collaboration. A traditional graph-based model will fail to discern MNFs.

Finally, each simplex in a collaboration network can be assigned a label, indicating certain properties depending on specific problems. An example will be illustrated in the datasets we examine in the next section.

3.3 Datasets

We study two real-world team science networks provided by the Army Research Laboratory:

1. *NS-CTA*: The ARL Network Science Collaborative Technology Alliance (NS-CTA) is an ongoing program studying multi-genre networks [33]. The program is currently in its fourth year. In this paper, we work with data from the first three years of this program, including 599 publications from 631 authors. This paper itself is based on work that is

part of the NS-CTA.

2. *CN-CTA*: The ARL Communications and Networking Collaborative Technology Alliance (CN-CTA) was a research consortium for the purposes of developing advanced communications for the military. The program ran from fiscal year 2002 to 2009 and produced a total of 960 publications by 518 authors.

In each of these networks, researchers were organized into official teams. Teams were created to form self-defined tasks and submitted proposals, and managers selected a subset of the proposals. People in each team may work in sub-groups on different research topics, instead of working only together as a whole team in a single project. Futher, they typically publish papers with both researchers in their teams and other CTA and non-CTA collaborators.

We extract two collaboration networks from each of the above team science networks: a team network, and a paper co-authoring network. More specifically the team network captures the organization of researchers into the official teams. Each researcher is represented as a vertex, and a set of vertices forms a simplex if and only if the corresponding researchers are in the same team. On the other hand, the paper co-authoring network reflects the collaboration in terms of publishing. In this case, each researcher is still a vertex, but a set of vertices forms a simplex if and only if the corresponding researchers were co-authors on a published paper. Additionally, for the team network, each simplex is assigned a weight, which is the number of papers that the corresponding researchers co-authored together. From this point, we refer to the team network and paper network extracted from NS-CTA and CN-CTA as NSTeam, NSPaper, CNTeam and CNPaper respectively.

In the team networks, the facet degree of a vertex is the number of maximal teams that the corresponding researcher is a part of. For the paper networks, this is the number of distinct publication collaborations that the researcher participates in. Note that the facet degree for the paper network does *not* simply count the *total* number of papers written by the author, i.e., we

Table 3.1: Statistics for NS-CTA and CN-CTA team science networks. GC* stands for Giant Component.

	NS-CTA		CN-CTA	
Metrics	Team	Paper	Team	Paper
Number of vertices	109	631	148	518
Number of edges	871	2133	1536	1248
Max vertex degree	50	143	76	68
Average vertex degree	15.98	6.76	20.76	4.82
Num Components	2	11	1	16
%vertices in GC	96.3	87.0	100.0	87.5
Diameter of GC*	5	9	5	12
Number of facets	77	324	56	341
Max facet size	16	11	22	10
Average facet size	5.45	4.38	9.77	3.31
Max facet degree	13	66	22	62
Average facet degree	3.85	2.25	3.70	2.18
Number of 2-MNFs	789	139	577	51

don’t add up the weights.

3.4 Structural Properties

As mentioned in section 7.1, team science networks are unique in the way they are organized and evolve, such as the incentivization for diversity and light management. These factors are likely different for the team network and the paper network. In this section, taking the NS-CTA and CN-CTA as case studies, we investigate the structural properties of team and paper networks.

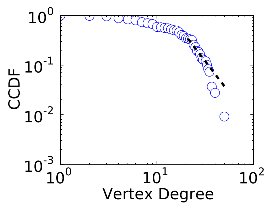
3.4.1 Overview of metrics for NS-CTA and CN-CTA networks

Table 3.1 summarizes the statistics of the NS-CTA and CN-CTA team and paper networks. The first set of rows shows traditional graph theoretic metrics and the second set of rows shows group-based metrics in terms of facets and MNFs.

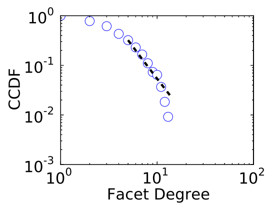
In both CTAs, the paper networks are 5-6 times larger than the corresponding team networks. This is because the paper networks also include non-CTA collaborators who are not part of the team networks. It appears that the paper networks are far more "spread out" than the team networks. This is indicated by significantly lower average vertex and facet degrees in the paper network, indicating less clustering, as well as significantly higher diameters and numbers of components. This could be due to the fact that the paper network has a bigger element of self-organization, and a large number of non-CTA members, bringing in a flavor of a general co-authorship networks like DBLP.

Recall that a 2-Minimal Non Face (2-MNF) is a set of vertices s of cardinality 3 such that every subset of s except s itself is a simplex. One interesting question is if the number of 2-MNFs are more dependent on network size, vertex degree or facet degree. Evidence from Table 3.1 indicates more support for the latter two. If we compare the 4 sets (NSTeam vs. CNTeam), (NSPaper vs. CNPaper), (NSTeam vs. NSPaper), and (CNTeam vs. CNPaper), we see that the number of 2-MNFs increases with increasing size for only one of them. Whereas, the number of 2-MNFs increases with increasing average vertex degree for 3 of them and with average facet degree for all four. It appears that the more tightly packed a network is, the more chances exist for a 2-MNF.

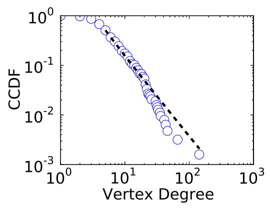
From the above, it appears that the team and paper networks, while comprising many same researchers, are somewhat different in their collaborative phenomena. One reason could be that the team networks were formed with some amount of management and incentivization, whereas the paper networks are drawn from pre-existing informal relationships prior to the team formation.



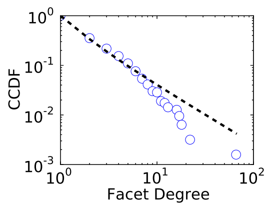
(a) NSTeam VD



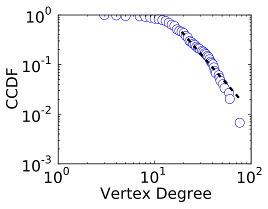
(b) NSTeam FD



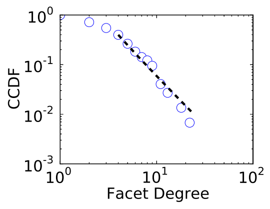
(c) NSPaper VD



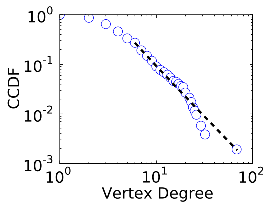
(d) NSPaper FD



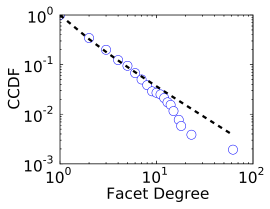
(e) CNTeam VD



(f) CNTeam FD



(g) CNPaper VD



(h) CNPaper FD

Figure 3.2: Vertex degree (VD) and facet degree (FD) distributions of NS-CTA and CN-CTA paper and team networks

3.4.2 Vertex and Facet Degrees: Power Law?

A remarkable feature of many natural phenomena is that the vertex degrees are power law distributed [19]. We now investigate if this property holds for team science networks, and whether it extends to facet degrees.

We adopt the rigorous method proposed by Clauset et al. [19] to decide if a distribution follows power law. Given a variable x, this method fits the complementary cumulative distribution function (CCDF) of the data to the Pareto cumulative distribution $P[X \geq x] \propto x^{\gamma}$, which is analogous to power law distribution. If plotted in a log-log scale, this distribution will appear as a straight line. We will report two parameters of the fitting obtained by their method: the exponent γ, and x_{min}. Since the power-law distribution often does not hold for the entire range of data, in particular not for smaller values, x_{min} is the smallest values of the corresponding variable x upon which the straight line power-law form still asserts itself.

Table 3.2: Power law fitting and likelihood ratio test results for NS-CTA and CN-CTA

Dataset	Power Law			Poisson		Log-normal		Exponential		Stretched exp.		Power law + cut-off		Support for
	x_{min}	γ	p	LR	p	LR	p	LR	p	LR	p	LR	p	power law
NSTeam VD	23 ± 4	5.1 ± 0.9	**0.10**	5.3	0.33	-0.9	0.44	-1.0	0.28	1.0	0.70	-1.0	0.15	good
NSTeam FD	5 ± 1	3 ± 1	0.07	-2.4	0.41	-2.9	0.12	-2.7	**0.02**	-3.0	**0.04**	-2.9	**0.02**	with cut-off
NSPaper VD	5 ± 1	2.6 ± 0.2	**0.14**	654	**0.01**	-2.5	0.21	30.6	**0.08**	33.7	**0.06**	-2.7	**0.02**	with cut-off
NSPaper FD	1.0 ± 0.8	2.2 ± 0.2	0.01	516	**0.00**	-5.7	**0.02**	104	**0.00**	-3.6	0.32	-6.1	**0.00**	with cut-off
CNTeam VD	19 ± 3	3.2 ± 0.3	0.05	133	**0.05**	-186	**0.00**	-157	**0.00**	-182	**0.00**	-183	**0.00**	with cut-off
CNTeam FD	4.0 ± 0.7	2.9 ± 0.3	**0.18**	24.4	0.36	-35.6	**0.00**	-37.0	**0.00**	-37.0	**0.00**	-37.0	**0.00**	with cut-off
CNPaper VD	6 ± 1	3.0 ± 0.2	**0.58**	275	**0.04**	-207	**0.00**	-179	**0.00**	-179	**0.00**	-179	**0.00**	with cut-off
CNPaper FD	1.0 ± 0.4	2.2 ± 0.1	**0.23**	441	**0.01**	-3.4	**0.09**	94.9	**0.00**	1.7	0.72	-3.4	**0.01**	with cut-off

The method proposed in [19] returns one of four judgments for a given data set: “none” indicates it is probably not power-law distributed; “moderate” indicates that the power law is a good fit but there are other plausible alternatives as well; “good” indicates that the power law is a good fit and that none of the alternatives considered is plausible; “with cut-off” indicates that the power law with exponential cutoff is clearly favored over the pure power law. Note that a power law distribution with an exponential cutoff has the following form:

$$P(z) \propto x^{-\gamma} e^{-x/x_c},$$

where γ and x_c are constants, and x is the variable in consideration, i.e., vertex or facet degree in our case.

The distributions of vertex degree (VD) and facet degree (FD) of NS-CTA and CN-CTA networks vis-a-vis a power law distribution are depicted in Figure 7.4(a). The fitting parameters for power law and the likelihood ratio test to compare power law distribution with other distributions are shown in Table 3.2. p-values are bolded in cases the corresponding tests are considered statistically significant: p-values greater than 0.1 for power law fitting, and less than 0.1 for log likelihood ratio tests [19]. If the log-likelihood ratios are positive, then the power-law model is favored over the alternatives. The last column lists judgment of the statistical support for the power-law hypothesis. Of the 8 combinations of {vertex-degree, facet-degree} $\times$ {team, paper} $\times$ {NS-CTA, CN-CTA}, 7 combinations can be deemed as power law with exponential cut off in the tail. One of them, the vertex degree of NSTeam, is a power law fit without cut-off.

For the vertex degree distributions of both NSTeam and CNTeam, the values of xmin are very high (23 ± 4 for NSTeam and 19 ± 3 for CNTeam) reducing the dependability of this assessment. However, the facet degree distributions display a power law with cut-off even with our strict methodology. Finally, we note that the two team science networks are quite similar

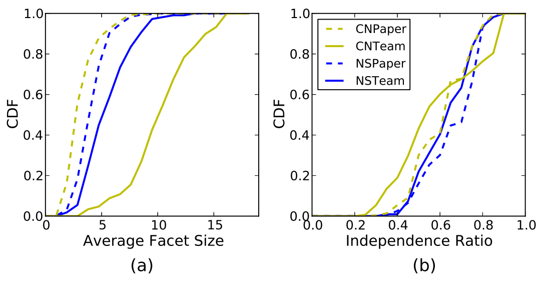


Figure 3.3: Cumulative Distribution Function (CDF) of average facet size per vertex and independence ratio for Team Science Networks.

to each other in terms of vertex and facet degree distribution.

3.4.3 Facet Degree and Size

The fact that vertex degrees and facet degrees show somewhat different distribution behavior above leads us to investigate the *overlap* among groups that a person participates in. Note that if there were no overlaps, and all facets were of equal size, then the vertex and facet degrees would correlate almost perfectly. Further, if there were overlaps but all facet sizes were the same, the ratio between the vertex and facet degrees would be a measure of the overlap. Since in real networks, facet sizes are clearly not single-valued, we construct the following new metric to measure the overlap:

Definition 3.4.1 *The* independence ratio $I(v)$ *of a vertex* v *is defined as the ratio between its vertex degree and sum of the sizes of all facets that contain* v.

Intuitively, the lower $I(v)$ is, the more facets incident to v overlap, i.e., the less independent the collaborating groups are. This measure is also technically valid for a graph, in which all simplices have dimension of at most one, but is trivial, namely $\frac{1}{2}$ for all vertices.

Figure 3.3 shows the cumulative distribution function (CDF) of the average facet size per vertex and the independence ratio for NS-CTA and CN-CTA networks, which clearly show that the two team science paper networks are similar to each other.

3.5 Metrics for Assessing Collaboration

While team science networks are formed with the goal of cross-fertilizing ideas from different fields, this in and of itself does not ensure that the members actually work together. Quantifying the level of collaboration within each team to assess if the team is functioning as intended would be valuable for managing team composition and network structure. In order to assess the level of interaction, we need to have an objective measure of the team's collaborative performance. For team science networks that are the focus of this paper, namely the NS-CTA and the CN-CTA, the number of publications jointly authored by the team members is a reasonable measure of their interactions.

Papers are seldom written by a team as a whole, but more typically by different sub-teams along topics of interest to the particular sub-team. Thus, we need a way to combine the collaborations between arbitrary subsets in the team into an overall assessment of a team's collaborative performance and, similarly, to assess the collaboration between a team and non-team members. In the remainder of this section, we devise metrics for evaluating the extent of collaboration of a team, both within itself – *intra-team collaboration*, and with other non-team members – *extra-team collaboration*.

3.5.1 Intra-Team Collaboration

We model each team as a simplex – in particular a facet – with the subset closure property implicit in the definition of the simplex elegantly capturing the sub-team collaborations. Recall that a non-empty subset of a simplex is termed the face of the simplex. To model the amount of collaboration, each face is assigned a non-negative weight indicating the collaborative performance, e.g., the number of papers published together by people in the face, termed the *collaborative performance output (CPO)* of the face. A higher CPO indicates more collaboration. Note that the CPO is assigned to both facets (teams) as well as non-facet simplices

Table 3.3: Five sample tasks in the NS-CTA, their CPO's (published paper counts) and the scores by each of the four proposed metrics.

Team	Members	Collaborative Faces and CPO's	lw(s)	ew(s)	dw(s)
T1	M01, M02, M03	{M02, M03 : 5}	2.5	5	2.5
T2	M04, M05, M06	{M04, M06 : 4} {M04, M05, M06 : 3}	5	10	5
T3	M07, M08, M09	{M09, M08 : 1} {M09, M07, M08 : 2}	2.5	5	2.5
T4	M10, M11, M12	{M11, M12 : 7} {M10, M12 : 3} {M11, M10, M12 : 2}	7	14	7
T5	M13, M14, M15, M16 M17, M18, M19	{M18, M13 : 1} {M17, M18 : 2} {M16, M15 : 1} {M19, M18 : 3} {M14, M16, M15 : 2} {M19, M18, M13 : 4} {M14, M16, M19, M18, M15 : 1}	3.83	9	0.84

(subsets of teams). If a face of dimension greater than 0 has a positive CPO, we call it a *collaborative face*. The collaborative faces with largest dimension in a simplex are termed the *largest collaborative faces*. Figure 3.4 shows two example facets: s_1 ={A, B, C, D, E} with 5 vertices and s_2 ={M, N, P, Q} with 4 vertices. The CPO's of their collaborative faces are shown in the adjacent top right table. {E,C}, for instance, is not a collaborative face. The largest collaborative faces are {A, B, C, D} with a CPO of 1 for s_1, and {M, N} with a CPO of 6 for s_2.

We quantify the collaboration among any subset of individuals comprising a team as the *collaborative score* of the corresponding simplex. The collaborative score of a simplex is defined as a function of the CPO's of its faces. Although the concept of a collaborative score is applicable to any non-facet simplex (sub-team), we are primarily interested in the collaborative score of a facet (team as a whole). In what follows, we consider the definition of collaborative score.

Let s denote the *target simplex* whose collaborative score needs to be evaluated. In our case, the simplices being scored are teams from the team datasets, while the paper datasets provide the CPO's of their faces. Denote by $F(s)$ the set of all collaborative faces, by $dim(s)$ the dimension of s. Let $c(k)$ denote the CPO of a simplex $k \subseteq s$. We use the following general rules of thumb to devise our metrics. Aspects that merit a higher score include:

1. A higher dimension of the largest collaborative faces in relation to the dimension of the target simplex.

2. The magnitude of collaboration, i.e., the CPO's of the faces within the target simplex.

We now describe the formulation of our proposed metrics. Each of the metrics biases one aspect over the others, and thus results in different assessment of the collaboration level. Note that the 0-dimensional faces, i.e., faces with a single vertex, do not affect the collaborative score of a simplex. For all metrics, simplices with higher collaborative scores exhibit a higher level of collaboration.

Definition 3.5.1 *The* linearly weighted collaborative score $lw(s)$ *of a simplex* s *is defined as:*

$$lw(s) = \frac{\sum_{k \in F(s)} dim(k) * c(k)}{dim(s)}$$

Here, the CPO's of the collaborative faces are weighted by their corresponding dimensions and normalized by the dimension of the target simplex.

One might argue that larger collaborations require more effort and should be suitably "rewarded" in the score. In the following two metrics, instead of linearly weighting the faces, we weight them by an exponential[2] function of the dimensions.

Definition 3.5.2 *The* exponentially weighted collaborative score $ew(s)$ *of a simplex* s *is defined as:*

$$ew(s) = \frac{\sum_{k \in F(s)} 2^{dim(k)} * c(k)}{dim(s)}$$

One problem with $ew(s)$ defined above is that the normalization is still linear in $dim(s)$, which tends to favor larger simplices. The following formulation "damps" the score exponentially using a function of the dimensions.

[2]We chose exponential based on our sense that for typical teams the difficulty of collaboration increased quite rapidly with the number of researchers. However, other functions, e.g. quadratic, are possible as well, and are a topic for future work.

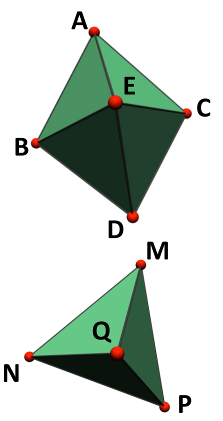
Definition 3.5.3 *The exponentially damped collaborative score $dw(s)$ of a simplex s is defined as:*

$$dw(s) = \sum_{f \in F(s)} 2^{dim(f)-dim(s)} * c(f)$$

For the example in Figure 3.4, we have: $dim(s_1) = 4$ and $dim(s_2) = 3$. As can be seen from the bottom right table of Figure 3.4, the scoring functions differ on which simplex is deemed the more collaborative. While $dw(s)$ deems (M, N, P, Q) to be more collaborative, $ew(s)$ deems (A, B, C, D, E) to be so, and $lw(s)$ assesses them to be equal. Each metric reflects a different emphasis on the CPO's – the number of papers – vesus the largest collaborative sub-team and the simplex dimension. We now evaluate these metrics on the team networks of NS-CTA and CN-CTA. While understanding the collaboration within these team science networks is certainly of interest, we also seek to answer the following questions: *Which of these metrics match "intuition"? How do these metrics correlate with each other? Are all of the three metrics useful? Which is the best metric for team science networks?*

We first assign collaborative performance outputs to the team simplices in each of NS-CTA and CN-CTA. The CPO's of a simplex in the team network is the number of papers published together by *exactly* the corresponding members in that team simplex. By "exactly", we mean that the highest-cardinality matching face is used. In other words, each paper is counted only once, for the maximal intersecting simplex between the paper simplex and the corresponding team simplex. For instance, if A, B, and C publish a paper together, it will only affect the CPO of the simplex {A, B, C}, but not the CPO of any subset or superset of {A, B, C}, such as {A, C}, {B, C}, {A, B}, and {A, B, C, D}. We now compute the three metrics defined above for each team as the target simplex.

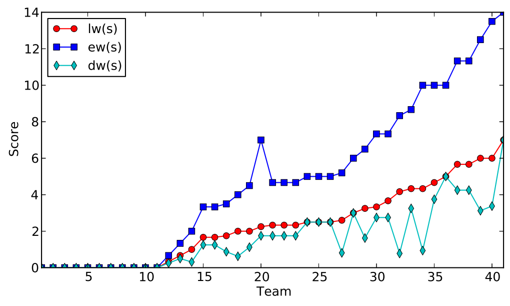
Table 3.3 illustrates each of the metrics for five anonymized sample teams within the NS-CTA. The second column contains the identifiers of the team members and the third column shows the CPO of each collaborative face in the team. The fourth column contains the largest



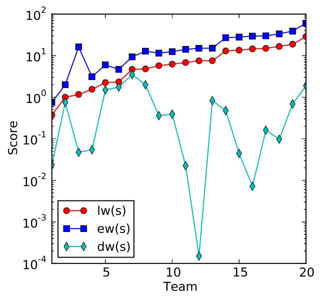
Face f	Collaborative Performance Output c(f)
{A, B}	1
{E, C, D}	2
{A, B, C, D}	1
{M, N}	6

Simplex s	lw(s)	ew(s)	dw(s)
s_1 = {A, B, C, D, E}	2	4.5	1.125
s_2 = {M, N, P, Q}	2	4	1.5

Figure 3.4: Example of two weighted simplices. The collaborative faces and their CPO's are given in the top right table. Other faces have CPO's equal zero. The CPO in this case is the number of papers published jointly by members of that face. The proposed collaborative scores are in the bottom right table.



(a) NS-CTA



(b) CN-CTA

Figure 3.5: Collaborative scores across teams for each of the metrics, with a view to discern correspondence in behavior.

collaborative dimension, followed by the three metrics $lw(s)$, $ew(s)$, and $dw(s)$. For each metric, we look at the rankings of the teams and consider if it matches our "intuition". All three metrics rank T4 and T2 as the most and second-most collaborative respectively. However, $dw(s)$ differs sharply in its evaluation of T5, which it rates last. This is because T5 is a high-dimensional group, and thus is penalized by $dw(s)$.

A complete picture of the relative values of the collaborative scores for all teams in NS-CTA and CN-CTA is shown in Figure 3.5. The teams on the x-axis are sorted in increasing order of $lw(s)$. Comparing the metrics, it can be seen that $lw(s)$ and $ew(s)$ "track" together, more or less, in the NS-CTA as well as CN-CTA. For the sample teams in Table 3.3, the rankings based on these two metrics are also the same (modulo tie breaking). To confirm this, we run a Spearman's correlation evaluation on these metrics. We observe a clear similarity between $ew(s)$ and $lw(s)$ with correlation coefficients of 0.99 (NS-CTA) and 0.84 (CN-CTA). Whether $ew(s)$ or $lw(s)$ is "better" depends upon how much value we give to higher-order collaborations. In our experience with team science research, the difficulty of collaboration between scientists (who are typically strongly opinionated!) does increase quite rapidly with size of the team. Thus, we recommend $ew(s)$ over $lw(s)$, i.e., we want to super-linearly reward higher-order collaborations.

In summary, we propose that $ew(s)$ and $dw(s)$ be used for evaluations. If the researchers have a lot of latitude in forming teams, and the alliance coordinators want to dissuade having members who are part of the team but hardly collaborate in papers, then $dw(s)$ should be used. On the other hand if the teams have been intentionally put together by the alliance coordinators, it does not make sense to penalize larger teams. Hence, $ew(s)$ should be preferred.

3.5.2 Extra-team Collaboration

A healthy collaboration profile should balance the number of collaborations with an individual, and the number of individuals with whom one has collaborated. In order to devise a metric for this purpose, we borrow from the popular concept of *H-index* [34], which is defined as follows: an author has an H-index of h if he/she has published h papers, each of which has been cited in other papers at least h times. In a similar way, we define the *collaboration index*, or *c-index* as follows:

Definition 3.5.4 *A simplex T has a c-*index *c if there are at least c non-simplex members, each of whom have collaborated with at least one member of T at least c times.*

The above definition easily instantiates to the case of an individual, which we shall refer to as *vertex c-index*. In other words, an individual has a vertex c-index c if there are at least c other individuals each of whom have collaborated with the individual at least c times.

We now evaluate these metrics on the NS-CTA and CN-CTA networks. Instead of all possible simplicies (sub-teams), we focus on the vertex (individual) and the team as organized officially in these two organizations. To compute the c-index of a vertex, we create an induced network from the paper dataset while using only the people present in the team dataset, i.e. only people officially in the NS-CTA and CN-CTA. For each person, we sort all of his/her neighbor vertices who have published at least a paper with him/her in decreasing order of the strength of collaboration between them–the edge weight–which is literally the number of common papers. The c-index of the corresponding vertex will be computed based on this sorted list of edge weights with respect to definition 3.5.4. The c-index of a team is computed similarly. Further, we normalize each team's c-index by the team size because c-index counts the collaborations with at least one member and hence by definition would be higher for larger team sizes.

Figure 3.6 shows the distribution of vertex c-index and normalized team c-index for NS-CTA and CN-CTA. These distributions are remarkably similar between the two networks. Given such similarity between two team science networks, we wonder if it might be that extra-team collaboration in team science networks has a certain “signature" in the way the collaborations are distributed. More data sets need to be analyzed to confirm this, but if so, it would provide a way to distinguish “anomalous" team science collaborations.

Finally, while there is a clear relationship between the vertex and facet degrees with Spearman's correlation coefficients greater than 0.82, there is no clear indication of such relationship between c-index and either vertex or facet degree. However, we observed that the person who ranks the highest in the number of collaborations also ranks the highest in vertex c-index, in both the NS-CTA and CN-CTA.

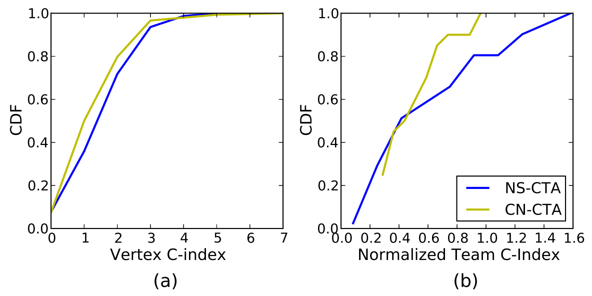


Figure 3.6: Cumulative Distribution Function (CDF) of (a) vertex c-index and (b) team c-index in NS-CTA and CN-CTA networks.

3.6 Related Work

The last decade has seen a spurt in the study of collaboration networks. In [11, 22], structural properties of publicly available scientific publication networks are analyzed, with the latter also investigating evolutionary aspects. Self-organization and classification into different kinds of small-world networks appear in [8, 24]. The quest to discern power laws in the distribution of social networks has been the subject of much work in the literature [19, 25]. Visual and other simplistic methods, however, may be misleading [19, 26]. A rigorous method for verifying if a distribution follows power law is given in [19], which we apply in this paper.

The scientific study of team science networks have recently received attention under the banner of ScITs (Science of Team Science) [31, 35]. Team formation has been studied in [36]. A study of the impact of the network structure on the performance of leaders and teams has been reported in [37, 38]. Results suggest that team performance is positively correlated with network density and team centrality. However, to our knowledge, the structural properties and quantitative collaboration performance metrics have not been studied.

Related to our work on collaboration levels is the concept of *cohesion* in social sciences literature, such as [39]. In contrast to these works, our goal is to assess whether and how well a team is collaborating based on its collaborative output, e.g., by joint analysis of team and paper networks.

Alternate representations for social collaboration networks to capture higher-order relations include the *bipartite graph* and *hypergraph* [32]. Our choice of the *simplex* representation is to reflect the natural closure property in the relation “is in the same team", and to facilitate future work building upon this paper that might leverage other metrics from the field of algebraic topology.

3.7 Concluding Remarks

We have studied two real-life team science networks: the ARL NS-CTA and CN-CTA. Using a more general representation than graphs – the simplex – we have analyzed metrics such as facet degree, facet size, independence ratio and minimal non-face in addition to the traditional graph-theoretic metrics. We have shown that the distributions of vertex and facet degrees resemble a power law with exponential cut-off.

We have investigated the problem of assessing intra-team and extra-team collaboration in team science networks and presented metrics for both intra-team and extra-team collaboration assessment. Evaluating these metrics on the NS-CTA and CN-CTA, we have examined and discussed these results vis-a-vis our intuition. Our analysis has helped provide recommendations on the situation under which each metric would be useful. Moreover, we have seen that the extra-team collaboration profile is remarkably similar in both networks.

The structural properties uncovered in this paper can be used as the basis for generative models of team science networks. Further, our proposed collaborative metrics can be applied to performance evaluation and improvement plan in team science networks.

Acknowledgment

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

Chapter 4

On Rising Stars in Dynamic Graphs

4.1 Introduction

Dynamic graphs can be used to represent many different types of networks in which the network entities are modeled as nodes and the relationships among them are modeled as time-varying links. Examples include online social networks, communication networks, and financial transactions. Mining and tracking the evolutionary patterns in such dynamic graphs is a problem of practical importance.

In this paper, we investigate nodes that behave anomalously in dynamic graphs, but with the important subtlety of looking at their evolutionary patterns with respect to their own and other nodes' trends. In general, this task can be done by tracking the evolution of each node's feature values on the graph over time. In particular, we propose the novel concept of **rising star** nodes, which simulaneously exhibit the following characteristics (formal definition is given in Section 4.2.1):

- $C1$: There is a sudden *increase* in the level of activities of the node.
- $C2$: Node properties change significantly with respect to the *history of the node itself*.

- $C3$: The change in node properties deviates significantly from the *global trend* of the network.
- $C4$: Node behaves differently from other *nodes with similar history*.

Characteristic $C1$ states that we are only interested in nodes that suddenly become more active than usual. In particular, we are concerned with finding rising stars in their early stages, that is nodes with low historical activity but quickly becoming very active. The early detection of such nodes can be useful in many ways. For example, the propagation of problematic events in a computer network can be detected and possibly isolated early before they become widespread. It must be noted that a very active node that suddenly becomes inactive might also be interesting. However, the type of nodes whose activities suddenly increase are more important and often require more attention from the analysis point of view. Thus, we focus on this type of rising stars with a sudden increase in activities.

Characteristic $C2$ asserts that a node's history must be taken into account when deciding if it is anomalous or not. This requirement sets our definition apart from the outlier detection techniques developed for static graphs [40–43]. A node might be extremely active at a specific time compared to other nodes in the network, but if it has always been that active, it would not be considered a rising star. Only if the change in node properties is unexpected given its history, would this node be a rising star.

$C3$ establishes that if the whole network has a global trend of increasing activities, then the bar for deciding if a node is a rising star should be raised as well. We thus propose to pinpoint a node as a rising star only if its behavior deviates significantly from this global trend. This characteristic distinguishes our definition from the common event detection problem in time series that only tracks the evolution of a single object—a node in our case.

$C4$ captures the fact that different groups of nodes may have different evolutionary patterns. Comparing a set of low-active nodes against a set of high-active nodes with regards to their recent changes will be unfair and possibly mask the true outliers within the low-active set. For

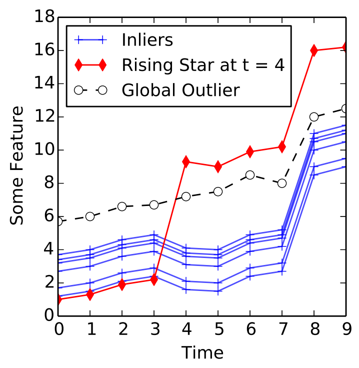


Figure 4.1: Example of a rising star: Each line is the evolution of node feature for a node over time. The red line is detected as a rising star at $t = 4$ but not at $t = 8$ when it is a global outlier.

example, in the DBLP bibliography dataset, it would be unfair to compare a professor with a prolific publication record with a new Ph.D. student. Such a new student should be matched against similar peers. Therefore, forcing a single unified gold standard on the whole population would not suffice. Instead, a rising star definition that differentiates nodes based on their similar recent history is required.

A specific yet simple example is shown in Figure 4.1, where the values of a fictional graph-feature (e.g., node degree) are tracked over time for 8 different nodes, resulting in the 8 lines in the figure. The six blue-plus lines have similar ranges of values and evolutionary patterns, and thus are *not* rising stars (inliers). On the other hand, the red-diamond line—the type of rising star we are interested in—has a distinctive evolutionary trend at $t = 4$. More specifically, it has a humble starting point among the objects with smallest feature values and is thus compared against the evolution of these nodes (Characteristic $C4$). After a short period of time, it quickly advances to be among the objects with highest feature values (Characteristic $C1$). Further, the node corresponding to the black-circle-dashed line is constantly more active than most other nodes and will always be identified as an anomaly from $t = 0$ to $t = 4$ by a traditional outlier detection technique (a global outlier). However, when the temporal aspect is considered, this node should not stand out as an entity of anomalous behavior since its evolutionary pattern does not have any sudden change and is very similar to that of the nodes with blue-plus lines

(Characteristic $C2$). Finally, the red-diamond line has another surge at $t = 8$, but it would not be a rising star at that time since such a surging behavior is also observed at all other nodes in the network at $t = 8$ (a global trend, Characteristic $C3$).

Our contributions in this paper are four-fold:

- We define a novel concept of "rising stars" in dynamic graphs that captures global network temporal behavior and compares nodes' temporal activity against others with similar historical patterns.
- We define "star creators" as nodes that significantly influence the emergence of many rising stars, and propose a Chernoff bound based approach for finding them.
- We empirically explore and analyze rising stars in three real-world dynamic graph data sets, using the nodes' features in their 1-hop neighborhoods (egonet). The empirical analysis demonstrates that our definitions of rising stars and star creators can effectively capture meaningful nodes with anomalous temporal behavior, which cannot be detected by snapshot-based outlier detection methods.
- We further analyze the topological and evolutionary patterns of the rising stars and star creators (and their relationship) detected in these graph data sets, drawing additional insights on how they evolve in real networks.

The remainder of paper is organized as follows: Section 4.2 gives the formal definition of rising stars and star creators. Section 4.3 studies the rising stars in real-world networks. Section 4.4 explores the evolutionary relationship among rising stars, star creators and "super stars". Section 4.5 discusses other subtleties of our proposed concepts. Finally, Section 4.6 covers related work and Section 4.7 concludes the chapter.

4.2 Rising stars and star creators in dynamic graphs

4.2.1 Definitions

We consider a dynamic graph G as a series of graph snapshots. Nodes and egdes can be added or deleted at each timestamp, and may have evolving weights. Edges are either undirected or directed. Denote a graph snapshot at time t as $G_t = (V_t, E_t)$, where V_t and E_t are the sets of nodes and edges at t.

Given a graph feature x, a node $v \in G_t$, we denote by $x_t(v)$ the value of feature x at timestamp t for v. Note that any structural and non-structural features can be used for our general notion of rising stars and star creators. In this paper, we use the egonet feature in our empirical analysis in Sections 4.3 and 4.4. Specifically, we track the egonet $Ego_t^v = (N_t^v, E_t^v)$ around v, where $N_t^v = \{v\} \cup \{u \in V_t | (u, v) \in E_t$ or $(v, u) \in E_t\}$ and $E_t^v = \{(u_1, u_2) | u_1, u_2 \in N_t^v; (u_1, u_2) \in E_t\}$, that is all nodes within v's one-hop neighborhood (including v itself) and all edges among these nodes. Table 4.1 lists five egonet features we investigate in this paper.

Table 4.1: Egonet features studied in this paper

Egonet feature	Explanation
Degree	Number of neighbors of the ego node
Weighted degree	Sum of edge weights between the ego node and its neighbors
Node weight	Weight of the ego node
Number of triangles	Number of triangles in the egonet
Egoweight	Sum of edge weights in the egonet

We now present the definition of rising stars, beginning by the temporal vector of the node feature.

Definition 4.2.1 *HISTORY VECTOR: Given a feature x, the history vector of a node v over a time window $[t_1, t_2]$ is define as:*

$$h^x_{t_1,t_2}(v) = [x_{t_1}(v), x_{t_1+1}(v), ..., x_{t_2}(v)]$$

where $x_t(n)$ is the value of x at timestamp t for node v.

Definition 4.2.2 *EQUIVALENT NODES: Two nodes u and v are said to be equivalent with regards to a feature x, a time window $[t_1, t_2]$, a distance function $\delta(.,.)$ operating on two history vectors, and a user-defined distance threshold Ω if $\delta(h^x_{t_1,t_2}(u), h^x_{t_1,t_2}(v)) \leq \Omega$. Then, we denote $u \equiv^{x,\Omega,\delta}_{t_1,t_2} v$, or $u \equiv_{t_1,t_2} v$ for short when x, δ and Ω are given.*

Intuitively, two nodes u and v are equivalent if they have similar history of changes in feature values over the given time period. Our assumption is that if two nodes are equivalent during a time window $[t_1, t_2]$, they are expected to be similar at timestamp $t_2 + 1$.

Thus, we define the "rising stars" as follows:

Definition 4.2.3 *RISING STARS: Given a time window size W and a user-defined parameter Θ, a node v is a rising stars at time t if*

$$x_t(v) > Q_3(S) + \Theta * IQR(S)$$

where $Q_1(S)$, $Q_3(S)$ and $IQR(S) = Q_3(S) - Q_1(S)$ are the first quartile, the third quartile and the inter-quartile range of the reference feature value set $S_t = \{x_t(u) | u \equiv_{t-W,t-1} v; u \in V_t\}$ respectively.

In essence, we compare the feature value of v at time t against the reference set S of feature values of all nodes that are equivalent to v during the history time window $[t-W, t-1]$. If the feature value of v is in the upper outlier range, i.e., $x_t(v) > Q_3(S) + \Theta * IQR(S)$, we call it a rising star, whose feature value increases suddenly and deviates significantly from the trend of equivalent nodes. In practice, $\Theta \geq 1.5$, where larger values are used for detecting rarer events. In our emprical study, we use $\Theta = 3$.

Critical for this definition is the selection of the distance function $\delta(.,.)$, such that it lead to nodes' equivalence in terms of capturing characteristics $C2$, $C3$, and $C4$, that is, nodes being compared against its own history, other nodes with similar history, and global trends (Since we only care about feature value in the upper outlier range, $C1$ is also met). We discuss this issue in depth in Section 4.2.2.

Next we define "star creators", whose influence in the emergence of rising stars is much more significant than other nodes'.

Definition 4.2.4 *STAR CREATORS: Given a small user-defined probability threshold* $q \in (0, 1]$ *and a user-defined deviation* $\Delta > 0$*, a node* v *is called a star creator if*

$$P[N_t^R(v) > (1 + \Delta)\mu_t^v] \leq q$$

where $N_t^R(v)$ *and* μ_t^v *are the number of rising star neighbors and the expected number of rising star neighbors of* v *at time* t *respectively.*

Intuitively, the star creators are the nodes whose number of rising star neighbors deviates too far from its expected value. To use this definition, we need to clarify how the parameters shall be chosen for the expected number of rising star neighbors μ_t^v, the probability thresholds q, and the deviation Δ. These are discussed in Section 4.2.3.

4.2.2 Equivalent nodes

To decide if two nodes are equivalent, any reasonable choice of the distance function $\delta(.,.)$ that captures the similarity of two history vectors can be used. A simple approach would be to use Euclidean or Manhattan distance in the space of $\mathbb{R}^W$, where W is the width of the history window. However, both Euclidean and Manhattan distances are too sensitive to noise in the history h and oblivious to recency of events. For example, consider three vectors $h_1 = [1, 2, 1, 1, 1, 1, 1]$, $h_2 = [1, 1, 2, 1, 1, 1, 1]$, and $h_3 = [1, 1, 1, 1, 1, 2, 1]$, which are the numbers of new scientific papers published by three different authors over a period of seven years. Clearly, h_1 should be considered to be more similar to h_2 than to h_3, since the differences between h_1 and h_2 happened in a more distant past. However, both Euclidean distance δ_{L_2} and Manhattan distance δ_{L_1} do not take this fact into consideration and produce the same distance values for $\delta(h_1, h_2)$ and $\delta(h_1, h_3)$, that is $\delta_{L_2}(h_1, h_2) = \delta_{L_2}(h_1, h_3) = \sqrt{2}$, and $\delta_{L_1}(h_1, h_2) = \delta_{L_1}(h_1, h_3) = 2$. Thus, we need to design a better distance function, which abides by the following principles:

- Principle A: δ should incorporate the recency of events, i.e., changes at timestamps closer to the current timestamp should be more important.
- Principle B: δ should capture the global trend of the whole network (to satisfy characteristic C3 in Section 4.1). If there is a sudden change in the feature values of all nodes in the network (a global trend), δ should adapt to this change.

Towards this task, we propose a mapping function that projects the space of history vectors onto a new 1-dimensional space, which satisfies the above principles, and the distance between history vectors to be computed as the difference between two mapped values.

Definition 4.2.5 *MAPPED HISTORY DISTANCE: Given a mapping function $f : \mathbb{R}^W \rightarrow \mathbb{R}$, and two history vector h_1 and h_2 over the same time window of size W, the mapped history distance between them is defined as:*

$$\delta_m(h_1, h_2) = |f(h_1) - f(h_2)|.$$

where $f(h_1)$ and $f(h_2)$ are called the mapped history. h_1 and h_2 are considered equivalent if $\delta_m(h_1, h_2) < \Omega$, where Ω is a small distance threshold. For a given node v, the tuple $(f(h_{t-W,t-1}(v)), x_t(v))$ is called the instant vector of node v at time t.

For the mapping function f, one may consider a local prediction model, such as an autoregressive model (AR, ARMA, ARIMA) or CUSUM, on the history vector of each node, as it can satisfy Principle A. More specifically, if the history vector $h_{t-W,t-1}(v)$ of a given node v is treated as a time series, we can learn an autoregressive model to predict the next timestamp

$x_t(v)$ in the series. However, this class of models is learned only on a single time series for each node v (hence we call it a local model), and is oblivious to the evolution of other nodes. If there are $N = |V_t|$ nodes in the network, we need to build N local models. Therefore, such models fail to capture the global trend of the network and do not satisfy Principle B.

Due to the above reasons, we propose to build a single global model for the whole set of nodes as the mapping function f instead. This global model still predicts x_t based on $h_{t-W,t-1}$ for each node, but is learned on the training set containing all pairs of $(h_{t-W,t-1}(v), x_t(v))$ $\forall v \in G_t$. Similar to a local model, a global model also automatically learns appropriate weights for each element of $h_{t-W,t-1}$, and thus captures the recency of events. However, different from N local models for N nodes, *only one single global model is built for all nodes in the network to capture the global trend at timestamp t, and thus helps the definition of rising stars satisfy Principle B and Characteristic $C3$.*

A side-effect of using a global prediction model as a mapping function is that the mapped history is interpretable: it is the expected value of the feature value *given that the model is correct*. If two nodes have similar history vectors, they are likely to have similar mapped history.

Note that any prediction model with good accuracy can be used as the global model. Thus, we opt for a general definition in this section, and discuss a specific example of a global model in our empirical study in Section 4.3.2.

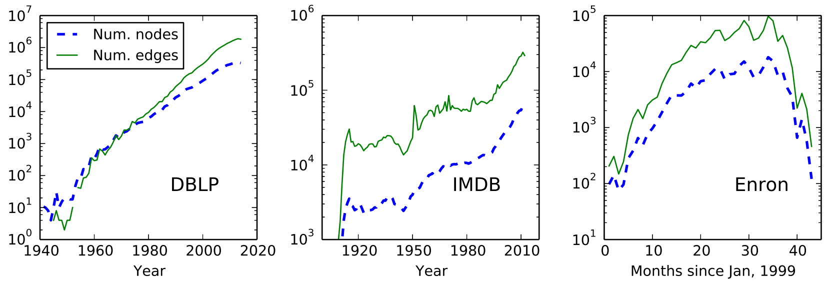


Figure 4.2: Number of active nodes and edges at each timestamp for our datasets

4.2.3 Chernoff-bound detection of star creators

In this section, we decide μ_t^v, q and Δ for Definition 4.2.4.

To compute the expected number of rising star neighbors μ_t^v of a node v at t, we assume nodes in the network become rising stars independently of each other. This assumption is a simplification of real-life situation and enables us to isolate neighborhoods with high number of rising stars compared to their density. Then, the probability that a random node in the network becomes a rising star at time t is $p_t = \frac{|V_t^R|}{|V_t|}$, where V_t and V_t^R are the set of nodes and the set of rising stars in the network snapshot G_t at time t. Now, for the egonet of a node v, each node u in its 1-hop neighborhood N_t^v will have the chance of becoming a rising star of p_t. Denote X_t^v as a random variable that is the number of rising stars in N_t^v. We can treat X_t^v as the sum of $|N_t^v|$ independent Bernoulli random variables (each variable for a neighbor of v), each having probability p_t of being equal to 1 (a neighbor is a rising star). Thus, its expected value is $\mu_t^v = E[X_t^v] = \sum_{u \in N_t^v} E[u \text{ is a rising star}] = |N_t^v| p_t$.

As per Chernoff bounds, the number of rising stars X_t^v in the neighborhood of v satisfies:

$$P[X_t^v > (1+\Delta)\mu_t^v] \leq e^{-\frac{\Delta^2 \mu_t^v}{2+\delta}} \tag{4.1}$$

where $\Delta \geq 0$ is a user-defined constant.

By setting the right hand side of Equation. 4.1 equal to the threshold q, that is $e^{-\frac{\Delta^2 \mu_t^v}{2+\Delta}} = q$, we can find the corresponding value of Δ. In particular,

$$\Delta^* = \frac{-\log q + \sqrt{\log q^2 - 8\mu_t^v \log q}}{2\mu_t^v} \tag{4.2}$$

What Chernoff bounds gives us is that the number of rising stars X_t^v in the egonet of v at time t will exceed $(1+\Delta^*)\mu_t^v$ with a probability less than q. Thus, for a small value of q, if $N_t^R(v) > (1+\Delta^*)\mu_t^v = (1+\Delta^*)\frac{|V_t^R|}{|V_t|}|N_t^v|$, we will mark v as an influential node, who happens to have too many rising stars in its neighborhood. We call such nodes the star creators since nodes who are connected to it tend to become rising stars much more than average.

To summarize, we can pick a small user-defined value for the probability threshold q, compute the corresponding Δ^* by Equation 4.2, and check if $N_t^R(v) > (1+\Delta^*)\frac{|V_t^R|}{|V_t|}|N_t^v|$ to decide if v is a star creator.

4.3 Rising stars in real-world networks

In this section, we explore the rising stars as defined in Definition 4.2.3 in three real-world datasets.

Algorithm 1 Finding stars at time t

```
Input: $W$; $\Omega$; $q$; $G_i = (V_i, E_i), i \in [t-W, t]$;
    Function $F_x$ that computes egonet feature $x$;
Output: Set of rising stars $V_t^R$, set of star creators $V_t^C$
 1: for $v \in V_t$ do
 2:     $x_t(v) \Leftarrow F_x(Ego_t^v)$
 3:     $h_{t-W,t-1}(v) \Leftarrow [F_x(Ego_{t-W}^v), ..., F_x(Ego_{t-1}^v)]$
 4: end for
 5: $lm \Leftarrow LinearRegression(x_t \sim h_{t-W,t-1})\ \forall v \in V_t$
 6: $f_v \Leftarrow lm(h_{t-W,t-1}(v)) \forall v \in V_t$
 7: $[m_1, m_2] \Leftarrow [min_{v \in V_t}(f_v), max_{v \in V_t}(f_v)]$
 8: $V_t^R \Leftarrow \emptyset$
 9: for $i = 0$ to $\frac{m_2 - m_1}{\epsilon}$ do
10:     $B_i \Leftarrow \{v \in V_t | f_v - m_1 \in [i\epsilon, (i+1)\epsilon]\}$
11:     $S \Leftarrow \left\{x_t(v) | f_v - m_1 \in \left[i\epsilon - \frac{\Omega}{2}, (i+1)\epsilon + \frac{\Omega}{2}\right]\right\}$
12:     $V_t^R \Leftarrow V_t^R \cup \{v \in B_i | x_t(v) > Q_3(S) + 3 * IQR(S)\}$
13: end for
14: $p_t \Leftarrow \frac{|V_t^R|}{|V_t|}$
15: $V_t^C \Leftarrow \emptyset$
16: for $v \in V_t$ do
17:     $\mu_t^v \Leftarrow p_t \times |N_t^v \cap V_t|$
18:     if $|N_t^v \cap V_t^R| > \left(1 + \frac{-\log q + \sqrt{\log q^2 - 8\mu_t^v \log q}}{2\mu_t^v}\right)\mu_t^v$ then
19:         $V_t^C \Leftarrow V_t^C \cup \{v\}$
20:     end if
21: end for
22: return $V_t^R$,$V_t^C$
```

4.3.1 Datasets

We study three different datasets[1]: the bibliography database DBLP, the Internet movie database IMDB, and the Enron email dataset. For DBLP and IMDB, we track the evolution of the co-authoring and co-starring networks over time, with a timestamp for each year. For Enron, we look at the email network with a timestamp per month. Each node is a researcher in DBLP, an actor or actress in IMDB, or an employee in Enron. The DBLP and IMDB networks are undirected: there is an edge between two persons if they collaborated in the same paper or movie. The edge weights are the number of times two corresponding persons worked together. The Enron email network, on the other hand, is directed: each email creates edges from the email sender to all receivers. Edge weights are the number of emails between two involving persons. Figure 4.2 summarizes the number of nodes and edges of the datasets over time.

4.3.2 Detection Procedure

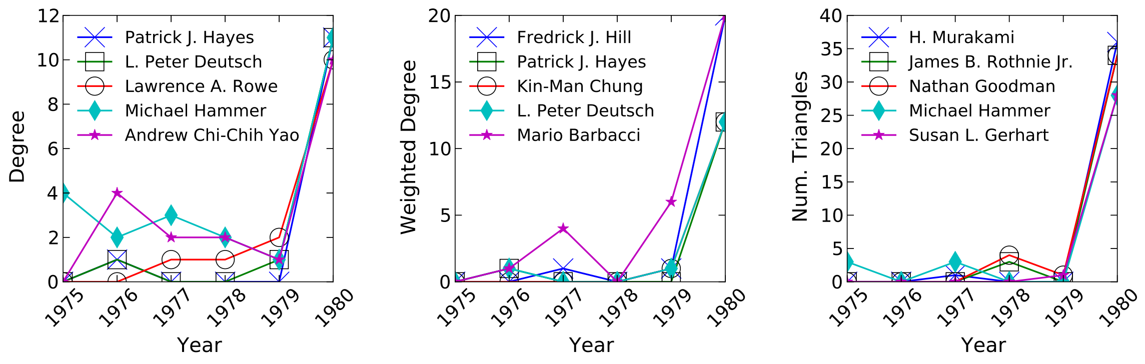
We consider the egonet features listed in Table 4.1 and adopt a linear regression model $lm(x_t \sim h_{t-W,t-1})$ as our mapping function $f(h)$ that predicts x_t given $h_{t-W,t-1}$, since it has a similar spirit as that of an autoregressive model for time series: $x'_t = \alpha_0 + \sum_{i=t-W}^{t-1} \alpha_i x_i$, where α_i are constants to be learned. In this case, our distance function becomes:

$$\delta_{lm}(h_1, h_2) = |lm(h_1) - lm(h_2)|$$

where lm is a linear regression model, taking a history vector h as input, and predicting the feature value x at the immediate next timestamp. Such a linear model can be learned efficiently and in a scalable manner for large datasets.

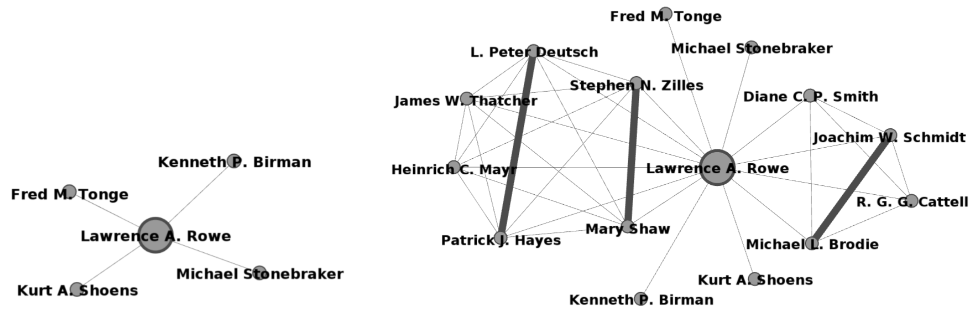
We discretize the prediction values of lm into bins of size $\epsilon = 10^{-5}$ to reduce computational overhead. In addition, $W = 5$ and $q = 0.01$. If $\delta_{lm}(h_1, h_2) < \Omega = 5$, h_1 and h_2 are equivalent.

[1]Code and data for this paper are available online at `http://www.cs.ucsb.edu/~mhoang/release/risingstars/`



(a) Degree (b) Weighted degree (c) Num Triangles

Figure 4.3: top-5 rising stars for different egonet features in DBLP, year 1980.



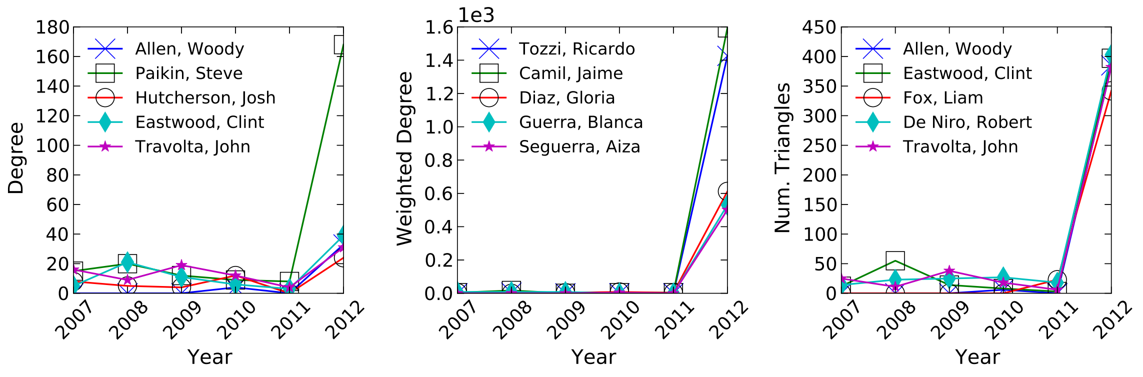
(a) Lawrence Rowe, 1979 (b) Lawrence Rowe, 1980

Figure 4.4: Egonets in 1979 and 1980 of Lawrence A. Rowe, who became a rising star in 1980. Edge weights are the number of co-authored papers between researchers.

Finally, Algorithm 1 summarizes the procedure for finding rising stars and star creators.

4.3.3 DBLP

The top-5 rising stars of DBLP in 1980 are reported in Figure 4.3 for different egonet features. We pick the year 1980, so that we can check if the detected rising stars have become “stars” today. Indeed, it can easily be verified that these people are now well-known computer scientists. Many of them are detected as rising stars across different egonet features. For example, Patrick J. Hayes, who has been an influential figure in Artificial Intelligence for over five decades, is detected in 1980 as a top-5 rising star for degree (Figure 4.3(a)) and weighted degree (Figure 4.3(b)). Similarly, our algorithm also found Lawrence A. Rowe (the



(a) Degree (b) Weighted degree (c) Num Triangles

Figure 4.5: top-5 rising stars for different egonet features in IMDB, year 2012.

founding director of the Berkeley Multimedia Research Center in 1995, Figure 4.3(a)), Michael Hammer (named by TIME as one of America's 25 most influential individuals, Figure 4.3(a) and Figure 4.3(c)), Andrew Yao (famous for the Yao's minimax principle, Figure 4.3(a)) and other famous scientists.

Mapping back to the graph, we provide in Figure 4.4 the egonets of one rising star in DBLP in 1980 (Figure 4.3(a)): Lawrence A. Rowe. As can be seen, there was a significant increase of activities in Rowe's egonet from 1979 to 1980.

4.3.4 IMDB

Figure 4.5 shows the top-5 rising stars in a single year 2012 for IMDB with respect to different features. It is obvious that these rising stars made a big "jump" in the year 2012 for the corresponding features.

For example, Steve Paikin (Figure 4.5(a)), who hosts a Canadian TV series *The Agenda with Steve Paikin* is spotted as a degree-rising star. This show started since 2006, but suddenly surged in 2012 after being restructured and became a hit. In this show, Steve Paikin is the host and interacts with many guest actors and actresses, leading to a sudden increase in his degree in the co-starring network. However, he did not work with the same people multiple times and thus is not in the top-10 rising stars of weighted degree. On the other hand, actress Aiza

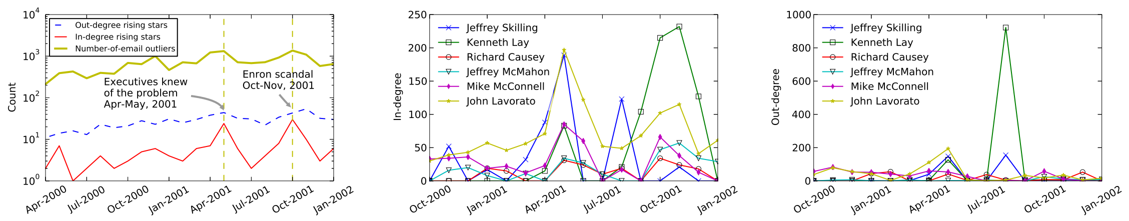


Figure 4.6: Rising stars in Enron: (a) Number of rising stars based on in-degree/out-degree vs. outliers based on email count over time; (b-c) In-degree/out-degree of some Enron executives over time. They had surging activities in (b-c) during the two turning points in (a).

Seguerra (Figure 4.5(b)), who was also on a TV series–*Be Careful with My Heart*–co-starred repeatedly with other people in the same series and thus stood out as a “weighted degree” rising star.

While the degree and weighted degree of a node show how well the corresponding person is connected with other people, the number of triangles and the egoweight indicate how often his or her collaborators interact with each other. Woody Allen (Figure 4.5(c)) is a very famous writer, director and actor, who has collaborated with many other people during his long and successful career. With his influence, he has also provided the opportunities for his collaborators to know each other, and open up future collaboration. The year 2012 marked a big increase in the activities in his egonet, showing that his influence helped create new connections in the co-starring network. Similar analyses are also true for other rising stars shown in Figure 4.5.

4.3.5 Enron

For the Enron email dataset, we know the timeline of the Enron scandal, which was revealed in October, 2001[2]. Therefore, the list of people who were responsible for this scandal, especially the Enron executives as listed in Table 4.2, can be used as a *ground truth* for rising stars. Specifically, we are interested in whether these people can be spotted as rising stars during the critical events for this scandal.

[2]`http://en.wikipedia.org/wiki/Enron_scandal`

Name	Rising stars (Year 2001)
Jeffrey K. Skilling	Mar, Apr, May
Kenneth L. Lay	May, Aug, Sep
Mike S. McConnell	May, Oct
John J. Lavorato	May
Jeffrey McMahon	May, Oct
Richard A. Causey	Jan, May, Jul, Oct, Dec

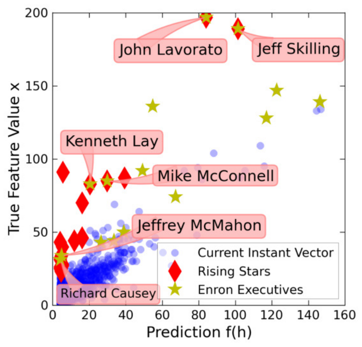
Table 4.2: Some Enron executives and when they are detected as rising stars based on in--degree/out-degree. May and October, 2001, were the two turning points (see Figure. 4.6(a)). Skilling and Lay were the two most responsible for Enron scandal and were among the earliest rising stars around the turning points.

To begin with, we look at the number of rising stars over time. Since our framework tries to find rising stars, the number of rising stars would likely surge when a big change is affecting the network, causing a sudden increase in the activities of many people. This phenomenon is confirmed in Figure 4.6(a). Two peaks in the number of outliers for the in-degree (number of email senders) and out-degree (number of email recipients) can easily be seen from this figure in May, 2001, and October, 2001. Based on the timeline of the Enron scandal, these two peaks correspond to two important events: (i) the executives knew of the problem but decided to "cook the books" during the period of April to May, 2001; and (ii) the Enron scandal was revealed in October, 2001.

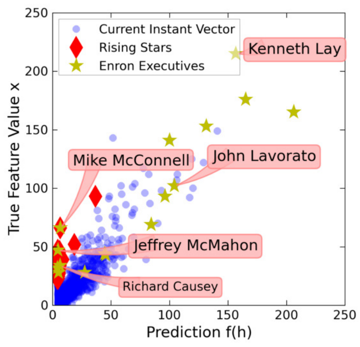
A closer look at the two peaks in the number of outliers reveals an interesting story. Figure 4.7 shows the instant vectors for the in-degree of nodes in the Enron email network at the peaks (the Enron executives listed in Table 4.2, as well as other executives, are marked as yellow stars). The second column in Table 4.2 also shows the months in which the executives are detected as rising stars. In addition, Figure 4.6(b) and 4.6(c) report the evolution of the in-degree and out-degree of some Enron executives who are identified as rising stars.

Two months prior to the first peak—March and April, 2001—Jeffrey Skilling suddenly stood out as a rising star although his activities were normal before that, as seen in Table 4.2 and Figure 4.6(b). On the other hand, most of other executives were still deemed normal. Note that Skilling is not identified by our method as a rising star before April, 2001. This observation indicates that Skilling was the first executive to know of the accounting problem in the company, leading to a significant increase in his email exchange. Indeed, he (as the CEO at this time) and Kenneth Lay (the new CEO after August, 2001) were the two most responsible persons for the scandal. After that, at the peak of May, 2001, most of the other executives also learned about this issue and became rising stars as well (Figure 4.7(a) and Table 4.2).

At the second peak in October, 2001 (Figure 4.7(b)), the highest executives (Skilling and Lay) already knew of the upcoming scandal and made necessary financial arrangement for themselves. While they still had high level of activities, they were not rising stars anymore because our method targets nodes with sudden increase in activities given low levels of activity before that. The rising stars now were other employees in the company, who learned about it later as the scandal was unfolding in October 2001, causing a big increase in email activities of many people, shown as the big group of red diamonds in Figure 4.7(b). Last but not least, as time goes by from May to October 2001 (Figure 4.7), the prediction $f(h)$ increased, suggesting that in general the volume of emails in the Enron network increased. However, the rising stars detected in these two months are very different, showing that our approach is effective as it does not merely look at volume of changes. To summarize, our definition of rising stars shows its strength in both detecting important events and capturing the initiators of such events.

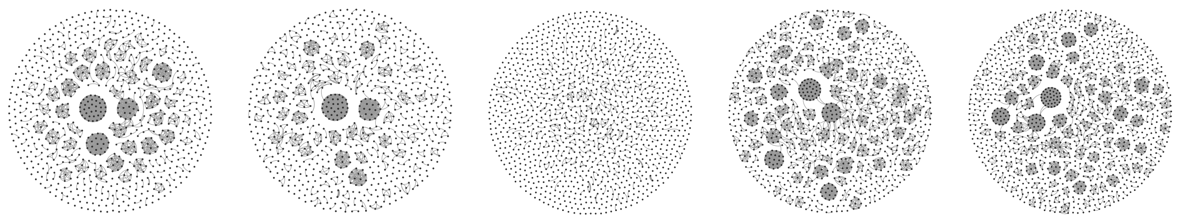


(a) May, 2001



(b) October, 2001

Figure 4.7: Rising stars in Enron during the peaks in Figure 4.6(a) for number of senders



(a) Degree (b) Weighted Degree (c) Node weight (d) Num. Triangles (e) Ego weight

Figure 4.8: Induced subgraphs of rising stars in DBLP (Year 2000).

4.4 Characterizing Rising stars and star creators

In this section, we provide further analyses on the graphical properties of the rising stars and the evolutionary relationship among the rising stars, star creators and super stars, the last of which is defined as:

Definition 4.4.1 *TOP-K SUPER STARS: The top-k super stars at time t for feature x are the top-k nodes with the highest values for x at t.*

4.4.1 Induced subgraphs of rising stars

Due to the collaborative nature of the nodes in DBLP and IMDB networks, an "event" (the appearance of a movie or a paper) affects the features of all nodes involved, especially when the event involves many individual nodes. Under that circumstance, we suspect that, for certain features in the egonet, there is a strong relationship between the rising stars in terms of their graph-topological positions. In particular, the question of interests is: *Are the set of rising stars more closely knit than a random subset of people of the same size?*

To investigate whether there is a strong coupling effect among the discovered rising stars, we first look at the induced subgraphs formed by the detected rising stars and examine their connectivity. Figure 4.8 shows the induced subgraphs of rising stars for five egonet features in DBLP network in the year 2000. Except for "node weight", the rising stars for all the other four features (degree, weighted degree, number of triangles, and ego weight) indeed exhibit strong clustering among them. This is because these four features are structural properties, and the structural changes introduced by rising stars affect these features of other nodes in their egonets as well. On the other hand, node weight is not a structural feature of an egonet and may not necessarily correlate with an emergence of rising stars in the egonet–the collaborators of a productive researcher with high node weight are not necessary productive as well. With this observation, we next evaluate this clustering effect quantitatively.

4.4.2 Clustering effect among rising stars

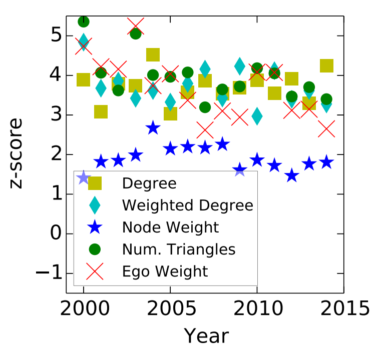
We use the decayed hitting time (DHT) [44] to quantitatively measure the extent to which the rising stars cluster and form small communities, by comparing the average DHT among the rising stars against that among a random subset of nodes in the same networks. More specifically, given a set of considered nodes in the network, multiple random walks are performed from each of these nodes, with a bound for the maximum number of steps, until another node in this set is visited. Intuitively, if the set of rising stars are more clustered than a random set of nodes, the average length of such random walks should be significantly lower for the set of the rising stars.

Given a set of nodes B, and a node $v \notin B$, [44] quantifies this effect by the decayed hitting time (DHT) defined by:

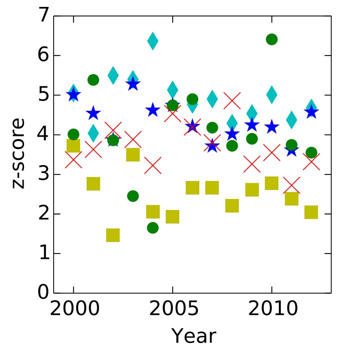
$$DHT(v, B) = \sum_{t=1}^{\infty} e^{-(t-1)} Pr(T_B = t | x_o = v)$$

where $Pr(T_B = t | x_o = v)$ is the probability that a random walk starting from v will hit a node in B after t step. The greater $DHT(v, B)$ is, the more likely that v will hit B.

What we are interested in, given a set of nodes V, is the average value of $DHT(v, V \setminus \{v\})$ over all nodes v in V, defined as follows:



(a) DBLP



(b) IMDB

Figure 4.9: z-score of average delayed hitting time of rising-star-induced subgraphs against random subgraphs of same sizes for each year 2000-2012.

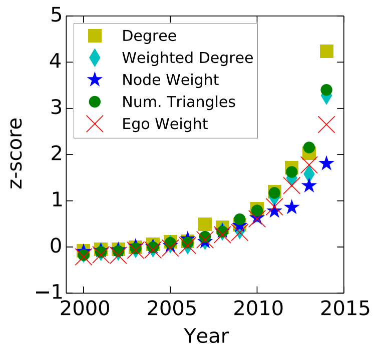
$$\rho(V) = \frac{\sum_{v \in V} DHT(v, V \setminus \{v\})}{|V|} \tag{4.3}$$

With this definition, we would like to verify if the average DHT of a subset of rising stars V^R is significantly higher than that of a random subset of nodes V^{rand} such that $|V^{rand}| = |V^R|$. We use the approximation algorithm in [44] to compute the z-score of $\rho(V^R)$ against $\rho(V^{rand})$:

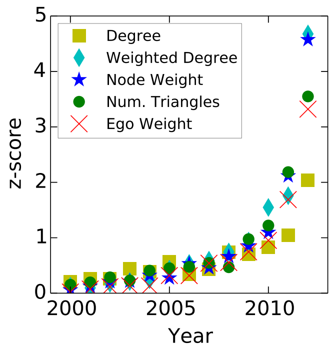
$$z\text{-score}(V^R) = \frac{\rho(V^R) - E[\rho(V^{rand})]}{\sqrt{Var[\rho(V^{rand})]}}$$

If $\rho(V^{rand})$ follows a Gaussian distribution, a z-score of 1.65 corresponds to a p-value < 0.05, signifying that $\rho(V^R)$ deviates significantly from the expected value of $\rho(V^{rand})$.

Figure 4.9 shows the z-scores for the rising stars in each year from 2000 to 2012 in DBLP and IMDB. As can be seen, the z-scores are in general much higher than 1.65, confirming our intuition that the rising stars often cluster together in the network. It must be noted that there is one big difference between DBLP and IMDB: the clustering effect is strongest for degree, and weakest for node weight in DBLP, while the reverse is true for IMDB. Node weight, however, is an exception because it is not a topological feature, and therefore the higher z-scores for node weight in IMDB are likely due to some other reason—perhaps a financial one: in order



(a) DBLP



(b) IMDB

Figure 4.10: z-score of average delayed hitting time of rising-star-induced subgraphs in DBLP 2014, and IMDB 2012 against random subgraphs of same sizes in years 2000-2014

for a movie/TV series to be popular and have a high expected return, actors/actresses that are growing into stardom (i.e., the rising stars, literally) are often cast together in it.

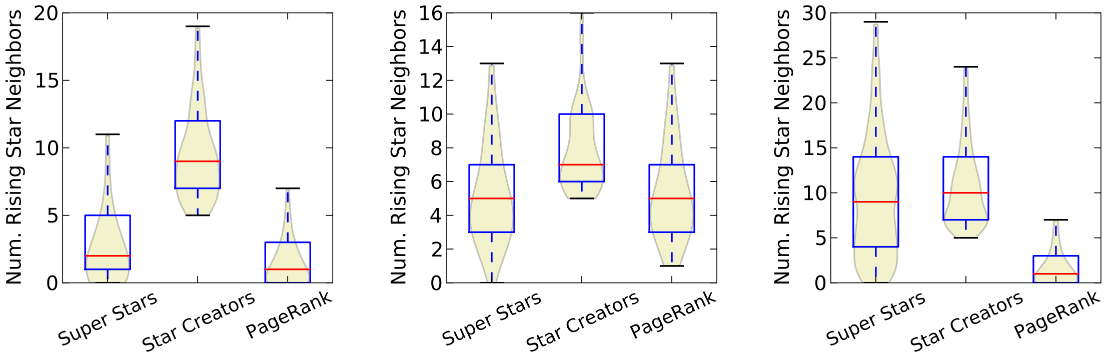
4.4.3 Evolution of the clustering effect

Given the fact that these rising stars are more tightly-knit than average, the next question is how such relationship evolves over time. To answer it, we investigate the evolution of z-score given a fixed set of rising stars. In particular, the evolutions of z-scores for the set of rising stars detected in 2014 for DBLP and 2012 for IMDB against random subgraphs of same sizes extracted from the corresponding network snapshots during the period 2000 to 2014 are shown in Figure 4.10.

Similar to Figure 4.9, the z-scores of the rising stars in the last years (2014 for DBLP, and 2012 for IMDB) in Figure 4.10 are very high. More importantly, the z-scores for these features increase over time until reaching their highest values in the years they are detected as rising stars. At first, the set of rising stars were just as clustered as a random subgraph in the network (year 2000). As time goes by, the future rising stars are gradually attracted to each other more so than other random nodes on their way to becoming ones. In other words, the rising stars

Star creators	$N_t(v)$	$N_t^R(v)$	Super stars	$N_t(v)$	$N_t^R(v)$	Top PageRank	$N_t(v)$	$N_t^R(v)$
Ian T. Foster	75	12	HongJiang Zhang	74	4	HongJiang Zhang	74	4
Nicholas Ayache	56	9	Thomas S. Huang	33	1	Ming-Yang Kao	40	0
Miguel Toro	34	6	Lajos Hanzo	23	0	Kang G. Shin	42	4
Hans-Peter Meinzer	34	7	Ming-Yang Kao	40	0	Ian T. Foster	75	12
Ya-Qin Zhang	33	7	Hong Yan	27	0	Alberto L. Sangiovanni-Vincentelli	54	3
Shih-Fu Chang	32	7	Mario Piattini	30	1	Ching Y. Suen	36	0
Stefan HaÃ§feld	32	9	Ian T. Foster	75	12	Heung-Yeung Shum	48	4
Robert Krempien	31	9	Mahmut T. Kandemir	28	0	Hong Yan	27	0
Marc Engels	31	6	Mario Gerla	41	4	Vladik Kreinovich	43	3
Heinz WÃűrn	30	9	Vladik Kreinovich	43	3	Ron Kikinis	58	3

Table 4.3: Top-10 star creators, super stars and PageRank nodes for feature node weight in DBLP, year 2001, and the number of rising stars in their egonet $N_t^R(v)$.



(a) Weighted Degree (b) Node Weight (c) Num. Triangles

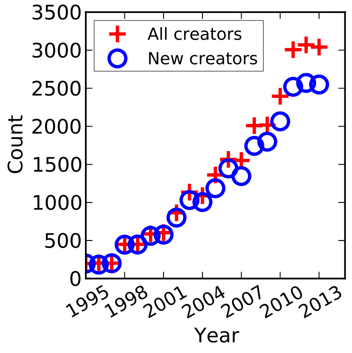
Figure 4.11: Star creators have more rising star neighbors than the top super stars and top–PageRank users in 2013.

evolve together over time.

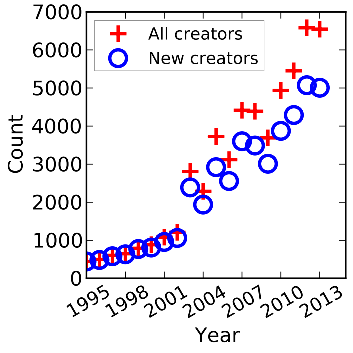
4.4.4 Star creators

In this section, we evaluate the star creators in DBLP by comparing them with two other types of influential nodes: (1) nodes with high PageRank [45] and (2) super stars. Note that nodes with high PageRank likely have high degree.

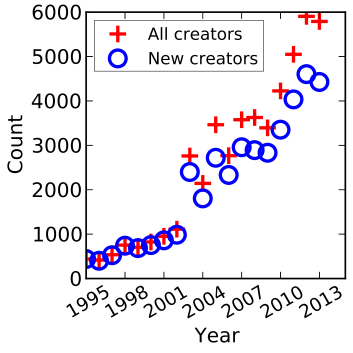
First, as defined in Section 4.2.3, star creators are nodes with much more rising star neighbors than other nodes in the network. Thus, we study the distributions of the number of rising stars within the 1-hop neighborhood for star creators, top super stars and top PageRank nodes. The results are shown in Figure 4.11 as violin plots (a box plot combined with a kernel density



(a) Degree

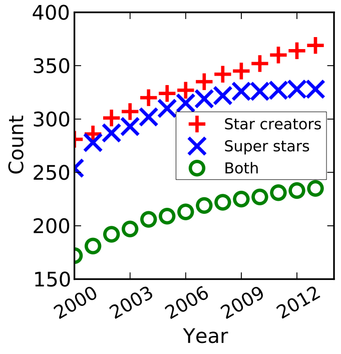


(b) Num. Triangles

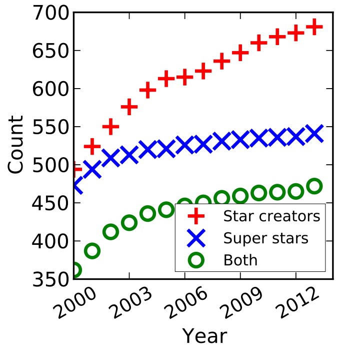


(c) Ego Weight

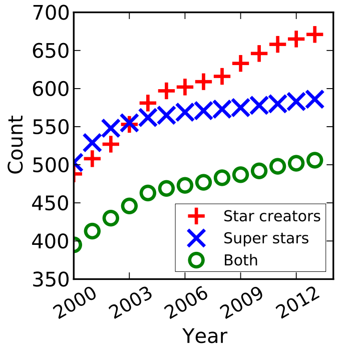
Figure 4.12: Number of star creators and new star creators (not a creator in the past) in each year.



(a) Degree

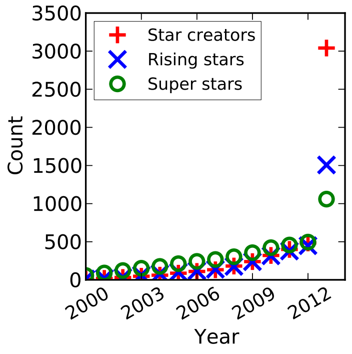


(b) Num. Triangles

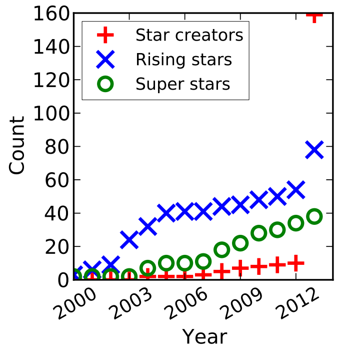


(c) Ego Weight

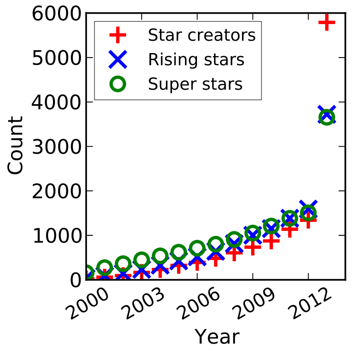
Figure 4.13: Rising stars in 2000, and who they became after that. Total number of rising stars: (a) 1521, (b) 4401, (c) 4475



(a) Degree

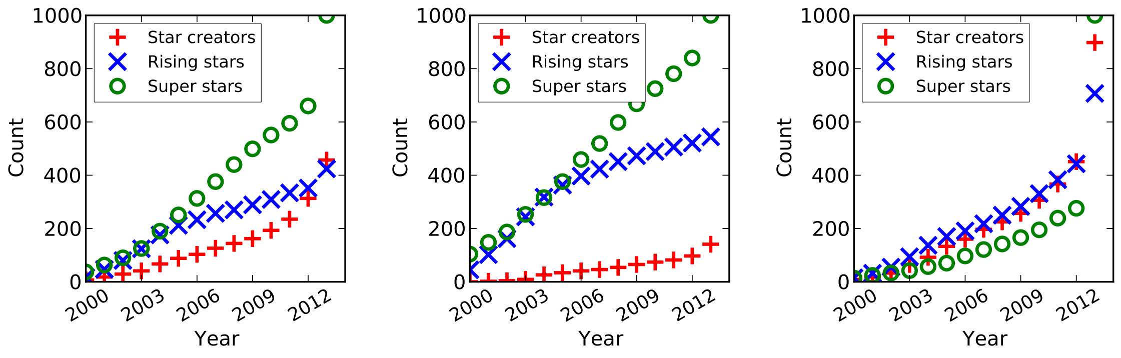


(b) Node Weight



(c) Ego Weight

Figure 4.14: Star creators in 2013, and who they were before.



(a) Degree (b) Node Weight (c) Ego Weight

Figure 4.15: top-1000 super stars (2013), and who they were before.

estimate of the probability density function). It can easily be seen that the star creators on average have more rising star neighbors across all five egonet features compared to both the super stars and top influential nodes based on PageRank. Additionally, the top-10 star creators, super stars and PageRank nodes for feature node weight in year 2001 of DBLP are shown in Table 4.3. As can be seen, while all three types of nodes cover prominent researchers, star creators have significantly more rising stars neighbors ($N_t^R(v)$) in year 2001 than super stars and top PageRank nodes. This observation suggests that influential nodes (super stars with highest feature values or top PageRank nodes with relatively highest number of neighbors) at each timestamp are not necessarily nodes that create many stars in their neighbors at that time.

Second, we ask if star creators are persistent in time, i.e., how likely a star creator at time t continues to create more rising stars in its neighbors at time $t' > t$. Figure 4.12 shows the number of star creators per year, as well as how many of the creators were not a creator ever before that (new star creators). As can be seen, while there are some persistent star creators over time, most star creators are indeed new. This observation suggests that there are some very prolific creators, but most of them are only temporarily active.

We next explore the evolution of rising stars and star creators with two questions: (1) *How many nodes became star creators and super stars after they were detected as rising stars?*; and

(2) *What types of nodes became star creators?*

For the first question, we track the set of rising stars in year 2000, and count how many of them became star creators, top super stars, or both, from 2000 to 2013. Figure 4.13 shows that a number of rising stars continued to be active and became more influential over time, that is they became super stars and star creators. However, most of the rising stars did not develop into stardom, which naturally reflects real life.

For the second question, we find the star creators in year 2013, and track their evolution from 2000 to 2013 to see who they became as time went by. As seen in Figure 4.14, there is a turning point right at the year 2013 when these star creators came into being. In this year, a large number of star creators were also rising stars and super stars. However, while some creators were also active in the past (past rising stars, super stars or creators), most of them only recently developed into stardom in 2013, which echos the observation in Figure 4.12.

Similarly, we look at the top-1000 super stars in 2013, and track their evolution from 2000 to 2013 in Figure 4.15. It can be seen that many super stars were also very active in the past. In addition, once a node became a super star, it likely continued to be a super star, as shown by the progress of the red circles over time in Figure 4.15. In other word, super stars likely persists over time.

To conclude this section, we note that a majority of star creators and super stars (>50% of each year) were rising stars at some point in the past.

4.5 Discussion

4.5.1 Comparison between quartile-based and density based approach

We compare our quartile-based definition of rising stars against Local Outlier Factor (LOF) [46], a density-based outlier detection that can be used to find outliers in the instant vector space

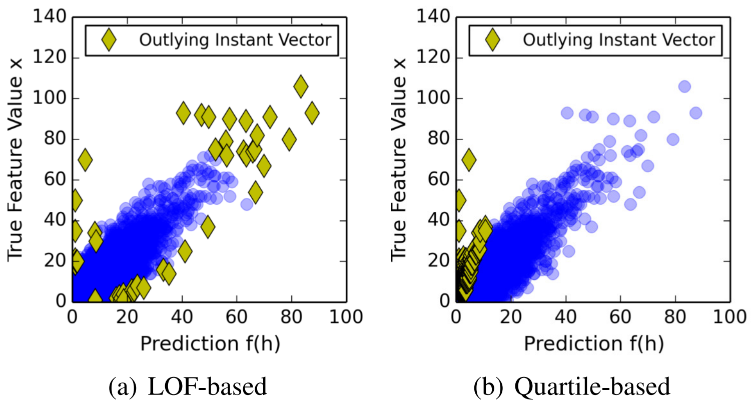


Figure 4.16: Top-50 outliers in the instant vector space $(f(h), x)$ for node weight using Local Outlier Factor (LOF) v.s. quantile-based approach (DBLP, node degree, year 2012).

$(f(h_{t-W,t-1}), x_t)$ at time t. Figure 4.16(a) plots the distribution of node degree against the mapped history using a linear regression model $f = lm$ for all nodes in DBLP (year 2000), and the top-50 LOF outliers (yellow diamonds in the figure). On one hand, many of the outliers have very small feature values, and thus are not rising stars as per Characteristic $C1$. On the other hand, many other nodes have high feature values, but these values do not deviate much from the predicted value. Thus, they do not satisfy Characteristic $C2$ and $C3$. Besides LOF, any outlier detection approach that merely focuses on the density in the instant vector space also has this problem, and therefore, is not a good candidate for finding rising stars. Figure 4.16(b) illustrates the same example as in Figure 4.16(a), but uses our quartile-based definition for rising stars. Now the outliers are true rising stars: true feature values deviate and are much higher than predicted feature values.

4.5.2 Comparison between history-based rising stars and feature-value-based outliers

Our rising star definition only compares egonets with similar history h. To better support this choice, we compare our rising stars with a simpler definition: outliers are the egonets with abnormally high feature values at a specific timestamp (feature-value-based definition). In particular, we look at the number of emails a person had during each month in the Enron datasets. A person is regarded as an outlier at a timestamp if the number of emails is more than $Q_3+3*IQR$, where Q_3 and IQR are the third quartile and the interquartile range. We compare these two different approaches in Figure 4.6(a). While there is a big overlap between the set of outliers identified by the two approaches, the number of outliers for the feature-value-based definition is significantly higher, making its utility significantly less. This definition does not take into account the different history of people. Therefore, people with high level of activity are always considered as outliers at each timestamp while they should not be identified as outliers over time. In this aspect, our method performs better, since it only compares objects with similar histories.

4.5.3 Capturing recent trends

Comparing an instant vector with other instant vectors within the same windows already captures the evolutionary trend at the last timestamp in the given time window. However, it does not capture the recent trend in the whole given time window. Capturing such a trend will have several benefits. First, not all nodes in the network follow the global shift in evolutionary pattern. Ignoring the recent past trend will likely lose this information. Second, since we only compare nodes with similar history vector, nodes in the sparse region of the instant vector space will have few equivalent nodes to be compared against. Consequently, the results for rising star detection in such regions will be less stable. Adding the recent past instant vectors to the pool for comparison will give us more reference points, leading to more stable results. Thus, in our empirical analysis, instead of merely comparing among the instant vectors at timestamp t over the time windows $[t-W, t]$, we also add to the reference pool the past instant vectors from timestamp $t-W$ to timestamp $t-1$. The reference feature value set at time t in Definition 4.2.3

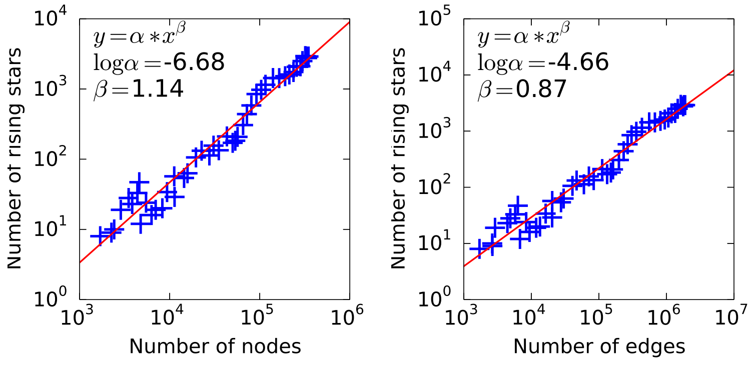


Figure 4.17: The power law between the number of nodes, edges and rising stars for feature node degree in DBLP. Each plus sign is for one year. α and β are fitting parameters for red solid lines.

now becomes:

$$\hat{S}_t = \{x_{t_i} | t - W \leq t_i \leq t - 1; m \equiv_{t_i - W, t_i - 1} n; m \in V_{t_i}\}$$

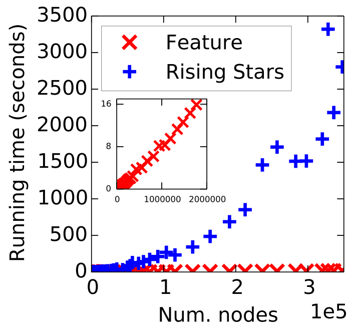
4.5.4 Evolution of number of rising stars

Claim 1 *POWER LAW OF RISING STARS: The number of rising star r_t, the number of nodes n_t, and the number of edges e_t at time t in a dynamic network follow a power law as the network evolves: $r_t \propto n_t^{\beta_{n,r}}$ and $r_t \propto e_t^{\beta_{e,r}}$.*

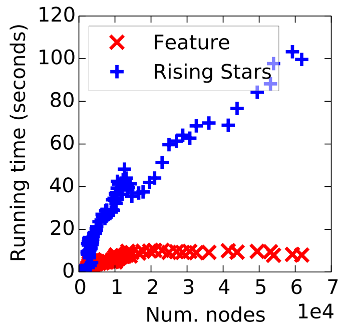
Claim. 1 is validated by empirical results in Figure 4.17, which contains the fitting results for rising stars based on node degree. The numbers of rising stars show clear growth patterns according to power laws, albeit with varying degrees in terms of growth rate. This observation is only a preliminary result and deserves more future research effort.

4.5.5 Computational Complexity

Our method for detecting rising stars can be divided into two steps: (1) Computation of egonet features for all nodes in a graph and (2) quartile-based outlier detection. As proved



(a) DBLP



(b) IMDB

Figure 4.18: Running time vs. number of nodes over time. The inset plot in Figure 4.18(a) shows the feature computing time vs. number of nodes for clearer visualization.

in [47], the computation of egonet features takes $O(N)$, where N is the number of nodes, for real-wolrd graphs. The linear regression model can be learned in $O(W^2N) \approx O(N)$, where W is the width of the time windows. After that, all nodes are sorted to find the quartiles for rising star detection, which takes at most $O(N \log N)$. Thus, in total, for each time step, the complexity is $O(N \log N)$, and our solution is thus scalable for large datasets. Figure 4.18 shows the running time of our framework (feature computation and rising star detection) against the number of nodes over time for two datasets DBLP and IMDP. As can easily be seen, the running time is within the bound of $O(N \log N)$. It is noteworthy that the running time of our method can be reduced further since the outlier analysis can be split and performed in parallel for each bucket.

4.6 Related Work

Our work can be put in the general theme of outlier detection in graphs. In [40], authors establish several laws governing the relation between different features of an egonet in static graphs, and uses them for outlier detection. Another body of work pays more attention to the structural pattern of the network, including [41–43]. If one applies methods designed for static methods in dynamic graphs, however, an egonet that is an inlier in a certain snapshot of the graph may stand out as an outlier if its evolutionary pattern over time is taken into account.

For dynamic graphs, a body of work deals with finding events in the evolution of the whole graph [48–50], i.e., time points when the structure of the graph changes significantly. As a by-product, the nodes or edges that possibly account for the event may also be found. [51] looks at individual edges in a network and compares them against the average behavior of all edges to find outliers, yet overlooking the graph aspects of the problem. [52] facilitates reservoir sampling to find abnormal edges by constructing a probabilistic model based on the connectivity pattern among nodes.

Most similar to our work is the detection of node outliers from dynamic graphs by tracking their egonet features [49, 53]. The main focus of these papers, however, is not the definition of outliers, but rather how to compute a set of features that capture the evolution patterns of nodes in a dynamic network. Our work, on the other hand, defines the novel concepts of rising stars and star creators. The features used in [49, 53] can be plugged into our framework and used for the detection of rising stars as well. Furthermore, we focus on the next step of empirically analyzing the evolutionary behavior of rising stars, star creators and super stars over time, instead of just finding outliers as in these papers.

4.7 Conclusion

In this paper, we define two novel concepts in dynamic graphs: rising stars and star creators. In particular, rising stars are nodes that suddenly become very active, compared to the history of both itself and other nodes, and often signify critical events in the graphs. By empirically studying three real-world networks (DBLP, IMDB and Enron emails), we find meaningful rising stars in their early stage of emergence. Furthermore, we perform deep analysis of the evolution of rising stars, star creators and the relationship among them in the dynamic graphs.

We discover interesting topological evolutionary patterns of these rising stars and star creators, opening up possible future research directions.

Part II

Mining Network Patterns to Summarize Network Processes

After having verified that the network structures do impact different networked behaviors, the next step is to mine network patterns that best summarize such an impact.

In Chapter 5, I summarize different processes in a networks by a small yet interpretable set of network patterns, each of which represents a local community of connected nodes frequently participating in the same network processes. I formulate this problem as a Binary Matrix Factorization with a network constraint, which I prove to be NP-hard. I then propose a greedy algorithm that incrementally adds the best patterns and make it more scalable with two further improvements. First, to decide which network processes contain which network patterns, I introduce two mapping algorithms with linear costs. Second, to systematically mine the exponential subgraph search space for good patterns, I devise two sampling algorithms based on Monte Carlo Markov Chain sampling. Experimental results on both synthetic and real-world datasets show that our solutions are scalable and find network patterns that effectively summarize network processes.

In Chapter 6, given a function that classifies a data object as relevant or irrelevant, I consider the task of selecting k objects that best represent all relevant objects in the underlying database. This problem occurs naturally when analysts want to familiarize themselves with the relevant objects in a database using a small set of k exemplars. In this paper, I solve the problem of *top-k representative queries* on graph databases. While graph databases model a wide range of scientific data, solving the problem in the context of graphs presents us with unique challenges due to the inherent complexity of matching structures. Furthermore, top-k representative queries map to the classic Set Cover problem, making it NP-hard. To overcome these challenges, I develop a greedy approximation with theoretical guarantees on the quality of the answer set, noting that a better approximation is not feasible in polynomial time. To further optimize the quadratic computational cost of the greedy algorithm, I propose an index structure called *NB-Index* to index the θ-neighborhoods of the database graphs by employing a novel combination of Lipschitz embedding and agglomerative clustering. Extensive experiments on real graph datasets validate the efficiency and effectiveness of the proposed techniques, which achieve up to two orders of magnitude speed-up over state-of-the-art algorithms.

In Chapter 7, I extract discriminative subgraphs from global-state networks. *Global-state networks* provide a powerful mechanism to model the increasing heterogeneity in data generated by current systems. Such a network comprises of a series of network snapshots with dynamic local states at nodes, and a global network state indicating the occurrence of an event. Mining discriminative subgraphs from global-state networks allows us to identify the influential sub-networks that have maximum impact on the global state and unearth the complex relationships between the local entities of a network and their collective behavior. In this paper, I explore this problem and design a technique called *MINDS* to mine *minimally discriminative* subgraphs from large global-state networks. To combat the exponential subgraph search space, I derive the concept of an *edit map* and perform Metropolis Hastings sampling on the map to compute the answer set. Furthermore, I formulate the idea of *network-constrained decision trees* to learn prediction models on a subgraph without compromising on the underlying network structure. Extensive experiments on real datasets demonstrate excellent accuracy in terms of prediction quality. Additionally, MINDS achieves a speed-up of at least four orders of magnitude over baseline techniques.

Chapter 5

Summarizing Network Processes with Network-constrained Binary Matrix Factorization

5.1 Introduction

Network structures exist in many different types of real-world data and play an important role in guiding various processes happening in the data. Some example network processes are: explicit and implicit information spreads in social networks, a cascade of function calls triggered by a program in a distributed system, congestion in traffic networks, the firing of brain regions in a brain network when a person performs a task, and the spread of failures in computer networks. In all of these scenarios, the network processes are affected by two main sources: (i) exogenous factors, and (ii) the network structure. For example, within the Twitter social network, a user will be exposed to a piece of news either by reading an external news website, or by reading the tweets of his/her friends. It must be emphasized that the information source and the direction of a process within a network are not always available explicitly. Specifically, we can determine without a doubt from who a user retweets in Twitter. However, this is not the case for the spread of a hashtag. If multiple friends of a user have used this hashtag before he/she does, we cannot attribute exactly from who this user learns about this hashtag.

In order to understand and analyze the network effect, the first step is to isolate repeated network patterns in the complex network processes. Each such network pattern represents a local community of connected nodes with similar dynamic behaviors. Identifying these patterns is helpful for understanding, controlling and predicting the behaviors of future network processes. Table 5.1 outlines the patterns we can mine from different network processes and their potential usage for various management, optimization, and analysis tasks.

In this paper, we *find the top-k network patterns to best summarize a set of network processes.* We define a network pattern as a connected subnetwork to capture the effect of the network structure on the progress of network processes. A pattern is essentially a local community of nodes who strongly and repeatedly affect each other's behaviors in various network processes due to their network connections. Such a pattern looks beyond individual active nodes and pinpoints the different impacts of different parts of the network on a network process. Fig. 5.1 shows an example network pattern in Twitter and the spreads of 6 Iran-related hashtags within this pattern. Here, users that participated earlier in the spread are coded as redder and bigger nodes; the directed edges point from earlier to later participating nodes. While the network structure guides the spread of the hashtags, it is clear from Fig. 5.1 that the order of infection and thus the edge directions are vastly different among different hashtags. Such an observation is prevalent in real-world data due to the randomness in user behaviors, such as when a user is active online or what external websites a user visits. Thus, if the exact order of infection and the edge directions of infection are included in the pattern definition, it would lead to small or infrequent patterns. Moreover, the edge directions are not available in many scenarios: e.g., implicit information spreads and brain network examples in Table 5.1.

Random node behaviors also make it harder for exact matching of network patterns into

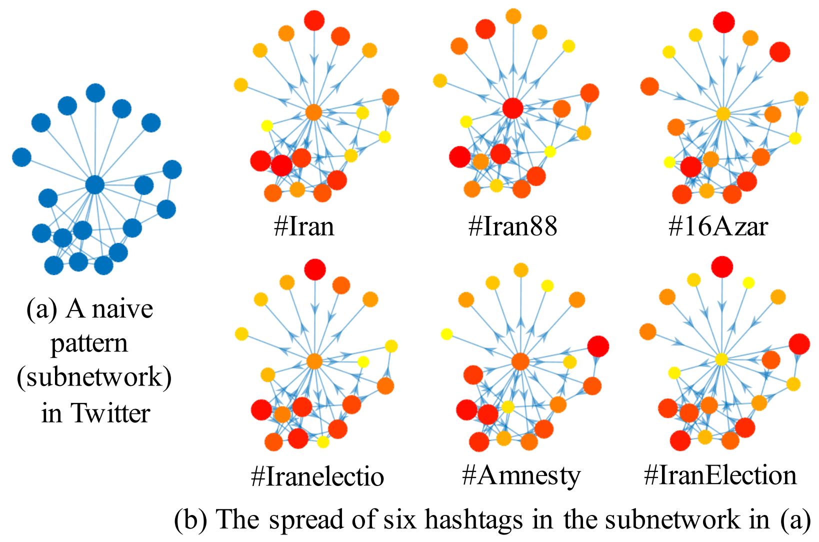

(a) A naive pattern (subnetwork) in Twitter
#Iran
#Iran88
#16Azar
#Iranelectio
#Amnesty
#IranElection
(b) The spread of six hashtags in the subnetwork in (a)

Figure 5.1: The spreads of 6 Iran-related hashtags in a Twitter's subnetwork. Bigger/redder nodes (users) participated earlier in the spread. Edges are directed from ealier to later participating nodes.

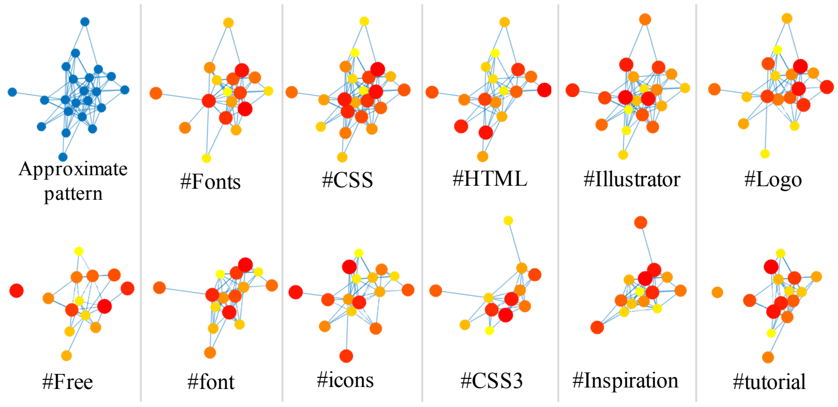


Figure 5.2: An approximate pattern in Twitter and the spreads of design-related hashtags in this pattern. Bigger/redder nodes (users) participated earlier in the spread. Edge directions are omitted for clarity.

Examples	**1) Explicit information spreads**	**2) Implicit information spread**	**3) Function call analysis**	**4) Traffic network analysis**	**5) Brain network**	**6) Computer network**
Nodes	Users	Users	Functions	Road segments	Brain regions	Computer servers
Edges	Friendship	Friendship	Call relationship	Intersections	Correlations	Connections
Network process	Explicit information spreads (known source and target nodes), e.g., retweets on Twitter.	Implicit information spreads (unknown source nodes), e.g., hashtag usage on Twitter.	Function calls	Spread of traffic jam	Activity of brain regions during a task.	Spread of failure or overload
Network state at time t	Directed graph among infected nodes	Set of infected nodes	Directed graph of function calls	Set of jammed road segments	Set of active regions	Set of failed, overloaded servers
Network pattern	A frequent path of spread.	A local connected community that frequently reacts to the same content.	A frequent function call path.	A set of frequently jammed connected road segments.	A set of brain regions active together during a task.	A set of connected servers that often fail together.
Pattern usage	Target and control information spread.	Target and control information spread.	Optimize frequent paths to reduce latency.	Redesign the patterns to reduce traffic jams.	Understand and optimize brain activity.	Redesign network to avoid the failed patterns.

Table 5.1: Example applications of network-constrained patterns.

network processes. A single mismatched node can cause the network process not to contain the pattern, which is understandably common in real-world data, especially in social media data and brain network data (it is not easy for a subject to focus on a single task when his/her brain activity is recorded, leading to noisy data). As a result, it is reasonable to adopt an approximate matching scheme that allows some nodes in a pattern to be missing in the network processes, as shown in the example in Fig. 5.2.

From the above observations, we aim to find network patterns that represent connected local communities that often react to the same stimuli with the following six characteristics:

- C1. Best summarize the network processes.
- C2. Correspond to connected subgraphs of G.
- C3. Appear frequently in the network processes.
- C4. Have considerable sizes (number of nodes).
- C5. Robust to noise in which nodes get infected.
- C6. Robust to noise in the order of infection.

Here, C1 naturally targets our summarizing goals, C2 captures the network effect, C3 and C4 guarantee that the obtained patterns are significant: i.e., they are large and connected subnetworks that appear frequently in different network processes. Finally, C5 and C6 deal with the fact that network processes in real life are highly stochastic: seldom do the exact same set of nodes get infected repeatedly and in the exact same order. In summary, we restrict our patterns to be *big, frequent, connected, and undirected subnetworks*.

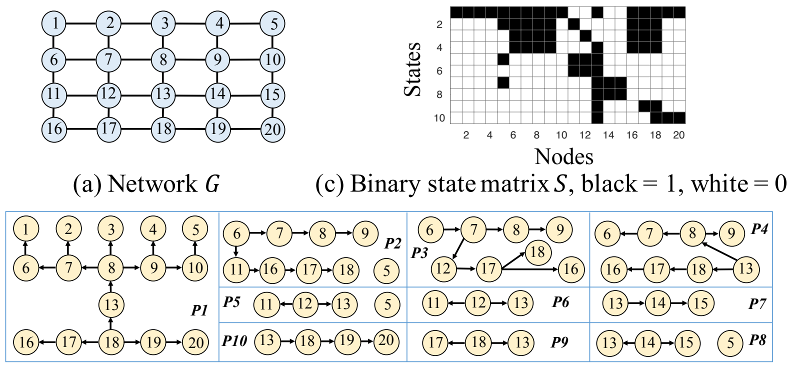
Our contributions are as follows:

- We introduce network-constrained Binary Matrix Factorization (netBMF) as a novel data mining challenge to find top-k network patterns to summarize complex network processes.
- We prove that netBMF is NP-hard and then propose a greedy solution. We next improve the greedy algorithm into four scalable variants by solving two sub-problems as shown in Fig. 5.7. First, we propose two linear algorithms to efficiently map network patterns to network processes. Second, we design two MCMC sampling algorithms to cope with the exponential subgraph space of a network.
- Extensive experimental results on both synthetic and three real-world datasets indicate that our proposed solutions are scalable and achieve a good trade-off between interpretability and accuracy compared to other baselines. Code and data are available at `https://tinyurl.com/yae6s36s`.

Outline: Section 5.2 presents baseline approaches. Section 5.3 formally defines the problem. Section 5.4 discusses the computational challenges. Section 5.5 presents our solutions. Section 9.5 shows experimental results. Finally, Section 5.7 discusses the related work and Section 5.8 concludes the paper.

5.2 Three baseline approaches

Consider the undirected network G in Fig. 5.3a. The spread of a disease—a *network process*—in G can be represented as a directed and possibly disconnected graph among the infected nodes. A node can get infected by an external source or a neighboring node. Fig. 5.3b shows 10 network processes in G. We define the final set of infected nodes at the end of a disease spread as its network state. We now discuss 3 approaches to finding network patterns from this set of network states.



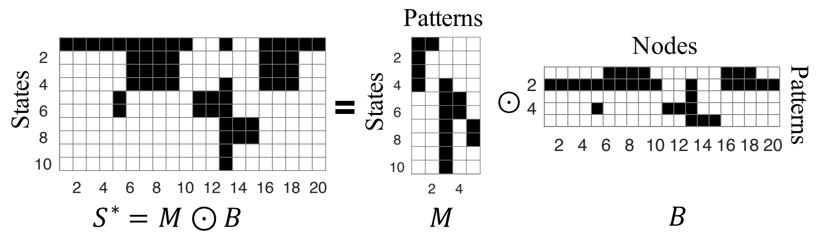
(a) Network *G* (c) Binary state matrix *S*, black = 1, white = 0

(b) Network processes ***P1-P10***, each corresponding to a row of matrix *S* in (c)

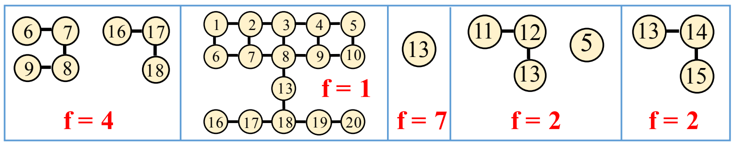
Figure 5.3: Network, network processes, and final network states (final set of participating nodes)

Binary Matrix Factorization (BMF): If we ignore the network structure, the final *network states* of these network processes can be described by a binary *state matrix* S (which states contain which nodes, Fig. 5.3c). Using BMF techniques such as ASSO [1], we can summarize S by a binary *pattern matrix* B (which nodes belongs to which patterns), and a binary *mapping matrix* M (which network states contain which patterns) (Fig. 5.4a). Denote $S^* = M \odot B$ as the recovered state matrix, where $\odot$ is binary matrix multiplication. The summarization error is $||S \oplus S^*||_F^2$, where $\oplus$ is logical XOR and the squared Frobenius norm $||.||_F^2$ counts the number of different bits between S and S^*. For the example in Fig. 5.4, the error of ASSO is 10. While ASSO succeeds in summarizing S (S^* in Fig. 5.4a is similar to S in Fig. 5.3c), ignoring the network leads to infrequent or disconnected patterns. Fig. 5.4b shows the induced subgraphs in G of the obtained patterns: the first and fourth patterns are *disconnected subgraphs*; the third one is a *single node with high frequency* ($f = 7$); the second one is a *huge subgraph with low frequency* ($f = 1$, only matches $P1$ in Fig. 5.3b) and is unlikely to fit unseen data. Thus, ASSO (and BMF in general) satisfies C1, C5, and C6, but not C2, C3, and C4.

Frequent subgraph mining: To capture network effect, we can find frequent subgraph patterns in the network processes using gSpan [2], a popular tool for frequent subgraph mining. Fig. 5.5a shows the top-5 frequent *connected and undirected subgraphs* from Fig. 5.3c.



(a) Top-5 binary patterns found by Binary Matrix Factorization (ASSO): B is the pattern matrix (patterns x nodes), M is the mapping matrix (states x patterns). If S is summarized using M and B, we can recover S as $S^* = M \odot B$, with summarization error $\|S \oplus S^*\|_F^2 = 10$.

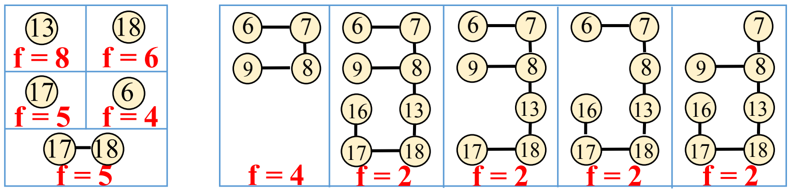


(b) Induced subgraphs in G of top-5 patterns in (a). Each pattern corresponds to a row in B. Their frequencies f correspond to the number of 1-bits in the corresponding columns of M.

Figure 5.4: Finding top-5 binary patterns in S with ASSO [1]

Unfortunately, the top-5 frequent subgraphs only have one or two nodes (the most active nodes) and do not capture any network effect. An obvious improvement is to rank the subgraphs by their coverage, i.e., the products of their frequencies and sizes, as shown in Fig. 5.5b. While the obtained subgraphs are bigger, they *overlap significantly* and thus are not powerful in summarizing networked behaviors. Besides, gSpan uses exact matching (a pattern must lie completely within a network process). Consequently, both of these approaches lead to remarkably higher summarization errors compared to ASSO (39 and 38 compared to 10). In summary, gSpan achieves C2, C3, C4, and C6, but not C1 and C5.

Network-constrained BMF: This paper introduces netBMF, which incorporates all characteristics C1-C6 to yield the result in Fig. 5.6 (frequency and size thresholds are 2 and 3) with only a small compromise in summarization error (14) compared to ASSO (10), while obtaining frequent and connected local communities that constitute the network processes. netBMF features an elegant trade-off between BMF (accuracy) and frequent subgraph mining (inter-



(a) Top-5 frequent subgraphs. Error = 39 (b) Top-5 subgraphs with highest coverage. Summarization error = 38

Figure 5.5: Finding top frequent subgraphs using gSpan [2].

pretability). It is generic, scalable and can be applied to the problem of BMF when the dataset is huge and has network constraints.

5.3 Problem Definition

Let us denote a network as $G = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ is the set of nodes, and E is the set of edges. A network process is an infection process over time on a network G. Each node can get infected from an external source outside G or a neighboring infected node in G. The *network state* $s \subseteq V$ of a network process is its *final set of infected nodes*.

Definition 5.3.1 (State matrix) *The state matrix of a set of network states* $\mathcal{S} = \{s_1, \ldots, s_m\}$ *is* $S \in \{0,1\}^{m \times n}$*, where* $S_{i,j} = 1$ *iff* $v_j \in s_i$*, i.e., node* v_j *is infected in the network process corresponding to* s_i*.*

Fig. 5.3c shows the state matrix for the processes in Fig. 5.3b.

Problem 5.3.1 (netBMF) *Given a network* $G = (V, E)$*, where* $V = \{v_1, \ldots, v_n\}$*, a state matrix* $S \in \{0,1\}^{m \times n}$*, the number of patterns* k*, a frequency threshold* $minf$*, and a size threshold* $minSize$*, network-constrained Binary Matrix Factorization (netBMF) finds a basis (pattern) matrix* $B \in \{0,1\}^{k \times n}$*, and a mapping matrix* $M \in \{0,1\}^{m \times k}$ *s.t.:*

1. *Summarization error* $g(S, B, M) = ||S \oplus (M \odot B)||_F^2$ *is minimized, where* $\oplus$ *is logical XOR,* $\odot$ *is binary matrix multiplication, and* $||.||_F$ *is the Frobenius norm.* g *counts the number of*

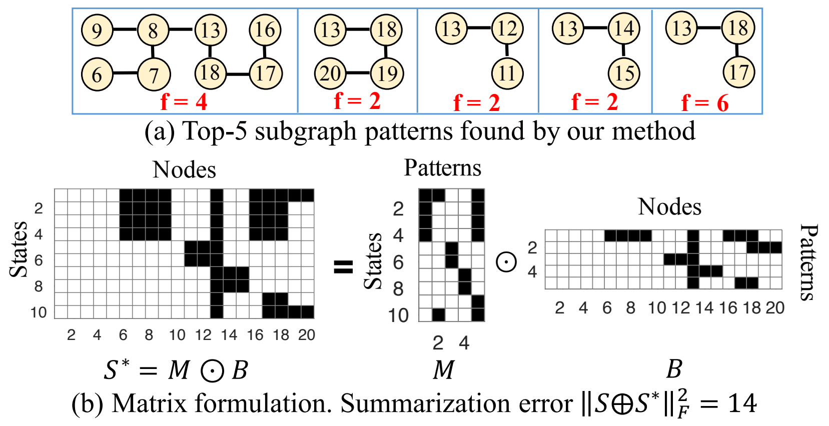

f = 4
f = 2
f = 2
f = 2
f = 6
(a) Top-5 subgraph patterns found by our method
Nodes
States
Patterns
$S^* = M \odot B$
M
B
(b) Matrix formulation. Summarization error $\|S \oplus S^*\|_F^2 = 14$

Figure 5.6: Top-5 patterns for the example in Fig. 5.3 by our method netBMF with minimum frequency 2 and minimum size 3.

different bits between S and $M \odot B$.

2. *The induced undirected subgraph in G of each row of B is connected, i.e., $G_q = (V_q, E_q)$ is connected $\forall q = 1, \dots, k$, where $V_q = \{v_j \in V | B_{q,j} = 1\}$, $E_q = \{(v_j, v_{j'}) \in E | v_j, v_{j'} \in V_q\}$.*
3. *Frequency: $f(q) = \sum_{i=1}^{m} M_{i,q} \geq minf$, $\forall q = 1, \dots, k$.*
4. *Size: $|V_q| = \sum_{j=1}^{n} B_{q,j} \geq minSize$, $\forall q = 1, \dots, k$.*

Denote $\mathcal{B} = \{G_q | q = 1, \dots, k\}$ as the net-basis corresponding to B. Each G_q is called a basis subgraph (a network pattern). We use $\mathcal{B}$ and B interchangeably in the rest of the paper.

The four conditions in Problem 5.3.1 aim at the six characteristics of network patterns discussed in Section 5.1. Specifically, condition (1) helps us satisfy characteristics C1 and C5 since both negative and positive errors are allowed in the mapping of basis subgraphs to network states, i.e., both 0s and 1s in S can be flipped in $M \odot B$. Condition (2) ensures that C2 is met. Furthermore, since the connectivity of a pattern is checked on the original network G instead of the directed graphs of the network processes, characteristic C6 is also met. Finally, conditions (3) and (4) guarantee C3 and C4 respectively.

Fig. 5.6 shows the result of netBMF for the example in Fig. 5.3 ($minf = 2$ and $minSize =$

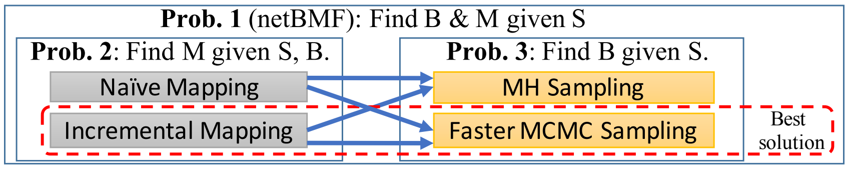

Prob. 1 (netBMF): Find B & M given S
Prob. 2: Find M given S, B.
Prob. 3: Find B given S.
Naïve Mapping
Incremental Mapping
MH Sampling
Faster MCMC Sampling
Best solution

Figure 5.7: Overview of four solutions for netBMF.

3). Each row of B corresponds to one connected subgraph in G, while each column of M indicates which network states (approximately) contain that subgraph. The approximated state matrix S^* in Fig. 5.6b closely resembles S in Fig. 5.3c, suggesting low summarization error ($g(S, B, M) = 14$). Moreover, all obtained patterns are connected subgraphs with at least 3 nodes (sum of each row in $B \geq minSize$), and are contained in at least 2 network states (sum of each column in $M \geq minf$). In conclusion, all four conditions in Problem 5.3.1 are satisfied.

We solve Problem 5.3.1 by solving two sub-problems (see Fig. 5.7): Problem 5.3.2 finds M when S and B are fixed, while Problem 5.3.3 finds B, assuming that M can be found for each given B and S using a solution to Problem 5.3.2.

Problem 5.3.2 (Optimal net-basis mapping) *Given a state matrix $S \in \{0,1\}^{m\times n}$ and a basis matrix $B \in \{0,1\}^{k\times n}$, find the optimal mapping matrix $\pi^o_{B,S} \in \{0,1\}^{m\times k}$ such that the summarization error is minimized, i.e.,*

$$\pi^o_{B,S} = \underset{M\in\{0,1\}^{m\times k} s.t. \sum_{i=1}^{m} M_{i,q}\geq minf, \forall q}{\arg\min} g(S, B, M) \tag{5.1}$$

Problem 5.3.3 (Optimal net-basis) *Given a state matrix $S \in \{0,1\}^{m\times n}$, find a valid basis matrix B or a net-basis $\mathcal{B}$ as in Problem 5.3.1, to minimize the summarization error, i.e.,*

$$B = \underset{B'\in\{0,1\}^{k\times n} \text{ s.t. } B' \text{ is a valid basis matrix}}{\arg\min} g(S, B', \pi^o_{B',S}) \tag{5.2}$$

Theorem 5.3.1 *Problem 5.3.1 is equivalent to Problem 5.3.3.*

Intuitively, given a net-basis $\mathcal{B}$, i.e., a set of network patterns, Problem 5.3.2 decides which network processes contain which network patterns to minimize the summarization error. Since Problem 5.3.3 is equivalent to Problem 5.3.1, it directly outlines a solution to netBMF, which we will use later in Section 5.5.

5.4 Properties of netBMF problem

5.4.1 NP-hardness

Theorem 5.4.1 *Problem 5.3.1 (thus Problem 5.3.3) is NP-hard.*

Proof: BMF (Problem 5.3.1 without conditions 2-4) is NP-hard [1]. Given any instance of the BMF problem, we can always construct an equivalent netBMF problem in polynomial time, where G is a full graph with nodes corresponding to columns of S, $minf = minSize = 1$. ■

5.4.2 Non-submodularity

Theorem 5.4.2 *Problem 5.3.3 (thus Problem 5.3.1) is not submodular.*

Proof: Denote $h(\mathcal{B}) = -g(S, B, \pi^o_{B,S})$. Since Problem 5.3.3 minimizes g, it maximizes h. We prove that h (thus Problem 5.3.3) is not submodular by a counter example. Consider a full graph with $V = \{1,2,3,4,5,6\}$, $\mathcal{S} = \{s\}$, where $s = \{1,2,3,4\}$, and four patterns $G_1 - G_4$, whose node sets are $V_1 = \{1,2,5\}$, $V_2 = \{2,3\}$, $V_3 = \{1,6\}$, $V_4 = \{2,4,6\}$. Hence, $S = [1,1,1,1,0,0]$. Choose $\mathcal{A} = \{G_1\} \subseteq \mathcal{B} = \{G_1, G_2, G_3\}$. It is easy to show that $\pi^o_{\mathcal{A},S} = [1,0,0]$, $\pi^o_{\mathcal{A}\cup\{G_4\},S} = [1,0,0,0]$, $\pi^o_{\mathcal{B},S} = [0,1,1]$ or $[1,1,0]$, and $\pi^o_{\mathcal{B}\cup\{G_4\},S} = [0,1,1,1]$. Thus, $h(\mathcal{A} \cup \{G_4\}) = h(\mathcal{A}) = -3$, $h(\mathcal{B}) = -2$, and $h(\mathcal{B} \cup \{G_4\}) = -1$. Therefore, $h(\mathcal{A} \cup \{G_4\}) - h(\mathcal{A}) < h(\mathcal{B} \cup \{G_4\}) - h(\mathcal{B})$ even though $\mathcal{A} \subseteq \mathcal{B}$. Thus, h is not submodular. ■

5.4.3 Computational challenges

Since Problem 5.3.3 is NP-hard but not submodular, we cannot bound the quality of a greedy algorithm as proved in [54]. Instead, we need to perform the following costly steps: (1) enumerate all possible connected subgraphs of G, (2) enumerate all possible sets of k subgraphs, and (3) solve Problem 5.3.2 for each set of k subgraphs. Because the number of subgraphs of a network grows exponentially with its size, both step 1 and step 2 are infeasible. Moreover, given a net-basis $\mathcal{B}$ of size k, to find the optimal mapping in step 3, we need to check all 2^k possible subsets of $\mathcal{B}$ for each network state, leading to a total cost of $O(2^k nm)$ time, where n is the number of nodes, and m is the number of network states. Clearly, this exponential cost is also not scalable for large k. Thus, netBMF presents us with three computational challenges *D1-D3*:

- ***Challenge D1****: How to find the best set of k subgraphs without checking all possible sets of k subgraphs?*
- ***Challenge D2****: Can we approximate the optimal net-basis mapping with an algorithm linear in k?*
- ***Challenge D3****. Can we find good candidate subgraphs without enumerating the entire subgraph search space?*

5.5 Solution to netBMF problem

We will address the challenges in Section 5.4.3 to devise a scalable solution for Problem 5.3.3. Section 6.5 targets ***challenge D1*** with a general greedy algorithm. Section 5.5.2 copes with ***challenge D2*** using two linear variants of Problem 5.3.2. Finally, Section 5.5.3 deals with ***challenge D3*** by exploring the exponential subgraph search space, which cannot be computed or stored in its entirety, with two sampling algorithms.

5.5.1 Greedy Algorithm

To tackle ***challenge D1***, we avoid checking all possible sets of k subgraphs by sequentially adding the basis subgraph with the highest marginal gain to $\mathcal{B}$, as shown in Algorithm 6. The marginal gain $\Delta(\mathcal{B}, b)$ is defined as follows.

Definition 5.5.1 (Marginal Gain) *Given a state matrix S and a net-basis $\mathcal{B}$, the marginal gain $\Delta(\mathcal{B}, b) \geq 0$ of a new basis subgraph b is the reduction in summarization error when b is added to $\mathcal{B}$, i.e.:*

$$\Delta(\mathcal{B}, b) = g(S, \mathcal{B} \cup \{b\}, \pi^o_{\mathcal{B}\cup\{b\},S}) - g(S, \mathcal{B}, \pi^o_{\mathcal{B},S}) \tag{5.3}$$

Algorithm 2 Greedy_Algorithm$(G, \mathcal{S}, k, minf, minSize)$

```
1: $\mathcal{B} := \emptyset$
2: **while** $|\mathcal{B}| < k$ **do**
3:     $best := \arg\max_b \{\Delta(\mathcal{B}, b) | |b| \geq minSize, f(b) \geq minf, b$ is a connected subgraph of $G\}$
4:     $\mathcal{B} := \mathcal{B} \cup \{best\}$
5: **end while**
6: **return** $\mathcal{B}$
```

Unfortunately, Algorithm 6 is still not scalable. The bottleneck lies in Line 3, where we perform two costly tasks corresponding to ***challenges D2*** and ***D3***: (i) iterating over all connected subgraphs b of G, and (ii) mapping all network states in $\mathcal{S}$ into the net-basis $\mathcal{B} \cup \{b\}$ to compute the marginal gain. For the former task, we propose to use sampling in Section 5.5.3. For the latter task, we propose two linear variants (w.r.t. k) of Problem 5.3.2 in Section 5.5.2.

5.5.2 Linear net-basis mapping

We cope with ***challenge D2*** by proposing two variants of Problem 5.3.2 that can be solved in linear time w.r.t. k: naïve mapping only allows exact matching, while incremental mapping allows missing nodes. Both variants incrementally update M (add one more column) when a new basis subgraph (one more row) is added to $\mathcal{B}$.

Naïve net-basis mapping

The first variant maps a basis subgraph b to a network state s only if s contains all nodes of b, or more formally:

Problem 5.5.1 (Naïve net-basis mapping) *Given a basis matrix $B \in \{0,1\}^{k\times n}$ and a state matrix $S \in \{0,1\}^{m\times n}$, a naïve mapping matrix is defined as $\pi^{naive}_{B,S} = M \in \{0,1\}^{m\times k}$, such that $M_{i,q} = 1$ iff $S_{i,j} = 1 \, \forall j$ s.t. $B_{q,j} = 1$, and $M_{i,q} = 0$ otherwise, i.e., $V_q \subseteq s_i$.*

Theorem 5.5.1 *netBMF problem with naïve net-basis mapping, namely netBMFn, is NP-hard but monotonic and submodular.*

Proof: The NP-hard proof is similar to that for netBMF. Monotonicity and submodularity come from the fact that given two net-basis $\mathcal{A} \subseteq \mathcal{B}$, if a basis subgraph $b \in \mathcal{A}$ is mapped to some network state s, then $b \in \mathcal{B}$ and will also be mapped. ■

Incremental net-basis mapping

The naïve mapping does not allow both positive and negative errors in mapping, reducing the frequencies of basis subgraphs. We next introduce the second variant that incrementally maps basis subgraphs in the row order of B to a network state s as long as this mapping reduces summarization error, leading to a linear cost. Fig. 5.8 provides a visual explanation of incremental mapping.

Problem 5.5.2 (Incremental net-basis mapping) *Given a basis matrix $B \in \{0,1\}^{k\times n}$ and a state matrix $S \in \{0,1\}^{m\times n}$, an incremental mapping matrix is defined as $\pi^{incre}_{B,S} = M^{(k)} \in \{0,1\}^{m\times k}$, where $M^{(0)}$ is empty and $M^{(q)} \in \{0,1\}^{m\times q}$ is the mapping matrix for the first q rows of $B \, \forall q \leq k$. Here, $M^{(q)}$ is computed incrementally by adding one new column to $M^{(q-1)}$ such that:*

$$M^{(q)} = \underset{M\in\{0,1\}^{m\times q} s.t. M_{i,j}=M^{(q-1)}_{i,j}, \forall i, \forall j<q}{\arg\min} g(S,B,M), \forall q \leq k$$

Theorem 5.5.2 *netBMF with incremental net-basis mapping, namely netBMFi problem, is NP-hard but not submodular.*

Quality Guarantee: Due to Theorems 5.4.2 and 5.5.2, we do not have any guarantee on the quality of Algorithm 6 for the optimal netBMF and netBMFi. However, from Theorem 5.5.1, Algorithm 6 guarantees an approximation of $\left(1 - \frac{1}{e}\right)$ or better for netBMFn. In particular:

$$g(S, B_{greedy}, \pi^{naive}_{S,B_{greedy}}) \geq (1 - \frac{1}{e})g(S, B^*, \pi^{naive}_{S,B^*}) \tag{5.4}$$

where B^* is the optimal solution of netBMFn, and B_{greedy} is the greedy solution by Algorithm 6 that iteratively adds the basis set with the maximum marginal gain [54]. Moreover, no other polynomial time algorithm can achieve an approximation guarantee better than $1 - \frac{1}{e}$ unless $P = NP$, as proved in [55].

Running time: The running time of both net-basis mapping variants is $O(mnk)$, where m, n, k are the number of states, nodes, and net-bases respectively.

Algorithm 3 netBMF($G, \mathcal{S}, k, minf, minSize, numSeeds, maxIter$)

```
1: $\mathcal{B} \leftarrow \emptyset$
2: while $|\mathcal{B}| < k$ do
3:     $best := \emptyset$
4:     $seeds$ :=Get_Seed_Nodes($G, S, \mathcal{B}, numSeeds$)
5:     for $seed \in seeds$ do
6:         $V_q$ := Sample_Basis($G, S, \mathcal{B}, minf, minSize, seed, maxIter$)
7:         $V_q^*$ := Improve_Basis($G, S, \mathcal{B}, minf, minSize, V_q$)
8:         if $\Delta(\mathcal{B}, V_q^*) > \Delta(\mathcal{B}, best)$ then
9:             $best := V_q^*$
10:        end if
11:    end for
12:    $\mathcal{B} \leftarrow \mathcal{B} \cup \{best\}$
13: end while
14: return $\mathcal{B}$
```

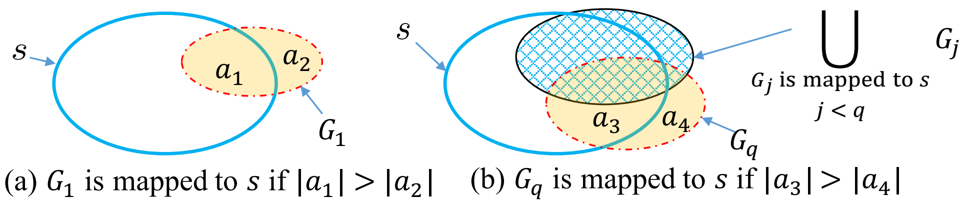

s
a_1
a_2
G_1
(a) G_1 is mapped to s if $|a_1| > |a_2|$
s
$\bigcup G_j$
G_j is mapped to s
$j < q$
a_3
a_4
G_q
(b) G_q is mapped to s if $|a_3| > |a_4|$

Figure 5.8: Incrementally map $\mathcal{B} = \{G_1, \ldots, G_k\}$ to network state s.

5.5.3 Incremental sampling of net-basis

To deal with the exponential subgraph space in ***challenge D3***, we sample this space to find subgraphs with high marginal gains. We first introduce the high-level sampling-based netBMF algorithm in Section 5.5.3. After that, we propose two sampling methods used by this high-level algorithm: a Metropolis-Hasting (MH) algorithm in Section 5.5.3, and a faster Monte Carlo Markov Chain (MCMC) algorithm in Section 5.5.3. Finally, Section 5.5.3 describes how to choose good seed nodes from which to start the sampling.

Sampling-based netBMF algorithm

Algorithm 3 is the high-level sampling-based netBMF algorithm which incorporates sampling into Algorithm 6. We start with an empty net-basis $\mathcal{B}$ in Line 1, and then greedily add the basis subgraphs with the highest marginal gains to $\mathcal{B}$ until we obtain k basis subgraphs in Lines 2-10. In each iteration, we first get a list of seed nodes for sampling in Line 4 (detailed in Section 5.5.3), then perform sampling from these seed nodes in the subgraph search space to get a good candidate subgraph in Line 6, and further improve this candidate in Line 7 (Algorithms 4 and 5 in Section 5.5.3). The best sampled subgraph is recorded in Lines 8-9, and later added to $\mathcal{B}$ in Line 10.

Sampling goal: In Line 6 of Algorithm 3, we need the undirected connected subgraphs of G with the highest marginal gain (Equation 5.3). However, we cannot guarantee that this exact subgraph will be sampled due to the random nature of sampling. Alternatively, we can sample subgraphs with probability proportional to their marginal gains, and visit the best one while doing so. Denoting τ_b as the probability of visiting a subgraph b, we need $\tau_b \propto \Delta(\mathcal{B}, b)$, where $\mathcal{B}$ is the current net-basis in an iteration of Algorithm 3. We can achieve τ with MH algorithm in Section 5.5.3.

Sampling space—Edit Graph (EG). Before discussing the detailed sampling methods, we formally design a way to navigate the subgraph search space using the concept of an edit graph, which represents all possible edits that can be performed on any basis subgraph b. We denote $b \rightarrow u$ and $b \leftarrow u$ as the new connected subgraphs obtained by removing and adding node u to a connected subgraph b respectively.

Definition 5.5.2 (Edit Graph) *The edit graph of a network $G = (V, E)$ is a directed edge-weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$, where $\mathcal{V} = \{b | b \text{ is an undirected connected subgraph of } G\} \cup \{\emptyset\}$, $\mathcal{E} = \{(b, b') | b' = b \leftarrow u \text{ or } b' = b \rightarrow u, u \in V, b \in \mathcal{V}, b' \in \mathcal{V}\}$, and $\mathcal{F} : \mathcal{E} \rightarrow \mathbb{R}$ is a function that assigns weights to edges in $\mathcal{E}$.*

Fig. 5.9 shows an example EG. The EG is simply a graph whose vertices are undirected connected subgraphs of G or the empty subgraph. For clarity, we use the term "node" for the network G, and "vertex" for the EG. An edge between two vertices in the EG indicates that we can edit the corresponding subgraphs into each other by adding or deleting one node. The edge weights reflect the potential impact of the edits on the marginal gain and are explained in Section 5.5.3. Since G is finite, the EG is also a *finite* space. Its size is *exponential* w.r.t. the size of G, making it infeasible to be computed or stored in its entirety. More importantly, the EG is a *connected graph*, thanks to the inclusion of the empty subgraph $\emptyset$. As a result, we can start at a random basis subgraph and, after a finite series of *local edits*, reach any other desired basis subgraph. We next exploit this property to perform sampling in the EG.

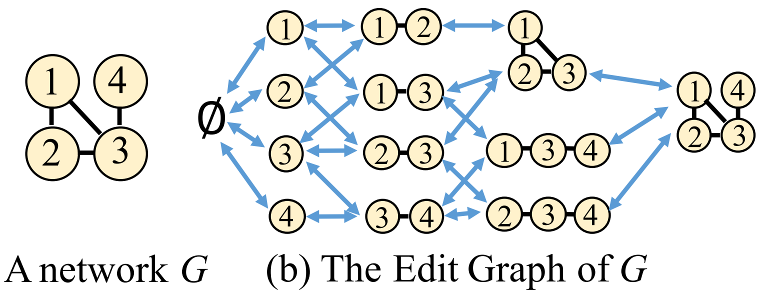


Figure 5.9: (a) An example network G and (b) its Edit Graph $\mathcal{G}$.

MH-sampling on the edit graph

We now utilize the MH algorithm to sample basis subgraphs in the EG with probability proportional to their marginal gains.

Metropolis-Hastings (MH) sampling: Given a state space $\Omega = \{a_1, a_2, ...\}$, let $w_i \geq 0$ be the value of state a_i. The MH algorithm samples state a_i with probability proportional to w_i, i.e., the *target visit probability distribution* is τ, where:

$$\tau_i = \frac{w_i}{\sum_{a_i \in \Omega} w_i} \tag{5.5}$$

The denominator in the above equation is infeasible to compute when Ω is large. MH algorithm simulates τ by converting the state space into a Markov chain with an arbitrary transition matrix Q called the *proposal distribution matrix*. If the current state X_t at time step t is a_i, MH algorithm finds the next state X_{t+1} as follows:

- Draw a random state a_j with probability Q_{ij}.
- Compute the *acceptance probability* α_{ij}:

$$\alpha_{ij} = min\left\{1, \frac{\tau_j Q_{ji}}{\tau_i Q_{ij}}\right\} = min\left\{1, \frac{w_j Q_{ji}}{w_i Q_{ij}}\right\} \tag{5.6}$$

- $X_{t+1} = \begin{cases} a_j & \text{with probability } \alpha_{ij} \\ a_i & \text{with probability } 1 - \alpha_{ij} \end{cases}$

The Markov chain as described above is *reversible*, *ergodic*, and satisfies the detailed balance property. Thus, the stationary distribution of this Markov chain has been proved to be unique and to converge to the target distribution τ in Eqn. 5.5.

To apply MH algorithm to sample the EG, we next specify the state values w_i and the proposal distribution matrix Q.

State space: Each vertex (basis subgraph) in the EG is a state in Ω. Since we are only interested in basis subgraphs with high marginal gain, particularly the best one, we set the target distribution to be approximately proportional to the marginal gains (Equation 5.3). Therefore, the state values w_b of basis subgraph b is computed as:

$$w_b = \Delta(\mathcal{B}, b) + 1 \tag{5.7}$$

Here, the addition of 1 is to avoid division by zero in Eqn. 5.6.

Proposal distribution matrix: The proposal distribution matrix Q contains the edge weights in the EG, which reflects the "quality" of the edits on any given basis subgraph b. An edit is "good" if it increases our chance of finding basis subgraphs with high marginal gains. Since it is costly to compute the exact marginal gains for all neighbors of a basis subgraph b in the EG, we approximate the quality of edits by a *potential vector* **y**.

Given the current net-basis $\mathcal{B}$, basis matrix B, mapping matrix M, and the current candidate basis subgraph b to be added to $\mathcal{B}$, we define $\hat{S}$ as the part of the state matrix S that has not been covered by any basis subgraphs in $\mathcal{B}$, i.e.,

$$\hat{S}_{i,j} = \begin{cases} 0 & \text{if} \quad (M \odot B)_{i,j} = 1 \\ S_{i,j} & \text{otherwise} \end{cases} \tag{5.8}$$

In addition, denote $\mathbf{x} \in \{0,1\}^{m\times 1}$ as the mapping of b to S, i.e., $\mathbf{x}$ is the last column of $\pi_{\mathcal{B}\cup\{b\},S}$: $x_i = 1$ iff state s_i is mapped to b. Finally, we define the *potential vector* $\mathbf{y} \in \{0,1\}^{1\times n}$ as follows:

$$\mathbf{y} = \mathbf{x}^T \times \hat{S} \tag{5.9}$$

Intuitively, y_j counts how many times a node v_j in G is contained in the network states that b is mapped to, without counting the portion of S that has been covered by the current net-basis $\mathcal{B}$. If y_j is high, v_j belongs to the same network states with b many times. In this case, adding v_j to b potentially increases the marginal gain, while removing v_j from b (if $v_j \in b$) potentially decreases the marginal gain. Denote A and D as the sets of connected basis subgraphs that can be obtained by adding and removing one node from b respectively:

$$A = \{b' = b \leftarrow v_j | v_j \in V\} \tag{5.10}$$

$$D = \{b' = b \rightarrow v_j | v_j \in V\} \tag{5.11}$$

We now define the proposal distribution matrix Q as follows:

$$Q_{bb'} = \begin{cases} \beta \frac{1}{C_A}(y_j + \epsilon) & \text{if} \quad b' = b \leftarrow v_j \in A \\ (1-\beta)\frac{1}{C_D}\frac{1}{y_j+\epsilon} & \text{if} \quad b' = b \rightarrow v_j \in D \end{cases} \tag{5.12}$$

where $\beta \in [0,1]$ is the probability of addition, $0 < \epsilon \ll 1$ is a small constant to avoid division by zero, C_A and C_D are two normalization constants defined as $C_A = \sum_{b'=b\leftarrow v_j \in A}(y_j + \epsilon)$ and $C_D = \sum_{b'=b\rightarrow v_j \in D}\frac{1}{y_j+\epsilon}$. Based on Eqn. 5.12, we perform an addition with probability proportional to the potential of the added node, while a deletion with probability inversely proportional to the potential of the removed node. The role of β is to account for the fact that the number of supergraphs of a basis subgraph b (which is equal to the number of neighbors in G of the nodes in b) is likely significantly larger than the number of its subgraphs (which is at most its size $|b|$). Thus, without β, it is unlikely for a deletion to be chosen in the sampling process. Note that the sampler would also explore some "bad" edits to avoid being stuck in local optimums.

Final MH algorithm in the EG: The MH algorithm for sampling the next basis subgraph with high marginal gain is shown in Algorithm 4. Here, the sampler starts at a given seed node, performs sampling in this node's neighborhood for at most $maxIter$ steps, and returns the visited basis subgraph with the highest marginal gain. Due to the randomness of sampling and the limitation of $maxIter$, it is likely that the basis subgraph returned by Algorithm 4 can still be further improved locally. Hence, we propose to greedily perform the best possible local edits on this basis set until no more good edits can be found, as shown in Algorithm 5.

Faster MCMC algorithm

Since we only care about a single basis subgraph with the highest marginal gain, there is no need to sample subgraphs based on the exact target distribution in Eqn. 5.5. Therefore, to reduce running time, we adopt a new acceptance probability instead of Eqn. 5.6:

$$\alpha_{ij} = min\left\{1, w_j/w_i\right\} \tag{5.13}$$

Here, we always accept a good move that increases marginal gain. Otherwise, the acceptance probability is equal to the ratio of the marginal gains of two basis subgraphs. As a result, we avoid computing Q_{bX_t} in Line 10, saving considerable computational cost, and replace Line 11 with Eqn. 5.13 in Algorithm 4. More importantly, the modified MCMC algorithm is still *finite, reversible, ergodic, and thus converges to a unique stationary distribution* even though we do not have a closed form of this distribution as in Eqn. 5.5 for MH.

Algorithm 4 Sample_Basis$(G, \mathcal{S}, \mathcal{B}, minf, minSize, seed, maxIter)$

```
1: $t := 0$
2: $X_t := \{seed\}$
3: Compute $\pi_{\mathcal{B}\cup\{X_t\}}(\mathcal{S})$ and $\Delta(\mathcal{B}, X_t)$
4: $best := X_t$
5: while $t < maxIter$ do
6:     $A := \{X_t \leftarrow u | u \notin X_t, \exists u' \in X_t, (u, u') \in E\}$
7:     $D := \{X_t \rightarrow u | u \in X_t, u$ is not a cut-vertex$\}$
8:     Compute $Q_{X_t b}, \forall\, b \in A \cup D$ based on Eqn. 5.12
9:     Choose a neighbor $b$ from proposal distribution $Q_{X_t b}$
10:    Compute $w_{X_t}$ and $w_b$ based on Eqn. 5.7 and $Q_{bX_t}$ based on Eqn. 5.12
11:    $\alpha := \frac{w_b Q_{bX_t}}{w_{X_t} Q_{X_t b}}$
12:    if $uniform(0, 1) \leq \alpha$ then
13:        $t := t + 1$
14:        $X_t := b$
15:        if $\Delta(\mathcal{B}, X_t) > \Delta(\mathcal{B}, best)$ & $|b| \geq minSize$ & $f(b) \geq minf$ then
16:            $best := X_t$
17:        end if
18:    end if
19: end while
20: return $best$
```

Choosing seeds for sampling

In practice, we cannot run the MH or MCMC algorithm for an infinite number of iterations to converge to the stationary distribution. As a consequence, it is more practical to start the sampler where we are more likely to find basis sets with high marginal gains. We sample seed nodes with probability proportional to their frequencies in $\hat{S}$ in Section 5.5.3, i.e., the number of nonzeros in the corresponding columns of $\hat{S}$.

Running time

A conservative upper bound for Algorithm 4 (for both MH and MCMC) is $O(mnkI + \hat{d}I^2)$, where I is $maxIter$, and $\hat{d}$ is the maximum node degree in G.

Algorithm 5 Improve_Basis$(G, \mathcal{S}, \mathcal{B}, minf, minSize, b)$

```
1: Compute $A^+$ and $D^+$ for $b$
2: while $A^+ \cup D^+ \neq \emptyset$ do
3:     $b := argmax_{b' \in A^+ \cup D^+, |b| \geq minSize, f(b) \geq minf} \Delta(\mathcal{B}, b')$
4:     Update $A^+$ and $D^+$ for $b$
5: end while
6: return $b$
```

5.6 Experiments

We compare four variants of netBMF algorithm to other baselines in four datasets. In summary, netBMFi with fast MCMC sampling (netBMFi-MC) achieves the best results in terms of summarization errors, quality of net-bases, and running time. NetBMF provides better net-bases than BMF (ASSO and TMF) and better accuracy than frequent subgraph mining (gSpan). NetBMF runs in linear time w.r.t. its parameters, and thus is more scalable than ASSO and gSpan. Finally, MCMC sampling is much faster than MH sampling.

5.6.1 Settings

Synthetic data: We generate a network of 1000 nodes (average degree = 10) using the Barabasi-Albert algorithm [56]. We sample 20 random ground-truth network patterns. For each pattern, we create 5 network states by deleting or adding 5 random nodes, leading to 100 network states in total.

Real data: We use three real world datasets extracted from two social networks with directed follower-followee relationships among users: Twitter.com [57,58] and Behance.net [59]. A network process is the spread of a hashtag via users' tweets on Twitter.com, or a creative

Dataset	SmallBehance	BigBehance	Twitter	Synthetic
#nodes (n)	1K	85K	22.3K	1K
#edges ($\|E\|$)	14K	13.5M	576K	39.2K
#states (m)	500	1326	1015	100
$\|\|S\|\|_F^2$	45.7K	376K	49.3K	2.1K

Table 5.2: Statistics of datasets

project via users' likes Behance.net. In both cases, we have implicit information spreads (unknown source nodes). Table 5.2 shows the statistics of datasets.

Baselines: We compare netBMF with two groups of baselines: (i) top-k frequent subgraph mining with best frequency (gSpan-Freq) or coverage (gSpan-Cover) among the top-1000 frequent subgraphs returned by gSpan [2] as explained in Section 5.1; (ii) Binary Matrix Factorization, including ASSO [1] and TMF [60]. We study four variants of netBMF based on pattern mapping approach (incremental or naïve) and sampling approach (MH or faster MCMC): netBMFi-MH, netBMFi-MC, netBMFn-MH, and netBMFn-MC.

Parameter settings: We set the default parameters as: $\beta = 0.7$, $numSeeds = 10$, $maxIter = 100$, $minSize = minf = 1$ or 2 for error and quality evaluations. For Small-Behance and Twitter, $k = 100$. For BigBehance, $k = 30$. For running time experiments, $n = 1000$ and $k = 10$. To test a parameter, we keep the others fixed. Note that the provided code for the baseline TMF does not handle $k > 30$. The result of each experiment is averaged over 5 runs of netBMF. Experiments were run on an Ubuntu machine with Intel Core i7-5930K CPU, 3.50GHz. Code was written in Python.

Evaluation metrics: We use the relative summarization error to evaluate the *accuracy of the decomposition*:

$$RelativeError(Q, B) = \frac{||S \oplus (Q \odot B)||_F^2}{||S||_F^2} \times 100\% \tag{5.14}$$

To check the *stability of sampling*, we use Jaccard index to compare the similarity of two different groups of patterns. In particular, given two net-bases $\mathcal{B}_1$ and $\mathcal{B}_2$, denote n_{11} as the number of node pairs that are in the same basis subgraphs in both $\mathcal{B}_1$ and $\mathcal{B}_2$, n_{10} as the number of node pairs that are in the same basis subgraphs in $\mathcal{B}_1$ but not in $\mathcal{B}_2$, n_{01} as the number of node pairs that are in the same basis subgraphs in $\mathcal{B}_2$ but not in $\mathcal{B}_1$. Then, the Jaccard similarity coefficient (or Jaccard index) between $\mathcal{B}_1$ and $\mathcal{B}_2$ is defined as:

$$J(\mathcal{B}_1, \mathcal{B}_2) = \tfrac{n_{11}}{n_{01}+n_{10}+n_{11}} \tag{5.15}$$

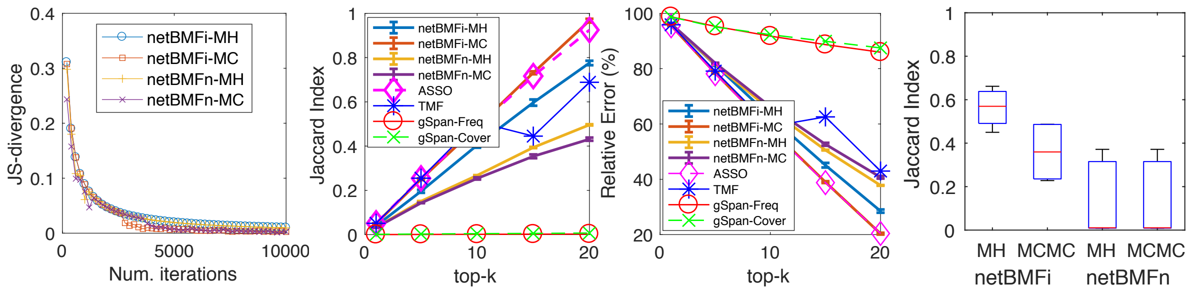
Clearly, $J(\mathcal{B}_1, \mathcal{B}_2) \in [0, 1]$. $J(\mathcal{B}_1, \mathcal{B}_2) = 1$ means $\mathcal{B}_1$ and $\mathcal{B}_2$ are identical, while $J(\mathcal{B}_1, \mathcal{B}_2) = 0$ means they do not overlap.

5.6.2 Convergence and stability of sampling

Convergence of sampling: Theoretically, all of our sampling algorithms converge to some unique stationary distribution (Section 5.5.3), which is confirmed experimentally in Fig. 5.10a. Here, the Jensen-Shannon (JS) divergence compares the visit distribution at every 200 iterations with the previously computed one. All variants of netBMF appear to converge quickly after 2000 iterations.

Recovering ground-truth synthetic patterns: Our algorithms consistently recover the 20 ground-truth patterns in the synthetic dataset at $k = 20$, with comparable or better accuracy than other baselines as reported in Fig. 5.10b. This figure shows the Jaccard index between the obtained net-bases and the ground-truth patterns in the synthetic dataset after 10 runs of netBMF, in comparison with other baselines. While our solutions are based on sampling, they produce similar net-bases between different runs, as confirmed by the small error bars for all netBMF variants (Fig. 5.10b). We observe similar results for the relative errors in Fig. 5.10c. Overall, netBMFi-MC is the best at recovering the ground truth.

Stability of sampled net-bases in real data: Our algorithms produce reasonably stable results despite the huge exponential search space with many different subgraphs having the same marginal gains. Fig. 5.10 shows the distributions of Jaccard index between all pairs of net-bases found in 10 different runs of our algorithms on the SmallBehance dataset. netBMFi allows both positive and negative errors, leading to more stable (and also bigger as shown later)



(a) Convergence (b) Recovering ground-truth patterns (c) Error (d) Small Behance

Figure 5.10: (a-c) **Synthetic dataset** ($minSize = 2$, $minf = 2$): (a) Convergence of sampling; (b) Jaccard index between obtained patterns and the 20 ground-truth patterns; (c) Summarization error. (d) Stability of sampled net-bases

net-bases than those found by netBMFn. netBMFi-MH is the most stable with Jaccard index around 0.6. Both variants of naïve mapping are much less stable, which is understandable because the high level of noise in real world data causes exact patterns to be small while many small patterns have the same marginal gains.

5.6.3 Summarization error

Variants of netBMF: Our results show that netBMFi-MC is the variant with the highest accuracy. All variants provide stable accuracy among five different runs as $minSize$, $minf$ and k vary in Fig. 5.11. Obviously, the errors decrease as k increases. When $minf = minSize = 1$ (no thresholds, Fig. 5.11a, c), the errors of all variants are almost identical. However, as the thresholds increase ($minf = minSize = 2$ in Fig. 5.11b, d), netBMFi and netBMFn diverge. Higher size and frequency thresholds make it harder to find exact patterns, leading to higher errors for netBMFn. Since MH sampling has a higher rate of rejection (Lines 11, 12 in Algorithm 4), it explores a "smaller" subspace of the subgraph search space, causing higher errors than MCMC. Fig. 5.12m, l further confirm that the errors increase as $minf$ and $minSize$ increase.

netBMF v.s. other baselines: Fig. 5.13 shows that our best variant (netBMFi-MC) is a

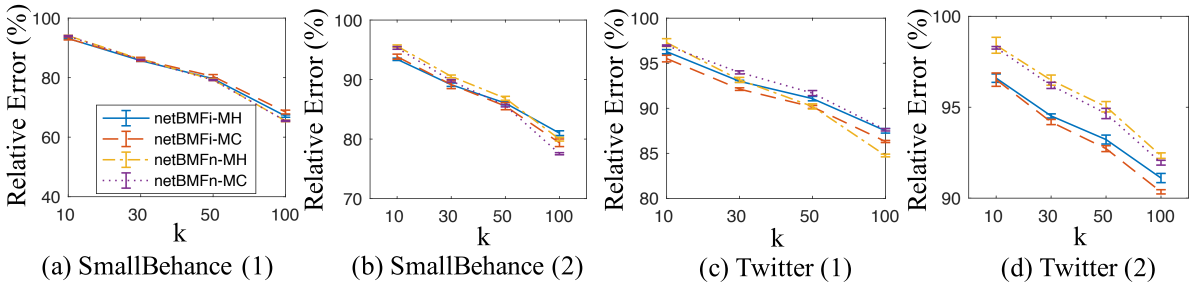


Figure 5.11: Comparing netBMF variants with $minf = minSize = 1$ in (a, c) and $minf = minSize = 2$ in (b, d)

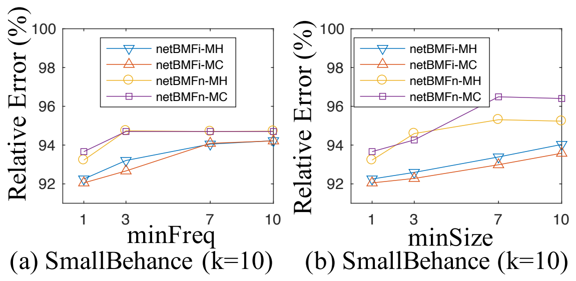
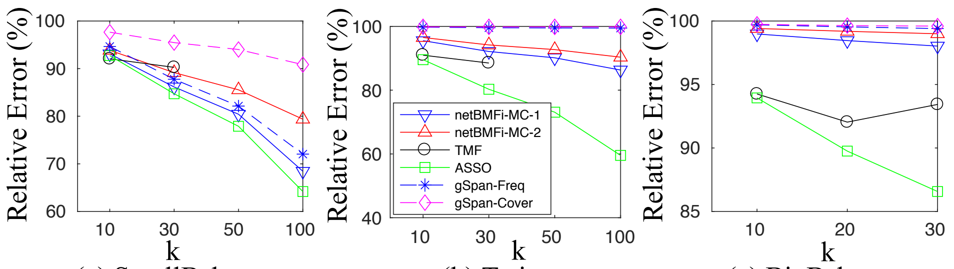


Figure 5.12: Impact of minSize and minfreq

good trade-off between BMF (ASSO, TMF) and frequent subgraph mining (gSpan) in terms of accuracy for various values of k. When size and frequency thresholds are 1, netBMFi-MC-1 produces comparable errors to the best baseline ASSO for SmallBehance. Since ASSO and TMF ignore the network structure, they achieve remarkably smaller errors than netBMF and gSpan, but attain subgraphs with extremely high number of connected components (see Section 5.6.4). Each disconnected pattern returned by ASSO or TMF is effectively a unions of hundreds of connected patterns returned by netBMF or gSpan. As expected, both versions of gSpan perform poorly since they retrieve very small subgraphs of high frequencies. On the contrary, both netBMFi-MC-1 and netBMFi-MC-2 obtain more diverse network patterns, and therefore have lower errors than gSpan.

Clearly, there is a trade-off between accuracy (characteristics C1, C5, C6) and connectedness (C2-C4) of patterns. Bigger patterns that capture longer infection paths in network processes are harder to find, and including such patterns needs a higher noise tolerance. We



(a) SmallBehance (b) Twitter (c) BigBehance

Figure 5.13: Compare netBMF with other baselines: Accuracy as k varies. Here, netBMFi-MC-1 and netBMFi-MC-2 use both size and frequency thresholds of 1 and 2 respectively.

thus continue to analyze the quality of net-bases in Section 5.6.4.

5.6.4 Quality of net-bases

We evaluate the quality of the basis subgraphs (or induced subgraphs for ASSO and TMF) with two types of metrics: (i) size, i.e., the average number of nodes; and (ii) connectedness, including the average number of connected components and clustering coefficients. Overall, netBMF obtains densely connected, large, and highly frequent basis subgraphs.

Fig. 5.14(a-c) report the results averaged over 100 basis subgraphs from different baselines. NetBMF variants outperform all other baselines w.r.t. the above metrics, i.e., the obtained basis subgraphs are big (high number of nodes), connected (only one connected component), and dense (high clustering coefficients). On the contrary, TMF and ASSO find big (high number of nodes in Fig. 5.14a) but disconnected patterns (high number of connected components within each pattern in Fig. 5.14b, and extremely low clustering coefficients in Fig. 5.14c). This is expected since both TMF and ASSO are oblivious to the network structure. On the other hand, gSpan cares more about connectivity of basis subgraphs than their coverage, yielding connected but mostly smaller subgraphs. While not plotted, the average frequencies of patterns returned by all netBMF variants are greater than 6 for Twitter, and 60 for both Behance datasets. As a result, netBMF is superior to all other baselines in summarizing network processes.

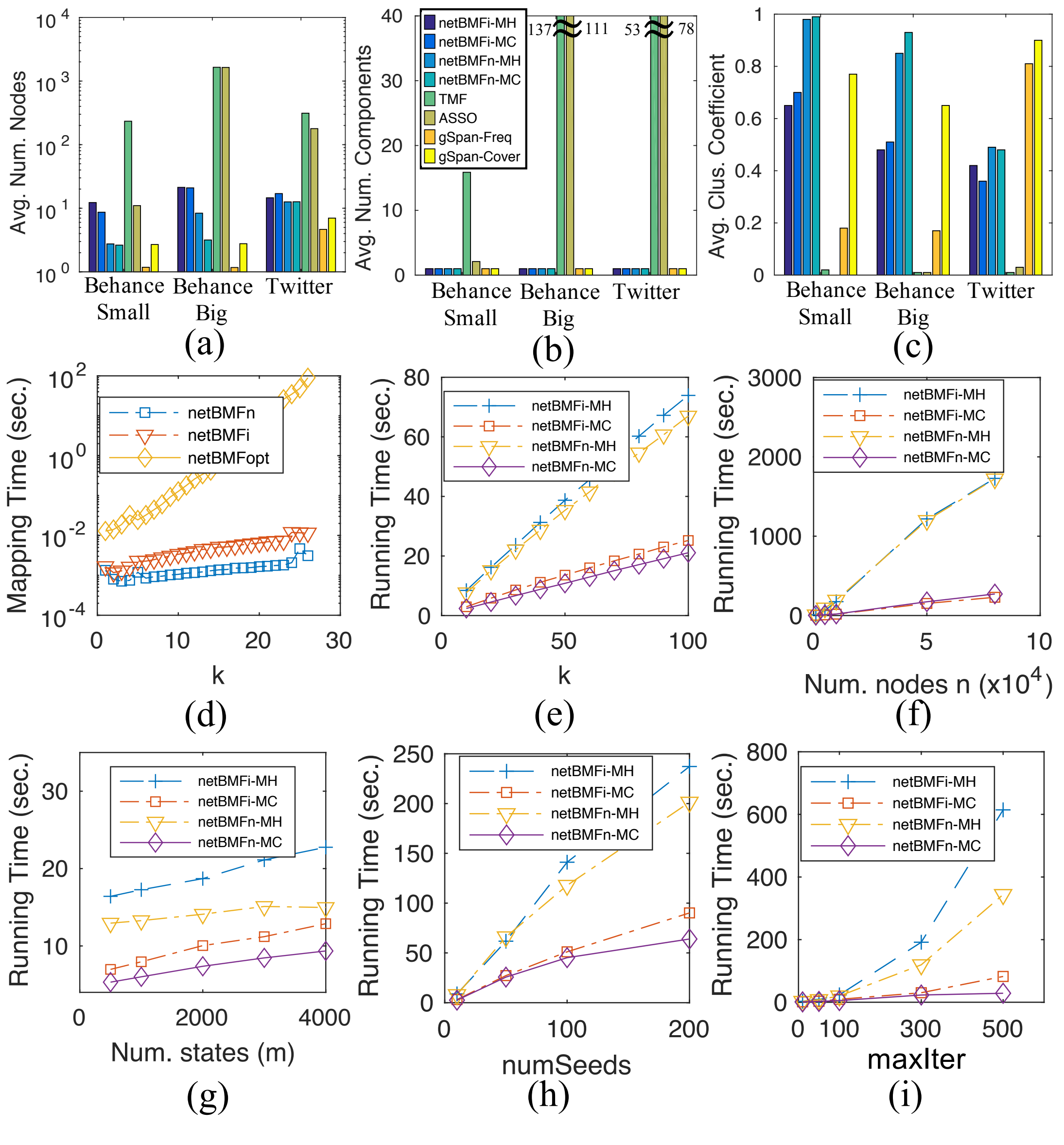


Figure 5.14: (a-c, shared legend in b) Quality of net-bases obtained by different baselines on three real datasets; (d-i) Running time for four variants of the netBMF algorithm as their parameters are varied. For all num. states in (g): $||S||_F^2/m/n = 0.1$.

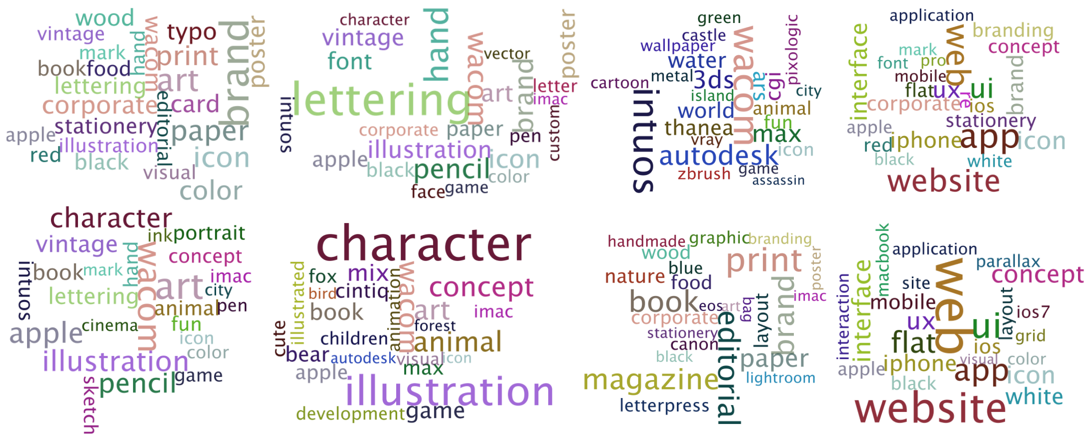


Figure 5.15: Word clouds of the tags on mapped projects for 8 different patterns in Behance.

Among our netBMF variants, the choice of MH or MCMC sampling does not affect the quality significantly, whereas netBMFi generally obtains bigger basis subgraphs than netBMFn, which is natural given that netBMFi allows more error in basis mapping. Note that in Fig. 5.14c, the clustering coefficients of the net-bases obtained by netBMFn are higher than those by netBMFi because netBMFn returns notably smaller subgraphs, as shown in Fig. 5.14a.

Fig. 5.1 and 5.2 show one naive and one approximate patterns from Twitter with their mapped network states (hashtags). The associated hashtags clearly belong to the same topic for each pattern. Fig. 5.15 also shows the 8 word clouds from mapped projects (network states) for 8 patterns in Behance. Clearly, there is a consistent topic in each word cloud, i.e., each pattern corresponds to a tightly connected local community that are interested in the same topic. Note that two different patterns (local communities) can be mapped to the same network states, leading to two similar word clouds such as the two rightmost word clouds in Fig. 5.15. In other words, the same topic may entail various different local communities across the network.

5.6.5 Running time

All four variants of netBMF have linear running time in k, n, m, and $numSeeds$, as shown in Fig. 5.14 (e-h). We further evaluate the speed of three different net-basis mappers on a single process state as the number of patterns k increases in Fig. 5.14d. As discussed in Section 5.4.3, the running time increases linearly in k for netBMFi and netBMFn, but exponentially in k for netBMFopt, making the optimal solution not scalable. Fig. 5.14i shows that the running time is quadratic in $maxIter$ as discussed in Section 5.5.3. A higher number of iterations likely leads to bigger visited subgraphs, which potentially have more neighbors in the EG and cause more computational cost for the transition probability Q in Lines 8 and 10 of Algorithm 4. Finally, MH sampling clearly costs more than MCMC sampling due to the computation of the transition probability $\mathcal{Q}_{X_t b}$ in Line 10 of Algorithm 4. MCMC sampling discards this computation and uses $\alpha = w_b/w_{X_t}$ instead in Line 11, reducing computational cost significantly. To put this into perspective, ASSO took more than 1 day, while netBMFi-MC took 38 minutes and netBMFi-MH took 50 minutes to find 100 patterns in the Twitter dataset. Finally, gSpan was very slow when the required frequency threshold (which is dependent on datasets) is low.

5.7 Related Work

First, our work is related to finding frequent subgraphs from a database of graphs [2,61–64]. However, we would like the frequent subgraphs to be "diverse" so as to minimize the summarization error. Therefore, the classic frequent subgraph mining solutions cannot be applied directly to our netBMF problem. Works on graph motifs [65] also find repeated network patterns but at different places in the same graph, instead of in different graphs as in our problem.

Another similar direction is Boolean Matrix Factorization (BMF) [1, 66–73]. BMF is similar to netBMF without conditions (2-4) in Problem 3. The most popular baseline for BMF, ASSO [1], has quadratic time complexity in m, making it not scalable. gDBMF [68] solves a naïve version of BMF, where $M \odot B$ must not contain any extra nonzeros compared to S. To deal with noise and unknown values, Minimum Description Length principles has been used, such as MDL4BMF [72], Nassau [73], PANDA [74], and PANDA+ [75]. Ternary matrix factorization (TMF [60]) also considers unknown values and runs in linear time w.r.t. m. TMF has been shown to outperform both ASSO and PANDA+, and is thus chosen as a baseline in our paper. FastStep [76] is a scalable BMF, which relaxes M and B to be real-valued matrices. More recently, [77] performs BMF via message passing to achieve smaller error than ASSO.

S and G can be incorporated together in coupled matrix factorization for real-valued data [78, 79]. For example, denoting D as the adjacency matrix of G, we can factorize matrices $S \in \mathbb{B}^{m\times n}$ and $D \in \mathbb{B}^{n\times n}$ together, i.e., $S \approx M \times B$ and $D \approx C \times B$, where $M \in \mathbb{R}^{m\times k}$, $B \in \mathbb{R}^{k\times n}$, and $C \in \mathbb{R}^{n\times k}$ is shared. However, this factorization is different from our problem, where the factor matrices must be binary.

Finally, netBMF is also related to finding frequent patterns based on their shapes or the most influential users in information cascades [80–83] and modelling information cascades (e.g., [84–86]). However, we focus on finding patterns on real data to reduce the summarization errors instead of building a theoretical cascade model.

5.8 Conclusion

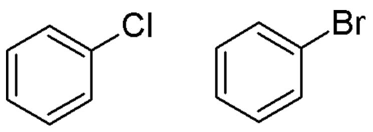
In this paper, we proposed a novel problem of summarizing network processes by mining coherent subsets of connected users who often engage in similar network processes. To cope with the NP-hardness of this problem, we designed a greedy algorithm and four scalable variants that incrementally find the best network patterns by sampling the exponential subgraph space. Extensive experiments demonstrate our solution to be efficient in mining relevant and accurate patterns of information spreads in both synthetic and real datasets. Possible future research includes investigating the temporal aspect of network processes, using directed subgraphs as network patterns, and predicting the evolution of network processes based on the obtained patterns.

Chapter 6

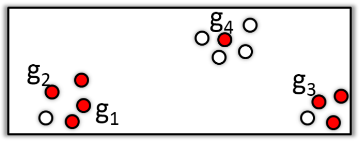
Answering top-k representative queries on graph databases

6.1 Introduction

Top-k queries play a critical role in various domains such as e-commerce, social networks, and multimedia and scientific databases. In traditional top-k formulations [87], given a function to quantify object relevance, the goal is to identify the k most relevant objects. However, if multiple objects in the answer set are highly similar to each other, the information content embedded in these objects is significantly diminished. For example, in drug design, chemists often want to extract from a molecular database only a small subset of molecules that exhibit high binding affinity toward certain protein targets while preserving other desirable properties such as low toxicity. More importantly, this subset should be compact and informationally dense to be manageable for a human-centric analysis and inexpensive assay. Traditional top-k framework would return the k best molecules according to a scoring function on the desired properties of the molecules. A hypothetical answer set under this framework is shown in Fig. 6.1(a), which contains two molecules: chloro-benzene and bromo-benzene. Even if both molecules



(a)



(b)

Figure 6.1: (a) An example of redundant information in traditional top-k answer sets. (b) A sample metric space where two objects are similar if they are located close to each other. Red objects denote those that are relevant.

are scored high by the top-k query function, a chemist is likely aware of the fact that replacing the chlorine with any other halogen would result in a molecule with similar properties. Therefore, to maximize the utility of the answer set, it is desirable to identify molecules that are high-scoring, structurally diverse, and representative of the different structural groupings in the database.

The possibility of information redundancy in traditional top-k formulations has ignited much interest in diversification of search results [88–95]. While different models of diversity exist, at their core, all models penalize the relevance of an object if it is not diverse enough with regard to the objects that have already been added to the answer set. Focusing on diversity however, is not enough to maximize the information content. Consider the metric space in Fig. 6.1(b), where red-filled objects denote those that are relevant. Assume g_1 has already been added to the answer set and g_3 and g_4 are equi-distant to g_1. In a diversity-aware model, both g_3 and g_4 have the same score since they are equally relevant and diverse. However, it is clear that g_3 is representative of a cluster of other relevant objects, whereas g_4 is a relevant outlier. In other words, a diverse cluster center such as g_3 encodes more information than a diverse outlier like g_4 and should therefore be preferred. Thus, to combat information redundancy, we address the following problem: *Given a budget k, which are the k relevant objects that best represent the underlying database?* More specifically, let $\mathbb{D}$ be a database, and each object $o \in \mathbb{D}$ is tagged with a feature vector $\overrightarrow{o}$. Users provide a query function $q : \overrightarrow{o} \rightarrow \{-1, 1\}$

to categorize o as relevant (1) or non-relevant (-1). In addition, an object o is *representative* of another object $o' \in \mathbb{D}$ if $d(o, o') \leq \theta$, for some distance function d and a user-provided distance threshold θ. Our goal is now to identify the k objects that are relevant and most representative of the remaining relevant objects in the database.

In this paper, we focus on the setting where each object is a graph. Such graphs can be used to model a wide range of scientific data such as chemical compounds [96], system call graphs [97], communication graphs [98], social networks [99] and gene interaction networks [100], as demonstrated in Table 6.1. Example 1 in Table 6.1 formalizes the problem for molecular libraries discussed earlier. Example 2 identifies the spectrum of information cascade patterns under a set of user-provided topics. A traditional top-k query is prone to identifying cascades from a single community of highly active users. For example, cascades arising out of populous countries such as USA, China or India are likely to eclipse remaining communities. This effect can be negated by being aware of the representative power of each cascade and favor those that are active and remain to be represented. Example 3 computes a summary of the most diverse set of bugs that occurred recently in software bug analysis. A traditional top-k query here is likely to identify function call graphs that share the same core bug-inducing subgraph. By rewarding representativeness and diversity, we can identify the entire spectrum of subgraphs that induce bugs. Finally, in social or collaboration networks such as DBLP [101] (example 4), each node (or user) can be described by a collection of tags to represent his/her interests or expertise areas. Given a set of expertise areas as query, we can identify the most knowledgeable groups, where each group is based on the neighborhood of a node. A traditional top-k query in this case is likely to return groups with large overlaps among them. In contrast, a top-k representative query finds neighborhoods that are relevant as well as non-overlapping. In a nutshell, the combination of feature vectors with graphs, as well as the need to maximize information content within a budget k, positions the proposed problem in a unique space, which has not been studied before. Our main contributions are as follows:

Table 6.1: Example applications with the combination of feature vectors and graphs. The properties of each graph g_i in the database is characterized by a feature vector $\overrightarrow{g_i}$. A graph is relevant if $q(\overrightarrow{g_i})$ is higher than some relevance threshold.

Property	**Example 1: Molecular Library**	**Example 2: Information Cascades**	**Example 3: Bug Analysis**	**Example 4: Social Networks**
Graph object g_i	A molecule in the database	An information cascade structure	A function call graph	d-hop neighborhood of an expert in a social network
Feature vector $\vec{g_i} = (v_{i,1}, v_{i,2}, \cdots, v_{i,m})$	Binding compatibility against m different proteins	Set of topics covered in the cascade	Count frequency over m days	Expertise areas of the social or collaboration group g_i
Target	Molecules with the most desirable properties	Most relevant cascades on a given topic set T	Frequently occurring bugs	Most knowledgeable groups of users on expertise areas E
Query function $q(\vec{g_i})$	$q(\vec{g_i}) = \sum_{j=1}^{m} v_{i,j}$	$q(g_i, T) = \frac{\lvert\vec{g_i} \cap T\rvert}{\lvert\vec{g_i} \cup T\rvert}$	$q(\vec{g_i}) = w^T \vec{g_i}$,	$q(\vec{g_i}, E) = \vec{g_i} \cap E$

- We propose a flexible and intuitive model for diversity to reward the representative power of the answer set. The flexibility is achieved by allowing users to define relevance at query-time and controlling the answer set size through a budget. While we focus on graphs, the proposed algorithm is generalizable to all metric spaces.
- We prove that top-k representative query is NP-hard and propose a technique to compute a constant factor approximation of the optimal answer set. We also show that no other polynomial time algorithm can provide a better approximation unless $P = NP$.
- We develop an index structure called *NB-Index* to index the θ-neighborhoods of graphs. Our proposed index structure employs a novel combination of Lipschitz embedding [102] and agglomerative clustering to expedite the answer set computation. In addition to fast answering of top-k representative queries, NB-Index also allows *interactive refinement* of θ to reach the optimal zoom level, while incurring minimal overhead cost.
- Empirical results on real graph datasets demonstrate higher information density in answer sets and a superior performance by up to 2 orders of magnitude speedup over state-of-the-art techniques.

6.2 Problem formulation

We assume a graph database $\mathbb{D} = \{g_1, \cdots, g_n\}$, where each graph g_i is tagged with a feature vector $\overrightarrow{g_i} = [g_{i,1}, \cdots, g_{i,m}]$ to characterize its properties. Based on a user-provided query function $q : \overrightarrow{g} \rightarrow \{-1, 1\}$, which operates on the feature vector $\overrightarrow{g}$, g is classified as either relevant (1) or irrelevant (-1). Our goal is to maximize the *representative power* of the answer set within a budget k.

Definition 6.2.1 TOP-k REPRESENTATIVE QUERIES: *Given a query function $q(\overrightarrow{g})$, distance threshold θ and a budget k, compute the set of graphs $\mathbb{A}$, such that*

$$\mathbb{A} = \arg\max_{\mathbb{S}}\{\pi_\theta(\mathbb{S}) \mid \mathbb{S} \subseteq \mathbb{L}_q,\ |\mathbb{S}| = k\} \tag{6.1}$$

where $\mathbb{L}_q = \{g \in \mathbb{D} \mid q(\overrightarrow{g}) = 1\}$ is the set of relevant graphs with respect to q, and $\pi_\theta(\mathbb{S})$ is the representative power of $\mathbb{S}$.

To quantify the representative power of a graph or a set of graphs, we first define the θ-*neighborhood* of a graph.

Definition 6.2.2 θ-NEIGHBORHOOD. *The θ-neighborhood of a graph g, denoted as $N_\theta(g)$, contains all relevant database graphs within a distance threshold θ from g.*

$$N_\theta(g) = \{g' \in \mathbb{L}_q \mid d(g, g') \leq \theta\} \tag{6.2}$$

where $d(g, g')$ is the classical graph edit distance [103,104] and θ is a user provided threshold. In other words, g is a structural representative of all graphs in its θ-neighborhood $N_\theta(g)$. Consequently, it is not desirable to have two graphs with a large overlap between their θ-neighborhoods in the answer set. By combining these intuitions, the representative power

$\pi_\theta(\mathbb{S})$ of a set of graphs $\mathbb{S}$ is defined as the proportion of relevant graphs represented by $\mathbb{S}$.

$$\pi_\theta(\mathbb{S}) = \frac{\left|\bigcup_{g\in\mathbb{S}} N_\theta(g)\right|}{|\mathbb{L}_q|} \tag{6.3}$$

The proposed model captures as much of the various relevant structural groupings as possible within the provided budget. As a natural consequence of maximizing the representative power, the answer set also favors structural diversity. For brevity's sake, hereon, we denote the representative power $\pi_\theta(\{g\})$ of a graph g as $\pi_\theta(g)$. Additionally, the notations $\pi(g)$ and $N(g)$ are used to denote the representative power and θ-neighborhood, respectively, for arbitrary θ's. Our notation is summarized in Table 6.2.

6.3 Existing diversity models

As discussed in Sec. 7.1, a number of models exist to favor diversity in the top-k answer set. In this section, we discuss two recent models that are closest to ours.

6.3.1 DisC

DisC [88] approach recognized the need to go beyond diversity and focus on the representative power of the answer set. Given the set of relevant objects $\mathbb{S}$, DisC attempts to find the smallest set of objects that represent all relevant objects. Mathematically, DisC computes an answer set $\mathbb{A}$, such that $\forall o_1 \in \mathbb{S}, \exists o_2 \in \mathbb{A}$, where $d(o_1, o_2) \leq \theta$ and $\forall o_2, o_3 \in \mathbb{A}, o_2 \neq o_3, d(o_2, o_3) > \theta$. There are two main differences between our approach and DisC.

1. Static set of relevant objects: DisC assumes that the set of relevant objects in a given database is static. At times, the set of relevant objects are query-dependent even though the underlying database may be static. For example, in a database of information cascade graphs, one might be interested in only those that discuss a certain topic, such as sports. Owing to this assumption, the index structure proposed in DisC needs to be rebuilt whenever the relevant objects change. In this paper, we focus on a dynamic setting where the relevant objects are defined at query time by the user through a query function $q(\cdot)$.

2. Coverage of answer set: DisC requires the coverage of all relevant objects while we quantify the coverage of an answer set and seek to maximize it. DisC's requirement to represent all relevant objects can make the answer set too large. Such scenarios occur when a dataset contains some objects that are relevant but not clustered together with other relevant objects. Graph g_4 in Fig. 6.1(b) is an example of such objects. These relevant outliers have low representative power. Consequently, their presence in the answer set dilutes the *compression ratio* $\frac{|\mathbb{S}|}{|\mathbb{A}|}$, which is the average number of relevant objects represented by each object in the answer set.

To study this phenomenon empirically, we examined the answer set produced by DisC in a graph database representing the DUD molecular repository [105]. On average, the graphs contain 26 vertices and 28 edges. We use the classical graph edit distance [103, 104] for this experiment under a distance threshold $\theta = 10$ to define if a graph g_1 represents graph g_2. First, we identify the set of relevant molecules that are active against the enzyme Acetylcholine ezterase (AChE), a key component for curing Alzheimer's disease [106]. Next, we plot the growth rate in the answer set size as the number of relevant objects is varied. Fig. 6.2(a) presents the results. The growth rate is almost linear and on average, the answer set is one third of the number of relevant molecules. Clearly, a higher compression ratio is desirable and this is compromised due to the presence of relevant outliers. More critically however, there is no control over the answer set size. This is especially true under the constraints of a small budget. Such kind of top-k scenarios routinely arise where either a manual analysis is required on the answer set objects, or higher level analytics of the identified objects is an expensive procedure. In essence, the top-k representative query problem is similar to that of dimensionality reduction

Table 6.2: Summary of the notations used.

Notation	Explanation
$q(\cdot)$	User-provided relevance function that maps each object to {-1,1}.
$\mathbb{L}_q$	The set of relevant graphs with respect to relevance function $q(\cdot)$.
θ	User-provided distance threshold to quantify if two graphs are similar.
$N(g)$	All graphs that are within distance θ of g, and therefore *represented* by g.
$\pi(\mathbb{S})$	The representative power of a set of graphs as defined in Eq. 6.3.
$d(g, g')$	The edit distance between graphs g and g'.
$d_v(g, g')$	The vantage distance between graphs g and g' as defined in Def. 6.6.2.

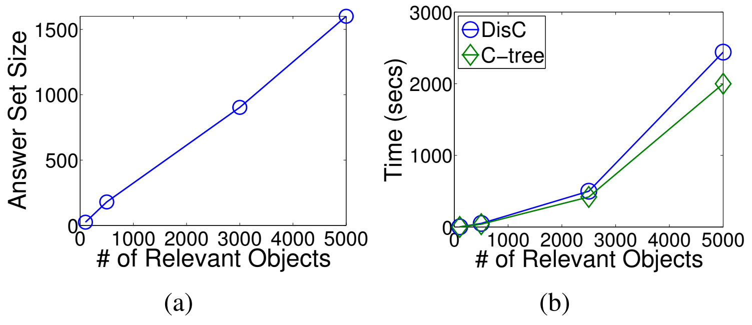


Figure 6.2: (a) The growth rate of DisC answer set size against the number of relevant objects. (b) The growth rate in the running time of the DisC model in the presence of graphs.

in high-dimensional spaces. In both tasks, the goal is to make the dataset more manageable by operating on a smaller subset of a desirable size and yet capture as much information as possible. The key in both these tasks is to allow the user to control the size (or dimensionality) of the subset. Accordingly, given a budget k, our goal is to select the k exemplars that best represent the relevant objects.

6.3.2 Tuning diversity models to reward representativeness

One of the most generic diversity models is proposed in [7]. Let us refer to this technique as *DIV*. Given a database of objects $\mathbb{D}$, the goal is to identify the subset $\mathbb{S} \subseteq \mathbb{D}$ of k objects, such that $\sum_{g \in \mathbb{S}} score(g)$ is maximized and $\forall g_1, g_2 \in \mathbb{S},\ d(g_1, g_2) > \theta$. $score(g)$ denotes the value of an object (or graph) g. In our problem, the goal is to maximize representativeness. Thus, by assigning $\pi(g)$ as the score of graph g, we can attempt to maximize the representativeness of the answer set. However, in this model, $score(g)$ of a graph g is assumed to be static and independent of other objects in the answer set. In contrast, in our model, $\pi(g)$ is dependent on other graphs that are already in the answer set. More formally, for DIV, when $\pi(g)$ is used as the score, even in the presence of the constraint $\forall g_1, g_2 \in \mathbb{S},\ d(g_1, g_2) > \theta$, the maximization function $score(\mathbb{S}) = \sum_{\forall g \in \mathbb{S}} score(g) = \sum_{\forall g \in \mathbb{S}} \pi(g) \neq \pi(\mathbb{S})$. Consequently, the model cannot be used to solve our problem. To ensure score independence, we need to enforce the stricter constraint of $\forall g_1, g_2 \in \mathbb{S},\ d(g_1, g_2) > 2\theta$. This constraint, however, poses the risk of ruling out highly representative graphs from inclusion in the answer set and therefore will not maximize $\pi(\mathbb{S})$.

From the indexing perspective, three different schemes are proposed in DIV. However, all of them rely on an upper bound computation framework, which assumes scores of graphs in the answer set to be mutually independent. As a result, the proposed indexing techniques cannot be used in our problem. Beyond the independence assumption, DIV constructs a *diversity-graph* where each node is a graph in the database and two nodes are connected if $d(g_1, g_2) \leq \theta$. Since θ is provided at query-time in our model, the diversity-graphs need to be computed online, a compute bottleneck. As a result, the flexibility of the model is compromised. We analyze the impact of mutual independence assumption on answer set quality and the non-scalability of the approach when applied to our problem in Sec. 6.8.

6.4 Properties of top-k representative queries

In this section, we analyze the complexity of the proposed problem and investigate the theoretical guarantees that can be achieved.

6.4.1 NP-hardness

Theorem 6.4.1 *Top-k representative query is NP-hard.*

PROOF: We reduce the *Set Cover problem* to the problem of answering top-k representative queries. The Set Cover problem is defined as follows: Given a universe of elements $U = \{e_1, e_2, \cdots, e_n\}$ and a collection of subsets $S = \{\ S_1,\ S_2,\ \cdots,\ S_m\}$, the Set Cover problem seeks to determine whether there exists a collection $S' \subseteq S$ of k subsets, such that $\bigcup_{S_i \in S'} S_i = U$.

Given an arbitrary instance of the Set Cover problem, we construct a graph database containing three sets of graphs: D_1, D_2, and D_3 all of which are relevant. D_1 contains graph s_i corresponding to each set $S_i \in S$, and D_2 contains graph u_j corresponding to each element $e_j \in U$. Graph $u_j \in N(s_i)$ if and only if $e_j \in S_i$. From symmetry of graph edit distance, the θ-neighborhoods of all graphs in D_2 can be computed analogously. Now, let $x = max\{\pi(u)|u \in D_2\}$. D_3 contains $|S|$ disjoint groups of graphs, such that each group G_i contains x graphs and all graphs in G_i are in the θ-neighborhood of graph $s_i \in D_1$. In other words, D_3 contains $x|S|$ graphs and it ensures that any graph in D_1 has a higher representative power than graphs in D_2 or D_3. We now perform top-k representative query on this database.

It is easy to see that there is a set cover of size k if and only if there exists an answer set $\mathbb{A}$ where $\pi(\mathbb{A}) = \frac{|D_2|+k(x+1)}{|D_1|+|D_2|+|D_3|}$. From our construction, no graph from either D_2 or D_3 will be selected in $\mathbb{A}$. Additionally, the number of graphs added to $\pi(\mathbb{A})$ from D_3 is a constant number kx. As a result, $\pi(\mathbb{A})$ is maximized when, in addition to the k answer set graphs and their corresponding kx neighbors from D_3, all graphs in D_2 are in $\pi(\mathbb{A})$. On the other hand, if subsets S_{i_1}, S_{i_2}, $\cdots$, S_{i_k} form a set cover, then including the corresponding graphs in answer set $\mathbb{A}$ results in $\pi(\mathbb{A}) = \frac{|D_2|+k(x+1)}{|D_1|+|D_2|+|D_3|}$. □

6.4.2 Submodularity

A function $f(.)$ is submodular if the marginal gain from adding an element to a set S is at least as high as the marginal gain from adding it to a superset of S. Mathematically, it satisfies:

$$f(S \cup \{o\}) - f(S) \geq f(T \cup \{o\}) - f(T) \tag{6.4}$$

for all elements o and all pairs of sets $S \subseteq T$. For submodular and monotone functions, the greedy algorithm of iteratively adding the element with the maximum marginal gain approximates the optimal solution within a factor of $(1 - \frac{1}{e})$ [107]. We omit the proof of monotonicity for Eqn. 6.3 due to space limitations.

Theorem 6.4.2 *Eqn. 6.3 is submodular.*

PROOF BY CONTRADICTION: Assume,

$$\pi(T \cup \{g\}) - \pi(T) > \pi(S \cup \{g\}) - \pi(S) \tag{6.5}$$

where S and T are sets of graphs, such that $S \subseteq T$, and $g \in \mathbb{D}$ is the graph being added. From Eqn. 6.5, we can say:

$$N(g)\backslash N(T) > N(g)\backslash N(S)$$
$$\text{or, } S \nsubseteq T \tag{6.6}$$

which contradicts the assumption that $S \subseteq T$. □

Thus,
$$\pi(\mathbb{A}_{greedy}) \geq (1 - \frac{1}{e})\pi(\mathbb{A}^*) \tag{6.7}$$

where $\mathbb{A}_{greedy}$ is the answer set computed using a greedy algorithm and $\mathbb{A}^*$ is the optimal answer set. Since the scoring function is normalized, i.e., $\pi(\emptyset) = 0$, no other polynomial time algorithm can provide a better approximation guarantee than $(1-\frac{1}{e})$ unless $P = NP$, as proved in [108].

Algorithm 6 Baseline-greedy($q(\cdot), \theta, k$)

```
1: Compute $\mathbb{L}_q$
2: $\mathbb{A} \leftarrow \emptyset$
3: while $|\mathbb{A}| < k$ do
4:     $g^* \leftarrow \arg\max_g\{\pi(\mathbb{A} \cup g) - \pi(\mathbb{A}) \mid g \in \mathbb{L}_q\}$
5:     $\mathbb{A} \leftarrow \mathbb{A} \cup g$
6:     for $\forall g \in \mathbb{L}_q \backslash \mathbb{A}$ do
7:         $N(g) \leftarrow N(g) \backslash N(g^*)$
8:     end for
9: end while
10: return $\mathbb{A}$
```

6.5 The simple greedy approach

Alg. 6 presents the pseudocode for the baseline greedy approach. From Theorem 6.4.2, Alg. 6 guarantees an approximation of $(1 - \frac{1}{e})$ or better. Unfortunately, in the presence of graphs, this approach is not scalable. The bottleneck lies in the neighborhood update step (lines 6-7), which is performed at each iteration and requires $O(n^2)$ edit distance computations in the worst case. Computing the edit distance between two graphs is NP-hard [104]. As a result, Alg. 6 is not scalable to large databases. To mitigate this bottleneck, we need to index the neighborhood update step. More precisely, given the θ-neighborhoods of two graphs, we need to compute their overlap without incurring a quadratic computation cost with respect to the neighborhood sizes.

Indexing graph edit distance has been studied in the context of top-k nearest neighbor queries [103, 104]. The demands of indexing θ-neighborhoods, however, are different.

1. Overlap quantification: Instead of identifying the k nearest neighbors to a query graph, our goal is to compute the extent of the overlap between θ-neighborhoods of two graphs without incurring a quadratic computational cost. This ensures an efficient update of their representative power at each iteration.

2. Order independence: In our problem, ordering neighbors based on distance is not necessary; just detecting containment within the θ-neighborhood is sufficient. In top-k similarity queries, the ordering is central to solving the problem.

To establish the scalability challenge in the presence of graphs, we study the running time growth rate of Alg. 6 against database size. Furthermore, we also investigate the impact of two index structures that are both built for indexing nearest neighbor queries: the state-of-the-art graph indexing technique C-tree [103] and the index structure proposed by DisC, which is an adaptation of M-tree [109]. Fig. 6.2(b) shows the results. Regardless of the underlying indexing technique, it takes more than 35 minutes to compute the answer even on a small dataset containing only 5000 objects. This result highlights the need to go beyond traditional graph indexing techniques, which optimize nearest neighbor queries. To scale our problem, we need to index θ-neighborhoods. One could consider pre-computing and storing the θ-neighborhoods of each relevant graph. This approach avoids the need to compute edit distance at query time. However, since both θ and $q(\cdot)$ are given at query time, such a strategy degenerates to storing the entire distance matrix of the graph database, resulting in $O(n^2)$ storage.

6.6 Indexing θ-neighborhoods

Equipped with the insights derived from Sec. 6.5, we proceed toward analyzing the properties of graph edit distance, and then design an index structure called *NB-Index*.

6.6.1 Triangular inequality based pruning

The graph edit distance satisfies triangular inequality when the individual vertex and edge distances are metric. Mathematically, $d(g_1, g_2) + d(g_2, g_3) \geq d(g_1, g_3)$ for any three graphs g_1, g_2, g_3. Based on this property, we can easily derive the following theorem.

Theorem 6.6.1 *For any two graphs g_1 and g_2, if $d(g_1, g_2) > 2\theta$, $N(g_1) \cap N(g_2) = \emptyset$.*

Theorem 6.6.1 highlights the property that, once the graph providing the highest marginal gain is added to the answer set, only the θ-neighborhoods of those relevant graphs within a distance of 2θ from that new graph will need to be updated. While this property does reduce the computational cost, scalability cannot be guaranteed. First, its efficiency depends on the value of θ. The smaller the value of θ is, the more efficient this property becomes. More importantly, to have a significant impact on the scalability, a technique must be designed to efficiently update the θ-neighborhoods of graphs which are within the 2θ boundary. Thus, to further improve scalability, we employ a Lipschitz embedding [102] of the metric space and design the concept of *Vantage Ordering*.

6.6.2 Vantage Ordering

Vantage Orderings (VO) perform a Lipschitz embedding of the metric space to speed up the indexing of the θ-neighborhoods.

Definition 6.6.1 VANTAGE POINT AND VANTAGE ORDERING: *Given a metric space $\mathcal{M} : \{\mathbb{D}, d : (\mathbb{D}, \mathbb{D}) \rightarrow \mathbb{R}_+\}$, let $v \in \mathbb{D}$ be a randomly selected graph. $\forall g \in \mathbb{D}$, $\mathcal{M}$ is embedded into a 1-dimensional feature space $\mathcal{F}_v$, where the feature vector of a graph g is $[d(v, g)]$. The 1-dimensional ordering of graphs in $\mathcal{F}_v$ is termed the Vantage Ordering of $\mathcal{M}$ with respect to the vantage point v.*

As defined above, VO captures a feature space representation of $\mathcal{M}$ from the viewpoint of the vantage point (VP) v. To disambiguate the resultant feature space of Lipschitz embedding from the feature vector representation of a graph on which the query function operates, hereon, we use the term vantage space to denote $\mathcal{F}_v$. Now, to quantify the distance between two graphs from a VP's viewpoint in $\mathcal{F}_v$, we introduce the concept of *Vantage Distance*.

Definition 6.6.2 VANTAGE DISTANCE $d_v(g, g')$: *Given a vantage point v, the vantage distance $d_v(g, g')$ between any two graphs $g,\ g' \in \mathbb{D}$ is $|d(v, g) - d(v, g')|$.*

Now, we observe the following.

Theorem 6.6.2 *Given a VP v, if $d_v(g, g') > \theta$, $g' \notin N(g)$.*

PROOF: From triangular inequality, $d(g, g') \geq |d(v, g) - d(v, g')| = d_v(g, g') > \theta$. Thus, $g' \notin N(g)$. □

Theorem 6.6.2 provides a mechanism to bound the maximum vantage distance between any two graphs g, g', where $g' \in N(g)$. It is straightforward to see that the information retained in the vantage space can be boosted by employing a set of VPs $\mathbb{V}$ rather than a single one. We denote this vantage space as $\mathcal{F}_{\mathbb{V}}$. In $\mathcal{F}_{\mathbb{V}}$, we can deduce the following theorem.

Theorem 6.6.3 *Let $\hat{N}(g)$=$\{g'|d_v(g, g')\leq \theta \; \forall v\in \mathbb{V}\}$. $\hat{N}(g) \supseteq N(g)$.*

Theorem 6.6.3 allows us to compute an upper bound on the actual θ-neighborhoods based on the VOs. The VOs can be pre-computed as part of indexing procedure, and thus, computing $\hat{N}(g)$ only requires $|\mathbb{V}|$ linear scans. Furthermore, since the entire computation is performed on the vantage space, the expensive NP-hard computations of edit distances are required only on graphs $g' \in \hat{N}(g)$.

Choosing VPs

The performance gain achieved due to VPs is directly proportional to the *False Positive Rate (FPR)*, which is the probability of a graph being in $\hat{N}(g)$ but not in $N(g)$. FPR is quantified as follows under the simplifying assumption that the distances $d(g, g')$ and $d(g, g'')$ are independent.

$$\begin{aligned} FPR &= P(g' \in \hat{N}(g) \backslash N(g)) \\ &= P(d(g, g') > \theta) \prod_{v \in \mathbb{V}} P(d_v(g, g') \leq \theta) \end{aligned} \tag{6.8}$$

It is clear from Theorem 6.6.3 that the higher the number of VPs, the better the tightness of $\hat{N}(g)$. On the other hand, both the computational cost of $\hat{N}(g)$ and the storage footprint of the

VOs increase with $|\mathbb{V}|$. It is therefore critical to have a deep understanding of the relationship between the FPR and $|\mathbb{V}|$. Towards that goal, given the number of VPs $|\mathbb{V}|$, we derive theoretical bounds on the FPR. The FPR is dependent on the underlying distribution of the graph distances. We thus choose the two most commonly encountered distributions, *Gaussian* and *Uniform*, and derive the bounds below.

1. Normally distributed metric space: Let us assume that the distance between any two randomly chosen graphs is distributed normally where $d(g, g') \in \mathcal{N}(\mu, \sigma^2)$ and $m\theta$ is the diameter of the metric space $\mathcal{M}$. Under this assumption,

$$\begin{aligned} P(d(g,g') > \theta) &= P\left(\frac{d(g,g') - \mu}{\sigma} > \frac{\theta - \mu}{\sigma}\right) \\ &= P\left(Z > \frac{\theta - \mu}{\sigma}\right) \\ &= 1 - \phi\left(\frac{\theta - \mu}{\sigma}\right) \end{aligned} \tag{6.9}$$

where $Z = \mathcal{N}(0,1)$ is the standard normal variable and $\phi(x)$ is the cumulative distribution function of Z.

The next step in computing the FPR is to compute $P(d_v(g, g') \leq \theta)\ \forall v \in \mathbb{V}$. With respect to a vantage point v, if $d(g, v) = x$, then $P(d_v(g, g') \leq \theta) = P(x - \theta \leq d(g', v) \leq x + \theta)$. Therefore,

$$
\begin{aligned}
& P(d_v(g,g') \leq \theta) \\
&= \int_0^{m\theta} P\left(d(g,v) = x\right) P\left(x - \theta \leq d(g',v) \leq x + \theta\right) \mathrm{d}x \\
&= \int_0^{m\theta} P\left(d(g,v) = x\right) P\left(\frac{x - \theta - \mu}{\sigma} \leq Z \leq \frac{x + \theta - \mu}{\sigma}\right) \mathrm{d}x \\
&\leq \int_0^{m\theta} P\left(d(g,v) = x\right) P\left(\frac{\mu - \theta - \mu}{\sigma} \leq Z \leq \frac{\mu + \theta - \mu}{\sigma}\right) \mathrm{d}x \\
&= \left(\phi\left(\frac{\theta}{\sigma}\right) - \phi\left(\frac{-\theta}{\sigma}\right)\right) \int_0^{m\theta} P\left(d(g,v) = x\right) \mathrm{d}x \\
&= 2\phi\left(\frac{\theta}{\sigma}\right) - 1 \qquad (6.10)
\end{aligned}
$$

Combining Eq. 6.9 and Eq. 6.10, the FPR for $|\mathbb{V}|$ vantage points can be expressed as:

$$
FPR \leq \left(1 - \phi\left(\frac{\theta - \mu}{\sigma}\right)\right)\left(2\phi\left(\frac{\theta}{\sigma}\right) - 1\right)^{|\mathbb{V}|} \qquad (6.11)
$$

2. Uniformly distributed metric space: Let us assume that $d(g,g') \in \mathcal{U}(0, m\theta)$ where m is the diameter of the metric space. Now, since the VPs are selected randomly, for any graph g, each dimension $d(v,g)$ is distributed uniformly. The FPR can therefore be quantified as:

$$
\begin{aligned}
FPR &= \frac{m\theta - \theta}{m\theta} \prod_{v \in \mathbb{V}} \frac{\theta}{m\theta} \\
&= \frac{m-1}{m} \frac{1}{m^{|\mathbb{V}|}} \qquad (6.12)
\end{aligned}
$$

Eqns. 6.11 and 6.12 allow us to select the appropriate number of VPs for a desired FPR. What is particularly attractive is that the number of VPs required is independent of the dataset size as long as the distance distribution remains the same.

6.6.3 Updating representative power based on clusters

Sec. 6.6.2 provides an upper bound on the θ-neighborhood of a graph based on a vantage space analysis. In this section, we analyze the structural space directly and develop techniques to update the representative power of a graph in the presence of clusters. The technique developed in this section allows *batch updates*: a single calculation to compute upper bounds on the representative power of a cluster of similar graphs. Here, a cluster is a set of structurally similar graphs. Fig. 6.3 shows a sample structural space of graphs with four clusters, c_1-c_4, depicted as shaded circles. Clusters c_1 and c_2 are contained within c_3. Consider two relevant graphs g_1 and g_2, where g_2 is contained within cluster c_3. The two dashed rings centered on g_1 with radii of θ and 2θ represent $N(g_1)$ and the space of graphs whose θ-neighborhoods overlap with $N(g_1)$. g_2 represents one such graph.

Now, assume that g_1 has been added to the answer set in the last iteration, and the θ-neighborhoods of the rest of the relevant graphs need to be updated. Since c_4 is more than 2θ away from g_1, neighborhoods of all graphs in c_4 are unaffected and don't need to be updated. In contrast, the neighborhoods of all graphs in clusters c_1, c_2 and c_3 are affected and need updating. Moreover, both c_1 and c_2 are fully contained within the θ-neighborhood of g_1. Thus, for any graph in c_1, their updated θ-neighborhoods must not contain any of the graphs in c_1. The same rule applies for c_2 as well. Whereas, this rule does not work for c_3 because it is not fully contained within the θ-neighborhood of g_1. However, graphs in c_3, such as g_2, could be affected. More specifically, $N(g_2)$, depicted using the dashed ring centered on g_2, needs to be reduced by its overlap with $N(g_1)$, depicted as the ring of radius θ centered on g_1. Furthermore, since c_1 and c_2 are completely contained within this overlap region, $N(g_2)$ must at least be reduced by the combined "area" of c_1 and c_2.

The above analysis gives us insights into how upper bounds can be computed on θ-neighborhoods based on the location of clusters. To formalize these insights, we first introduce the definitions

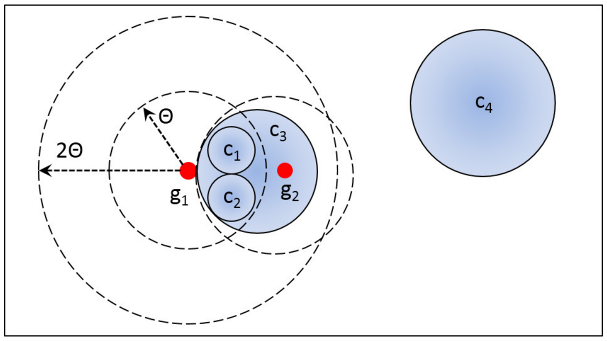


Figure 6.3: A representation of the structural space in the presence of clusters.

of *cluster radius* and *cluster diameter* with respect to a metric space. Since a cluster is essentially a set of graphs, we use set notations in the following definitions and proofs. Let $centroid(c)$ denote the centroid graph of a cluster c.

Definition 6.6.3 CLUSTER RADIUS AND DIAMETER: *The radius of a cluster is the maximum of all the pairwise distances between the centroid and other constituent graphs in the cluster. Similarly, the diameter is the largest of all the pairwise distances.*

$$radius(c) = max\,\{d(centroid(c), g)\ \forall g \in c\}$$

$$diameter(c) = max\,\{d(g, g')\ \forall g,\ g' \in c\} \leq 2 \times radius(c)$$

Now, given a cluster c, we can derive the upper bound $d_{ub}(g, c)$ and lower bound $d_{lb}(g, c)$ for the distance between a graph g and $\forall g' \in c$ using triangular inequality.

$$d_{ub}(g, c) = d(g, centroid(c)) + radius(c)$$

$$d_{lb}(g, c) = max\,\{0, d(g, centroid(c)) - radius(c)\}$$

Now, let g represent the latest graph added to the answer set, c be a cluster, and $\pi(g')^*$ and $\pi(g')$ be the representative power of graph g' after and before g is added to the answer set respectively. Additionally, let $c_q = c \cap \mathbb{L}_q$ be the subset of relevant graphs in c. Using the distance bounds stated above, we have the following theorems:

Theorem 6.6.4 *If $d_{lb}(g, c) > 2\theta$, then $\forall g' \in c$, $\pi(g')^* = \pi(g')$.*

PROOF: We omit the proof for its simplicity. □

Theorem 6.6.5 *If $d_{ub}(g, c) \leq \theta$ and $diameter(c) \leq \theta$, then $\forall g' \in c$, $\pi(g')^* \leq \pi(g') - \frac{|c_q|}{|\mathbb{L}_q|}$.*

PROOF: Since $diameter(c) \leq \theta$, for any graph $g' \in c$,

$$N(g') \supseteq c.$$

Since $d_{ub}(g, c) \leq \theta$, $N(g) \supseteq c$. Thus,

$$N(g) \cap N(g') \supseteq c.$$

$$\text{or, } \pi(g')^* \leq \pi(g') - \frac{|c_q|}{|\mathbb{L}_q|} \qquad \square$$

Theorem 6.6.6 *Let c be a cluster and $\mathbb{C} = \{c' \mid c' \subseteq c, c'$ satisfies Theorem 6.6.5$\}$ be the set of sub-clusters of c that satisfy Theorem 6.6.5. Additionally, all clusters in $\mathbb{C}$ are disjoint, i.e., $\forall c_i,\ c_j \in \mathbb{C}, c_i \cap c_j = \emptyset$. If $\theta \leq d_{ub}(g, c) \leq 2\theta$ and $diameter(c) \leq \theta$, then $\forall g' \in c$, $\pi(g')^* \leq \pi(g') - \sum_{c' \in \mathbb{C}} \frac{|c'_q|}{|\mathbb{L}_q|}$*

PROOF: Since $diameter(c) \leq \theta$, for any graph $g' \in c$,

$$N(g') \supseteq c.$$

Now, $N(g) \supseteq \bigcup_{c' \in \mathbb{C}} N(c')$. Thus,

$$N(g) \cap N(g') \supseteq \bigcup_{c' \in \mathbb{C}} N(c').$$

Since clusters in $\mathbb{C}$ are disjoint,

$$\pi(g')^* \leq \pi(g') - \sum_{c' \in \mathbb{C}} \frac{|c'_q|}{|\mathbb{L}_q|} \qquad \square$$

The three theorems above allow us to obtain tight upper bounds on the representative power of a graph by performing a small number of distance computations. More specifically, instead of computing the actual θ-neighborhood via pairwise distance computations with all database graphs, an upper bound can be computed by leveraging the presence of clusters in a database. The clusters can be precomputed while indexing and thereby facilitating fast computation of upper bounds during query time. Besides, the efficiency is further magnified since Theorems 6.6.4-6.6.6 provide bounds on an entire cluster of similar graphs by performing a single computation operation.

An important constraint that drives the efficiencies of Theorems 6.6.5 and 6.6.6 are the diameters of clusters. If the diameter of any cluster is above θ, then it cannot be used to estimate the overlap between the θ-neighborhoods of two graphs. Therefore, the lower the value of θ, the less effective Theorems 6.6.5 and 6.6.6 become. Fortunately, the efficiency of Theorem 6.6.4 is inversely proportional to θ. As a result, the combined efficiency is *elastic* in nature; for a small θ, Theorem 6.6.4 is effective, whereas for a large θ, Theorems 6.6.5 and 6.6.6 are effective.

6.6.4 NB-Index

Secs. 6.6.2 and 6.6.3 provide us with two efficient mechanisms to compute upper bounds on the representative power of the database graphs: one based on VO and the other based on clusters. In this section, we develop the index structure called *NB-Index*, which unifies the two different approaches into one coherent framework.

NB-Index contains two components:

1. Vantage Points: A set of VPs, $\mathbb{V}$, is chosen to capture a feature space representation of database graphs. For each VP, NB-Index maintains the VO of the entire database.

2. NB-Tree: A hierarchical clustering is performed on the database to group structurally similar graphs into disjoint sub-spaces. In our clustering procedure, disjoint clusters are formed in a top-down, recursive manner to form a tree. In this tree, each leaf node is a graph and non-leaf nodes are clusters of its children. As a result, two clusters can overlap only if one is a descendant of the other. Each non-leaf node stores the centroid, radius, and diameter of the corresponding cluster. At leaf nodes, only the feature vectors of the corresponding graphs are stored. While building the tree, we ensure that each non-leaf node contains at most b children. The fan-out controls the desired depth of the tree. For an on-disk implementation, b should be chosen based on the disk-page size for optimizing the cost of node look-ups. On the other hand, in a memory-resident implementation, a small b yields better performance.

The clustering procedure starts at the root node, which contains the entire database. First, b graphs are chosen as pivots to partition the dataset. The first of the b pivots is chosen randomly. For the second pivot, we choose the graph that is farthest from first one. Continuing in the same manner, in each iteration, the pivot is the graph whose minimum distance to the already chosen pivots is the highest. The process is repeated through b iterations. Next, the pivots are assigned as cluster centroids, and all remaining graphs in the database are assigned to the cluster with the closest centroid. Finally, the process is repeated recursively on each of the clusters till the size of a cluster drops below b.

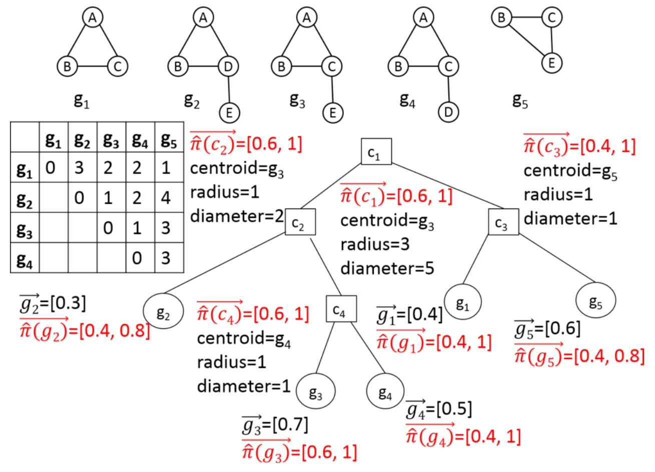
As already discussed, computing edit distance is NP-hard, and the indexing procedure requires a large number of pairwise edit distance computations. Although indexing is an offline procedure, it is desirable for it to be as efficient as possible. Towards that goal, the VPs are computed first and are utilized to expedite NB-Tree construction. More specifically, to determine if pivot p is the closest pivot to a graph g, we first compute the VP-based distance $\forall v \in \mathbb{V}$,

$\max\{d_v(g,p)\}$, which is a lower bound on the actual edit distance $d(g,p)$. If the lower bound is larger than the currently closest edit distance between g and any of the already evaluated pivots, then p is discarded from consideration. Similarly, to compute the radius of a cluster from the centroid p, if the upper bound $\forall v \in \mathbb{V}$, $\min\{d(v,p) + d(v,g)\}$ is smaller than the current radius, then g is ignored. As a result, expensive edit distances are computed on a small minority of pairs. The diameter of a cluster is set to the summation of the two largest distances from the centroid.

Example 6.6.1 *Fig. 6.4 demonstrates the NB-Tree built on the corresponding graph database at $b = 2$. Although we do not compute the entire edit distance matrix during index construction due to its quadratic computation cost, it is shown in the figure to help guide the readers. For simplicity, we use a 1D feature vector, denoted as $\vec{g_i}$, to characterize each graph, and these are chosen arbitrarily. The explanation of $\hat{\pi}$-vectors (highlighted in red), which are computed during query time, is discussed in the next section.*

Storage Cost: The storage cost for maintaining the VO of the entire database is $O(|\mathbb{V}||\mathbb{D}|)$, where $\mathbb{V}$ is the set of VPs, and $\mathbb{D}$ is the graph database. The second component of the index is the NB-Tree. Assuming a balanced branching factor of b, the height of NB-Tree is bounded within $\log_b |\mathbb{D}|$. Since the number of nodes at each level of NB-Tree follows a geometric progression, the storage cost is bounded by $O(\frac{b|\mathbb{D}|-1}{b-1})$. The total storage cost is therefore $O(\frac{b|\mathbb{D}|-1}{b-1}) + |\mathbb{V}||\mathbb{D}|)$.

Index Construction Time: For each VP, we need to compute its distance to all graphs in the database. Thus, the cost of constructing the VO is $O(|\mathbb{V}||\mathbb{D}|)$. For NB-Tree, at each level, each graph is compared to b pivots to identify the closest one. Since the height of the tree is bounded by $O(\log_b |\mathbb{D}|)$, a graph is compared to b pivots $\log_b |\mathbb{D}|$ times. Thus, the computation cost is $O(|\mathbb{D}| \times b \log_b |\mathbb{D}|)$. Combining the costs of both VPs and the NB-Tree, the total index construction time is $O\left(|\mathbb{D}| \times (|\mathbb{V}| + b \log_b |\mathbb{D}|)\right)$.



	g_1	g_2	g_3	g_4	g_5
g_1	0	3	2	2	1
g_2		0	1	2	4
g_3			0	1	3
g_4				0	3

g_1
g_2
g_3
g_4
g_5
$\overrightarrow{\hat{\pi}(c_2)}$=[0.6, 1]
centroid=g_3
radius=1
diameter=2
c_1
$\overrightarrow{\hat{\pi}(c_1)}$=[0.6, 1]
centroid=g_3
radius=3
diameter=5
$\overrightarrow{\hat{\pi}(c_3)}$=[0.4, 1]
centroid=g_5
radius=1
diameter=1
c_2
c_3
$\overrightarrow{g_2}$=[0.3]
$\overrightarrow{\hat{\pi}(g_2)}$=[0.4, 0.8]
$\overrightarrow{\hat{\pi}(c_4)}$=[0.6, 1]
centroid=g_4
radius=1
diameter=1
c_4
$\overrightarrow{g_1}$=[0.4]
$\overrightarrow{\hat{\pi}(g_1)}$=[0.4, 1]
$\overrightarrow{g_5}$=[0.6]
$\overrightarrow{\hat{\pi}(g_5)}$=[0.4, 0.8]
$\overrightarrow{g_4}$=[0.5]
$\overrightarrow{\hat{\pi}(g_4)}$=[0.4, 1]
$\overrightarrow{g_3}$=[0.7]
$\overrightarrow{\hat{\pi}(g_3)}$=[0.6, 1]

Figure 6.4: The NB-Tree built on the shown 5 graphs at $b = 2$. The $\hat{\pi}$-vectors, highlighted in red, are maintained only during query time. For simplicity, we assume all graphs are relevant.

6.7 Query processing

The goals of the query processing algorithm are two-fold:

1. Flexibility: Maximize flexibility by allowing query time definitions of the relevance function $q(\cdot)$ and the distance threshold θ.

2: Interactive Refinement: While domain scientists have a crude idea on the appropriate θ for the task at hand, the optimal one might often be reached through a series of trials. This operation is similar in nature to finding the optimal "zoom" level in Google Maps. In other words, fine-tuning θ is an interactive procedure and a good index structure should adapt to those needs by performing refinements faster than a brand new query with a new relevance function.

To achieve the above outlined goals, the query processing algorithm runs in two phases: the *initialization* phase and the *search-and-update* phase. The initialization phase is performed only once as long as $q(\cdot)$ remains unchanged. For any subsequent refinements of θ, only the search-and-update phase needs to be repeated.

1. Initialization phase: VPs drive the initialization phase. First, VOs are used to compute a $\hat{\pi}$*-vector* for each relevant graph.

Definition 6.7.1 $\hat{\pi}$-VECTOR. *The $\hat{\pi}$-vector of a graph g is a vector of $\hat{\pi}_{\theta_i}(g)$ computed at* *multiple distance thresholds θ_i, where $\hat{\pi}_{\theta_i}(g) \geq \pi_{\theta_i}(g)$ is an upper bound on the representative power of g computed using Theorem 6.6.3. Mathematically,*

$$\overrightarrow{\hat{\pi}(g)} = [\hat{\pi}_{\theta_1}(g), \cdots, \hat{\pi}_{\theta_n}(g)] \tag{6.13}$$

where $\theta_1 < \theta_2 < \cdots < \theta_{n-1} < \theta_n$ are n different thresholds.

The $\hat{\pi}$-vector allows us to compute an upper bound on the representative power for any given distance threshold θ. More specifically, a binary search can be performed on the indexed thresholds in the $\hat{\pi}$-vector to find the smallest $\theta_i \geq \theta$ and an upper bound $\hat{\pi}_{\theta_i}(g)$ can be derived to guide the searching process. We discuss which θs to index in Sec. 6.7.1.

Once the $\hat{\pi}$-vector is computed for all leaf-nodes (or graphs), the information is propagated upwards on NB-Tree by recursively computing and storing the ceiling of the $\hat{\pi}$-vectors of all children of a non-leaf node. Owing to this initialization procedure, an upper bound on the marginal gain in representative power of any graph g can be computed from any of its ancestor nodes. More specifically, for any non-leaf node n, and any set of graphs $\mathbb{S}$, the following property can be guaranteed.

$$\pi(\mathbb{S} \cup \{n\}) \geq \pi(\mathbb{S} \cup \{c\}) \text{ for any child } c \text{ of } n \tag{6.14}$$

Note that in the absence of interactive refinement, the $\hat{\pi}$-vector is not required since the input θ is already known; $\hat{\pi}_\theta(g)$ can be computed directly. However, if the threshold is refined to θ', then $\pi_{\theta'}(g)$ needs to be recomputed. More importantly, the information contained in any of the previously used θs cannot be leveraged. Thus, we pre-compute the representative powers at a series of distance thresholds in the initialization phase and use them as required for refinements. In other words, the initialization phase is insulated from refinements of θ.

Example 6.7.1 *Fig. 6.4 demonstrates how $\hat{\pi}$-vectors are constructed and propagated upwards in the tree through the ceiling operation. For simplicity, we assume all graphs are relevant.*

Algorithm 7 nextGraph($q(\cdot), \theta, k$)

```
1: PQ ← priority queue containing $NB\text{-}tree.root$
2: lb ← $0$
3: best ← **null**
4: **while** PQ is not empty **do**
5:    entry ← PQ.dequeue()
6:    **if** entry.gain<lb **then**
7:       **return** best
8:    **end if**
9:    **if** entry is a graph **then**
10:      Compute $\hat{N}_\theta(entry)$ using VOs
11:      $N_\theta(entry) \leftarrow \{g \mid g \in \hat{N}_\theta(entry),\ d(entry, g) \leq \theta\}$
12:      gain← $\pi(\mathbb{A} \cup entry) - \pi(\mathbb{A})$
13:      **if** gain> lb **then**
14:         best← entry
15:         lb← gain
16:      **end if**
17:   **else**
18:      **for** each child c ∈ entry **and** $q(entry) \geq \eta$ **do**
19:         gain← $\pi(\mathbb{A} \cup c) - \pi(\mathbb{A})$
20:         **if** gain> lb **then**
21:            PQ.insert(gain,c)
22:         **end if**
23:      **end for**
24:   **end if**
25: **end while**
26: **return** best
```

The $\hat{\pi}$-vector is computed on $\theta_1 = 1$ and $\theta_2 = 3$.

2. Search and Update: The second phase can be divided into two subcomponents: search and update, which are performed iteratively k times.

2a. Search: Alg. 7 presents the pseudocode. First, a search is performed to identify the graph providing the highest marginal gain in representative power. The search starts from the root node (line 1). When a node is explored (line 5), each of its children is added to a priority queue where the nodes are ordered based on their marginal gains in representative power (lines 15-18). Since the marginal gain at any non-leaf node provides an upper bound on the marginal gains of all graphs in its subtree, they are used to prioritize the exploration of the unexplored nodes. The process is repeated iteratively till a graph is found whose actual marginal gain is higher than all candidates in the priority queue (lines 6-7). Only for such a graph, its θ-neighborhood is computed using graph edit distance to compute the exact marginal gain in the representative power (lines 8-14). As a result, expensive edit distance computations are reserved for only strong candidates and the computational cost is drastically reduced.

2b. Update: The cluster based bounds identified in Sec. 6.6.3 drive the update step. At the addition of a graph g to the answer set, the $\hat{\pi}$-vectors of nodes in NB-Tree are impacted. Now, instead of accurate re-computations of $\hat{\pi}$-vectors, a search is initiated from the root to identify clusters (or non-leaf nodes) that are within a distance of 2θ from g and consequently, affected (Theorem 6.6.4). For such clusters, using Theorems 6.6.5-6.6.6, the $\hat{\pi}$-vectors are updated. To identify the relevant graphs in a cluster, only graphs in its subtree need to be scanned. As noted earlier, Theorems 6.6.4-6.6.6 provide the benefit of computing a single upper bound for a group of structurally similar graphs that are located in the same cluster. Finally, the ceilings of the changed $\hat{\pi}$-vectors are propagated upwards, as in the initialization step, to reflect the “current state” of the search algorithm.

To summarize, the initialization phase customizes the index for the query function. The ’Search and Update’ phase then takes over. The ‘Search’ procedure iteratively retrieves the graph with the highest representative power, and ‘Update’ computes its impact on the representative power of the remaining relevant graphs. In the entire query processing algorithm, expensive edit distance computations are performed only on strong candidates in the search step and with cluster centroids in the update step. All remaining neighborhood computations are performed using VOs and thus, the entire algorithm is computationally efficient.

Table 6.3: Graph datasets used for evaluation

Dataset	Avg. # of nodes	Avg. # of edges	# of graphs
DUD	26	28	128332
DBLP	55	263	11362
Amazon	29	189	9120

6.7.1 Choosing thresholds to index in the $\hat{\pi}$-vector

A bad choice of thresholds in the $\hat{\pi}$-vector would produce loose upper bounds which in turn would reduce the efficiency of the search-and-update procedure. Therefore, to make a more informed selection, the following factors need to be considered: domain knowledge, the history of previous queries, and the memory budget. Note that the selection of thresholds to index is an offline procedure and is decided by the index designer. The user who issues the query needs no prior information.

The number of thresholds to index can easily be decided based on the memory budget. For example, in an index with 1 million nodes, a $\hat{\pi}$-vector of size 10 would consume $\frac{4\times10\times10^6}{10^6}$ =40MB. For the more critical decision of choosing the thresholds, depending on the information and resources available, one of the following schemes can be adopted.

1. Query log: If the distribution of distance thresholds on previous top-k queries is available, then the thresholds to index can be sampled (without replacement) from that distribution.

2. No information is available: When no prior information is available, the distribution of $\pi(g)$ across θ can be computed from a small sample of graphs in the database. Next, the thresholds to index can be chosen proportional to the slopes in the distribution. More specifically, a higher number of thresholds should be indexed for the θ intervals where the slope is high since such intervals indicate regions where the upper bounds vary steeply even for a small difference in θ's.

Table 6.4: Compression ratios (CR) $|N_\theta(\mathbb{A})|/|\mathbb{A}|$ and $\pi(\mathbb{A})$ achieved by REP and DIV [7] at various ks. The last row indicates the compression ratio of DisC along with its answer set size in parentheses.

Property	**DUD**						**DBLP**						**Amazon**					
	REP		**DIV(θ)**		**DIV(2θ)**		**REP**		**DIV(θ)**		**DIV(2θ)**		**REP**		**DIV(θ)**		**DIV(2θ)**	
	CR	$\pi(\mathbb{A})$	CR	$\pi(\mathbb{A})$	CR	$\pi(\mathbb{A})$	CR	$\pi(\mathbb{A})$	CR	$\pi(\mathbb{A})$	CR	$\pi(\mathbb{A})$	CR	$\pi(\mathbb{A})$	CR	$\pi(\mathbb{A})$	CR	$\pi(\mathbb{A})$
$k = 10$	115	0.23	31	0.06	28.8	0.06	45.4	0.16	23.1	0.08	19.1	0.07	84.3	0.17	37.8	0.21	27	0.12
$k = 25$	80	0.40	22.9	0.11	16.4	0.08	25	0.22	12	0.11	10.7	0.09	41.9	0.46	19.6	0.22	15.9	0.17
$k = 50$	58	0.58	17	0.17	9.9	0.09	15.3	0.27	7.5	0.13	6.4	0.11	23.7	0.52	12	0.26	9.3	0.2
$k = 100$	35	0.70	10.1	0.2	4.7	0.09	11.6	0.41	4.7	0.17	4	0.14	15	0.66	6.9	0.3	5.4	0.24
DisC	2.77 (1782)						1.78 (1590)						2.5(912)					

6.8 Experiments

This section summarizes experimental results that examine the quality and the scalability of our approach:

1. Information Density: The proposed model has a higher information density than DisC [88] and a diversity-based model [7].

2: Scalability: By indexing θ-neighborhoods of graphs instead of their nearest neighbors, NB-Index ensures scalability in answering top-k representative queries on graph databases.

6.8.1 Datasets

We use three different graph datasets to benchmark our techniques. Table. 6.3 summarizes the various properties of the datasets.

1. DUD: The DUD [105] dataset contains 128,332 molecules that were assayed against 10 different protein targets. Consequently, each molecule is tagged with a 10-dimensional feature vector that represents its binding affinity to each of the targets. In the graph representation of a molecule, each vertex corresponds to an atom, and the edges represent atom-atom bonds. Our goal is to identify molecular structures that are compatible with the structure of the protein targets. The DUD dataset allows us to assess the natural correlations that exist between the feature and the structural space, and their resultant effects on top-k queries.

2. DBLP: In the DBLP network [101], each node corresponds to an author, and two authors are connected if they have at least one paper together. Furthermore, each author is tagged with the community he/she belongs too. For our evaluation, the goal is to understand if the most active groups collaborate within the community or span across multiple communities. For that purpose, we replace the author in each node label with the person's community. Next, we extract the complete 2-hop neighborhood subgraph around each node to construct our graph database. Finally, the combined activity level of each graph, denoting a collaboration group, is characterized with a one-dimensional feature vector.

3. Amazon: The Amazon network [101] is structured in a manner similar to DBLP. Each node corresponds to an item, and two items are connected if they are frequently co-purchased. In addition, each item is classified under a category. Our goal is to understand if cross-category coupling exists among items that are popular. Thus, similar to the processing in the DBLP dataset, we replace the node labels with the category of the item and then extract its 2-hop neighborhood subgraph. As in the DBLP dataset, a feature vector is used to characterize the popularity of each co-purchase graph.

6.8.2 Experimental setup

Our algorithms are implemented in Java 1.6.0 and benchmarked on an Intel i7 2.67GHz quad core processor PC with 12GB of main memory running Ubuntu 12.10. We refer to our proposed model as *REP*, and the diversity model in [7] is termed *DIV*.

1. Quality Evaluation: To evaluate the quality of the answer set produced REP, we compare its representative power with the full DisC answer set, and the top-k answer sets of DIV.

2. Scalability Evaluation: We benchmark the performance of NB-Index against the 'Grey-Greedy-DisC(Pruned)' algorithm of DisC [88], the 'div-cut' approach in DIV [7] and C-tree [103]. For DisC, we stop the computation as soon as it attains a size of k. For DIV, we use C-Tree to compute the 'diversity-graph', which is subsequently used by the 'div-cut' algorithm, to speed up processing.

Due to the extremely slow running times of DisC, DIV and C-tree on the entire DUD database, we select a random sample of $25,000$ graphs. Nonetheless, for scalability experiments against dataset size, we use the entire DUD database. For DBLP and Amazon, we use the entire dataset for all experiments.

Query Arguments

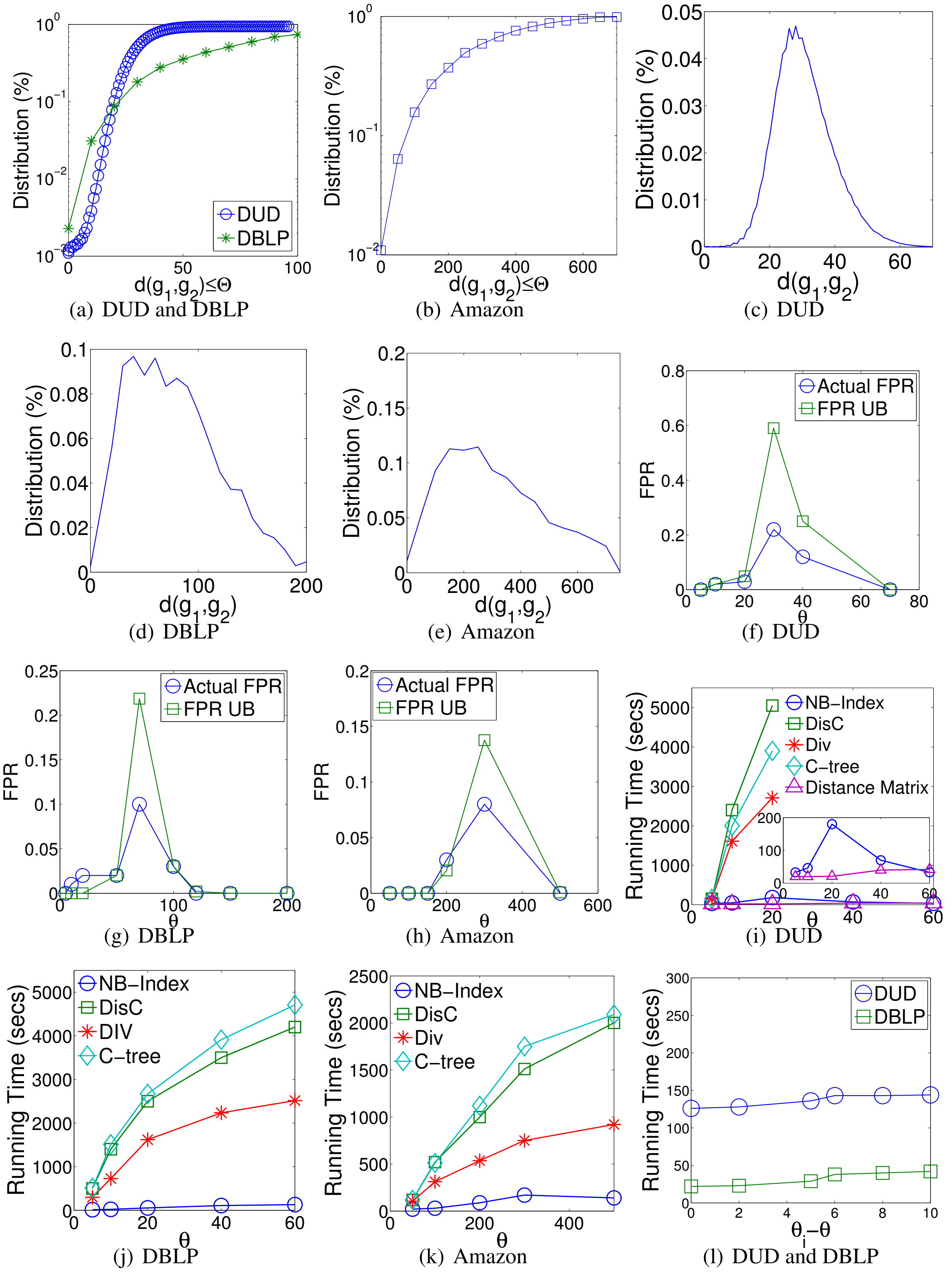
1. **$q(\cdot)$:** One primary goal in REP is to support query time definition of relevant graphs. DisC assumes that the set of relevant graphs is static and known apriori. To model this requirement, in DUD, we select a random subset of d dimensions where $1 \leq d \leq 10$, and the feature-space score is quantified as $\sum_1^d \frac{g_i}{d}$. Both the DBLP and Amazon datasets are characterized using 1D feature vectors, which act as their feature space scores. For all three datasets, a graph is considered relevant if its feature-space score falls within the first quartile. For example, a graph in the DBLP dataset is relevant if its activity level is within the top-25%.

2. **θ:** Our goal is to select a θ that is realistic and yet poses a significant scalability challenge in answering top-k representative queries. Towards that goal, we study the cumulative distribution of the graph distances in each dataset (Figs. 6.5(a)-6.5(b)). While the distributions are similar for DUD and DBLP, distances between graphs in the Amazon dataset are much farther apart. Based on these observations, for DUD and DBLP we set $\theta = 10$ and for Amazon, $\theta = 75$. For DBLP and DUD, addition of a graph to the answer set impacts the $\pi(g)$ of any graph g within a distance of $2\theta = 20$; for Amazon, graphs within a distance of 150 are affected. Thus, across all three datasets, each addition to the answer set affects one-tenth of the entire database on average, and for all these affected graphs, their representative power needs to be recomputed. Consequently, the chosen θs present a significant scalability challenge.

3. **k:** The default k is set to 10.

Parameters

1. **$\hat{\pi}$-vector:** The $\hat{\pi}$-vector is chosen based on the cumulative distance distributions in Figs. 6.5(a)-6.5(b). Given the high slope between $\theta = 10$ and $\theta = 40$ for both DUD and DBLP, we index 10 different distance thresholds at 5, 12, 16, 20, 25, 30, 35, 40, 75, 100. We index at θs beyond 40 just for the sake of completeness so that a query on any θ can be supported. For



(a) DUD and DBLP (b) Amazon (c) DUD (d) DBLP (e) Amazon (f) DUD (g) DBLP (h) Amazon (i) DUD (j) DBLP (k) Amazon (l) DUD and DBLP

Figure 6.5: (a-b). The cumulative distance distributions. (c-e) The distance distributions. (f-h) Observed FPRs, and FPR Upper Bounds (FPR UB) with θ. (i-k) Growth rates of query times against θ and (l) distance to the closest indexed threshold θ_i.

amazon, we adapt $\hat{\pi}$-vector using the same strategy. Owing to the high slope between $\theta = 50$ and $\theta = 200$, we index at 20, 60, 90, 120, 150, 180, 200, 300, 400, 500. We intentionally do not index at the default query θs of 10 and 75 to mirror a realistic scenario.

2. Number of VPs: The number of VPs is chosen based on Eqn. 6.11. The distance distributions in the three datasets (Figs. 6.5(c)-6.5(e)) are approximated as a Gaussian of their means and standard deviations. To limit the FPR below 5% at the realistic zone of $5 \leq \theta \leq 20$ for DUD and DBLP, and at $50 \leq \theta \leq 200$ for Amazon, we choose 100 VPs. We perform a detailed analysis on the performance of VPs in Sec. 6.8.3.

3. Branching Factor: The maximum branching factor in NB-Index is set to 40.

6.8.3 Quantitative analysis

In this section, we quantify the performance of the proposed techniques over the current state of the art.

Efficacy

One of our primary motivations in this work is to allow users to control the size of the answer set, so that a smaller set of exemplars can be identified and investigated in further detail. Towards that goal, we compare the *compression ratios (CR)* of REP, with the ones achieved by DisC and DIV. The CR of an answer set $\mathbb{A}$ is defined as $\frac{|N_\theta(\mathbb{A})|}{|\mathbb{A}|}$, which is simply the ratio of the number of relevant objects represented to the answer set size. In this experiment, for DUD, we consistently use the same subset of feature vector dimensions since it is essential to capture the natural correlations between the feature and structural space, which in turn affects the CR.

Table 6.4 presents the results. The CRs in REP are significantly higher than both DisC and DIV. This result establishes the utility of operating in a budgeted setting as well as the advantage of a representative-aware model over a diversity-aware model. For DIV, we compute the result at two different settings. In the first setting, DIV(θ), we use the original model in [7], which ensures all answer set objects are at least θ apart. However, this model assumes that the scores of graphs are mutually independent. In reality, the representative powers of the graphs are dependent on each other. Therefore, to also study the performance when the independence assumption is followed, we enforce the stricter condition of $\forall g_1, g_2,\ d(g_1, g_2) > 2\theta$. The produced answer set is denoted as DIV(2θ). As can be seen, DIV(2θ) further lowers the CRs since the stricter condition to guarantee score independence rules out many representative graphs from inclusion in the answer set.

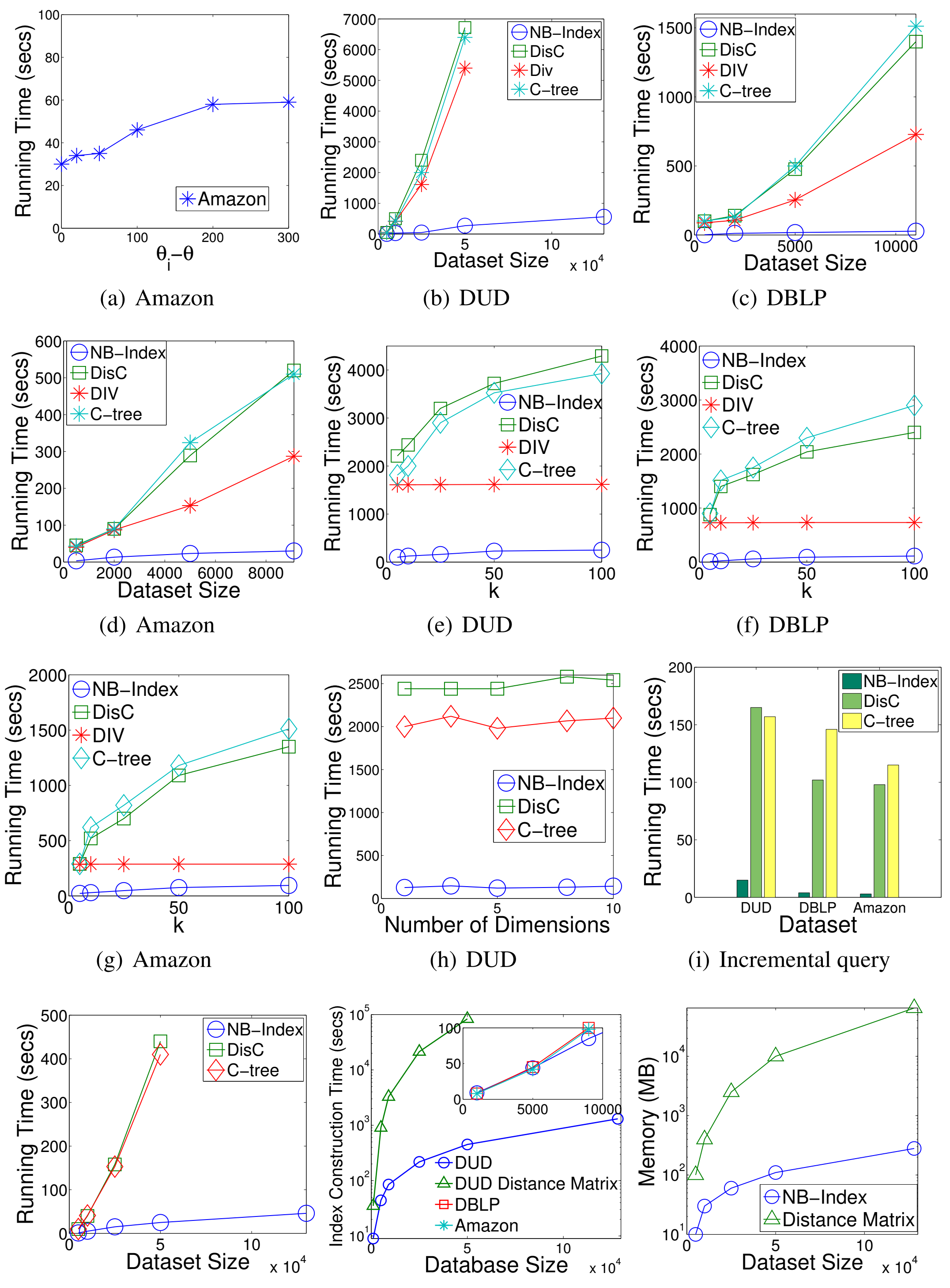
To further analyze the representation of relevant objects, we also compute the growth of $\pi(\mathbb{A})$ with k. As shown in Table 6.4, REP covers close to a quarter of the relevant objects using just 10 exemplars. As k increases beyond 100, it is futile to try to represent relevant objects using exemplars, since either the remaining non-represented objects are outliers or their θ-neighborhoods have large overlaps with already represented objects. Consistent with the results observed while analyzing the CRs, DIV(θ) and DIV(2θ) have up to 6 times lower representative powers due to relying on an indirect maximization approach. Overall, DIV is unable to model the semantics of representativeness, and consequently, the quality of the answer set is compromised.

A detailed theoretical analysis on the optimal number of VPs is performed in Sec 6.6.2 to predict an upper bound on the False Positive Rate (FPR). We now verify how well the empirical results conform to the theoretical analysis. Figs 6.5(f)-6.5(h) present the results. The FPR is highest in the DUD dataset, due to the low standard deviation of the distances. Intuitively, if all graphs are similar to each other and clustered together, the performance of VPs deteriorates. For this reason, on DBLP and Amazon, which are not as densely clustered, the FPR is much lower. At low θs in DBLP and DUD, the FPR Upper Bound (≈ 0.01) is slightly lower than the observed FPR. This results from the fact that the distance distribution in DBLP and Amazon slightly deviates from a normal distribution.

Scalability

First, we evaluate the performance of NB-Index against the distance threshold θ. Figs. 6.5(i)-6.5(k) show the growth rates of query times as θ is varied. NB-Index is up to two orders of magnitude faster than existing techniques. Due to the enormous running times of DisC, C-tree and DIV on DUD, we do not show their results beyond $\theta > 20$. The growth rates of existing techniques are similar since they all index nearest neighbor queries through triangular inequality; while DisC and C-tree propose their own index structures, in DIV, we use C-tree as the underlying graph indexing technique to construct the 'diversity-graph', on which DIV performs further processing. A bulk of the computation time is spent in initializing the θ-neighborhoods of relevant graphs. This operation needs to be performed online since both θ and the relevance function are query parameters. DIV assumes the representative powers of graphs to be mutually independent, and thus does not re-compute them after each answer set update. Consequently, at the cost of quality, it is faster than both DisC and C-tree. In NB-Index, majority of the computations happen in the feature space through VPs. As already shown in Figs. 6.5(f)-6.5(h), the FPRs in VPs are low. Furthermore, beyond VPs, the cluster based bounds outlined in Theorems 6.6.4-6.6.6 index the θ-neighborhoods in structural space itself and allows batch updates of representative power.

To further evaluate our performance, we also compare its running time to the scenario where the distance matrix of the entire graph database is pre-computed. Due to the fast running times of both the approaches, the inset in Fig. 6.5(i) presents a zoomed-in view to better compare the two approaches. As discussed in Sec. 6.5, pre-computing the distance matrix is not advisable since it incurs a huge storage cost. Nonetheless, we benchmark our performance against the best-case scenario from the running time perspective. Except at $\theta = 20$, the performance of NB-Index is 1.5 to 2 times higher than storing the distance matrix. Another important property that is emphasized in the inset of Fig. 6.5(i) is that NB-Index is most efficient at the



(a) Amazon (b) DUD (c) DBLP

(d) Amazon (e) DUD (f) DBLP

(g) Amazon (h) DUD (i) Incremental query

(j) Incremental query (k) Index construction (l) Index size

Figure 6.6: Growth rate of query times with (a) difference between user-provided distance threshold θ and closest indexed threshold θ_i, (b-d) dataset size, (e-g) k and (h) number of dimensions. (i) Running times for incremental zoom-in and zoom-out. (j) Growth rate of incremental zoom-in and zoom-out query time with dataset size. Growth rate of (k) index construction time and (l) indexing memory footprint against dataset size.

two extreme ends of the θ range. When θ is large, Theorems 6.6.5 and 6.6.6 are more effective in pruning the search space. On the other hand, Theorem 6.6.4 is more efficient at low θs. Thus, mid-range θs present the most challenging scenario, which results in a bell shaped growth rate of the query time.

Next, we analyze the performance deterioration as the difference between the user-provided distance threshold θ and the closest higher indexed threshold θ_i in NB-Tree increases. This analysis shows how sparsity in the $\hat{\pi}$-vector affects the performance of NB-Index. Note that a non-indexed θ only affects the performance of the structural bounds. The performance of VOs remain unaffected. Figs. 6.5(l), 6.6(a) show the growth rates of running times. Even when the difference is as high as 10 for DUD and DBLP, and 300 for Amazon, NB-Index consumes less than 25 seconds of additional time. As evident from Figs. 6.5(a)-6.5(b), the slopes are extremely steep between $10 \leq \theta \leq 20$ for DUD and DBLP and $50 \leq \theta \leq 200$ for Amazon. Thus, upper bounding π_{10} with π_{20} (in DUD and DBLP) provides a particularly tough setting. However, even in such cases, the performance of NB-Index is significantly better thanks to the efficiency of the VOs and the fast upper bounds derived using Theorems 6.6.5 and 6.6.6.

Figs. 6.6(b)-6.6(d) demonstrate the growth rate of running time against dataset size. For DUD, we vary the size from $5,000$ to the entire database containing $128,332$ graphs. NB-Index scales significantly better than all three existing techniques and achieves a performance that is more than an order of magnitude faster. This result stems from indexing the θ-neighborhoods of graphs, which negates the need to perform $O(n^2)$ nearest neighbor queries. Figs. 6.6(e)-6.6(g) further establish the superiority of NB-Index as k is varied from 5 to 100. The growth rate of query time with k is much lower for NB-Index due to the same reason of indexing the θ-neighborhoods of graphs. The running time of DIV is almost constant across k since it assumes the representative powers of graphs to be mutually independent. Thus, after the diversity-graph is constructed, all remaining computations occur in the feature space, which is minuscule when compared to the cost of constructing the diversity-graph.

Fig. 6.6(h) investigates the performance as the number of dimensions in feature vectors is varied between 1 and 10 in the DUD dataset. A feature vector of dimension d is constructed by randomly choosing a subset of d dimensions from the overall 10. The query time is almost identical across different numbers of dimensions for all three techniques since the cost of operating in the feature space is negligible compared to the cost of computing graph edit distances while updating θ-neighborhoods. The minor variation in running times results from the correlations between feature and structural space where a higher correlation results in faster query times.

Fig. 6.6(i) compares query times in the interactive θ refinement scenario. DIV is not included in this experiment since it assumes θ to be an offline parameter. In this experiment, first, a query is performed on the default θ outlined in Sec. 6.8.2. Next, a new θ is selected, which is either 10% smaller or larger, and the answer set is re-computed. This process is repeated 20 times and the average computation time is reported. NB-Index tackles refinements of θs within 10 seconds across all three datasets. On the other hand, although a lot faster than a new query, DisC and C-tree takes up to 160 seconds to adapt to a new θ. To further study the response times in an interactive setting, we analyze its growth rate against dataset size in Fig. 6.6(j). We only use the DUD dataset since for the other two datasets, a query refinement can be answered within 5 seconds. Similar to the previous scalability results against dataset size, NB-Index is more than an order of magnitude faster.

Finally, we analyze the computation and storage costs of the proposed index structure. Fig. 6.6(k) presents the growth rate of the index construction time against dataset size. While the main plot compares the construction time of NB-Index with that of computing the entire distance matrix in the DUD database, the inset shows the construction times across all three datasets. In DUD, which contains a total of $\approx 130,000$ graphs, the index construction completes within 20 minutes. Compared to the cost of computing the entire distance matrix, we are more than two orders of magnitude faster. This efficiency is achieved through VP-based

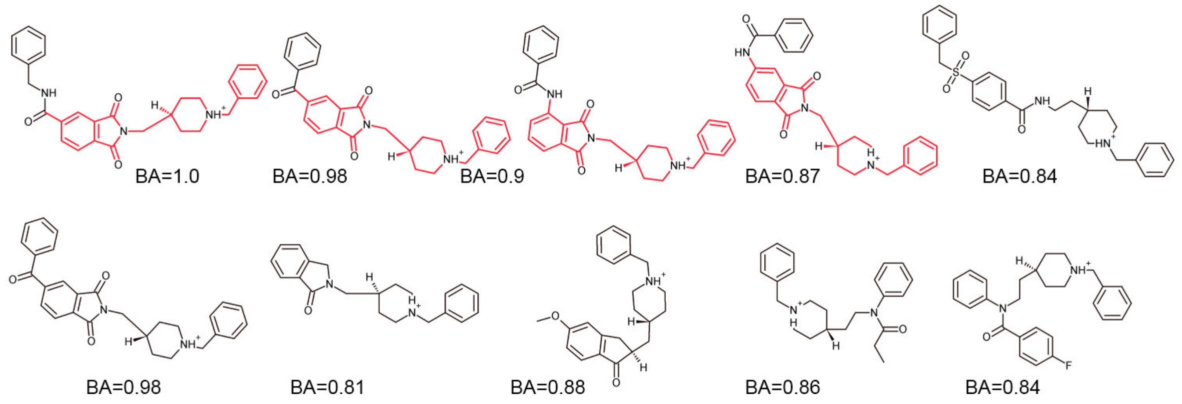


Figure 6.7: Answer sets computed using traditional top-k query (first row) and top-k representative query (second row). In the top row, the subgraph in red occurs in four of the five molecules, indicating the core structure of a single class of molecules.

pruning in pivot and radius computations where actual edit distances are computed for less than 1% of the candidate pairs. The scalability of NB-Index also extends to its storage cost. As expected from the theoretical analysis, a linear growth rate is observed in Fig. 6.6(l) and less than 300MB is required to store the index for the entire DUD dataset. The reported result includes the memory consumption from storing the $\hat{\pi}$-vectors, which are computed at query-time.

6.8.4 Qualitative analysis

In this section, we highlight the utility of a representative model over the traditional top-k framework. Towards that goal, we perform a top-k query and a top-k representative query using the same scoring function and compare the two answer sets. To construct the database, we extract molecules from the DUD repository that are active against the enzyme Acetylcholine ezterase (AChE). Extracted molecules are characterized based on their binding affinity to AChE, which is one of the primary targets for drugs treating Alzheimer's disease [106]. A molecule is considered relevant, if its binding affinity, denoted as BA, is within the top quartile.

Fig. 6.8.4 shows the two top-5 answer sets. As illustrated in Fig. 6.8.4 using color coding, all molecules in the traditional answer set are structurally similar to each other. Although they exhibit high binding affinity toward AChE, the traditional answer set represents just one structural family of high scoring molecules. On the other hand, the representative answer set contains molecules that are all structurally diverse, high scoring, and representative of other high-scoring molecules in the dataset. The efficacy of the proposed model is well highlighted in the shown result where the entire class of molecules in the traditional answer set is summarized by the top molecule in the representative answer set. In the context of drug discovery, the traditional answer set provides only one class of molecules that can be further optimized to develop drugs. Whereas, the representative answer set provides five different classes of molecules, each of which can be studied further for drug discovery.

6.9 Related Work

A large body of work exists on result-set diversification [89–95, 110]. A detailed comparison with DisC [88] and [7] has already been performed. A closely related query, called top-k typicality queries is formulated in [111,112]. An object o is considered "typical" if its distances to other objects in the database are small. More formally, the probability density of o is used as its typicality score. While the typicality score is conceptually similar to the proposed idea of representative power, the paper does not explore the coverage maximization aspect. Consequently, the typicality scores of two objects are independent and there is no penalty when highly typical objects from the same cluster are included in the answer set. As a result, the problem in [111] is not NP-hard, whereas ours is. In addition, instead of high-dimensional objects, we focus on graphs where computing the distance itself is NP-hard. Consequently, the scalability challenges faced are unique to our problem.

The idea of representativeness has been explored in the context of skyline queries [113]. An object in a database is a skyline point if it is not *dominated* by any of the database objects. The representativeness of a skyline point is the number of point it dominates. The goal in this work is to identify the k skyline points that dominates the maximum number of database points. Thus, while in our work, an object represents another object if their distance is within a threshold, in [113], an object o_1 represents object o_2 if o_1 dominates o_2. Consequently, the modeling requirements and the proposed indexing techniques are different. In addition, we focus on graphs whereas [113] focuses on high-dimensional points.

Existing work on diversification of graph-based searches [94, 95, 110] are limited to single large networks. In the single graph setting, an answer set constitutes of graph nodes, whereas in our problem, the answer set contains actual graphs. Due to this fundamental difference in problem formulation, existing diversification techniques on graphs cannot be applied to our problem.

Beyond diversification of results, indexing graphs for similarity queries [103,104,114,115] is related to our problem. As shown in our experimental results, indexing graphs for nearest neighbor queries is not enough to scale top-k representative queries. Consequently, our work takes a more direct approach by indexing the θ-neighborhoods of graphs, which lies at the core of updating graph representative powers. Embedding graphs into a feature space has been explored by Zou et. al. [116]. Their goal is to index shortest part queries between nodes by embedding nodes in a feature space. In our work, instead of a single large graph, we have multiple graphs each of which is embedded in a feature space. Additionally, instead of indexing shortest paths, our goal is to index their θ-neighborhoods, which is a function of graph edit distance. Due to this basic difference in the entities being embedded and their associated indexing goals, the technique, the bounds derived from the techniques, and the error guarantees do not transfer.

6.10 Conclusion

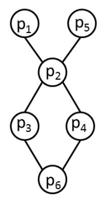
In this work, we formulated the problem of top-k representative queries on graph databases. By allowing online definitions of object relevance and answer set budget, the proposed formulation provides a higher degree of flexibility to users. We showed that the problem is NP-hard and submodular. Based on this result, we designed a greedy constant factor approximation of the optimal answer set. To achieve scalability, we designed an index structure called NB-Index that indexes the θ-neighborhoods of database graphs by employing a novel combination of Lipschitz embedding and agglomerative clustering. NB-Index not only facilitates fast answering of queries, but also allows interactive refinement of θ to reach the optimal zoom level. Empirical evaluations on real graph datasets demonstrate significantly higher information density and a performance improvement of up to two orders of magnitude over the state of the art.

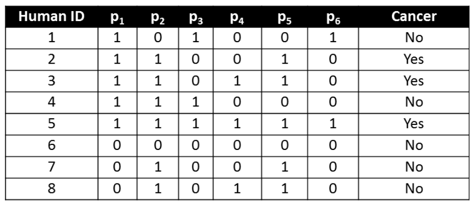
Chapter 7

Mining Discriminative Subgraphs from Global-state Networks

7.1 Introduction

The ability to capture multiple snapshots of a network leads to a "global state" network in which the snapshots share the same structure but have different values on nodes and/or edges. Furthermore, the network-guided evolution of the local states jointly determines the global network state in each snapshot. Such global-state networks (GS-network) can model a myriad of domain specific features such as traffic congestion in transportation networks [117], evolution of opinions and sentiments on social networks [118], gene expression levels on protein-protein interaction networks [119] and scaffolds in molecular libraries [120]. For example, in protein-protein interaction networks, the expression levels of individual proteins encode logical functions that determine the presence or absence of a disease. In social networks, opinion expressed on a movie by a certain user affects the opinions of his/her friends which in turn sets off a word-of-the-mouth cascade that ultimately decides the global consensus. *How do local node labels govern the evolution of the global network state? Can we save cost by monitoring*





Human ID	p_1	p_2	p_3	p_4	p_5	p_6	Cancer
1	1	0	1	0	0	1	No
2	1	1	0	0	1	0	Yes
3	1	1	0	1	1	0	Yes
4	1	1	1	0	0	0	No
5	1	1	1	1	1	1	Yes
6	0	0	0	0	0	0	No
7	0	1	0	0	1	0	No
8	0	1	0	1	1	0	No

Local and Global states of the network

Figure 7.1: A GS-network based modeling of protein-protein interaction data on eight different humans (snapshots). The GS-network models the occurrence of cancer. A protein expression level of 1 denotes abnormal activity, whereas 0 indicates normal expression levels.

only a discriminative subgraph and still be able to predict the global network state accurately? In this paper, we investigate these questions.

Consider the problem of inferring biological outcomes from the human protein-protein interaction network (*PPI*). A hypothetical example is shown in Fig.7.1. In a PPI, each node corresponds to a protein and two proteins are connected by an edge if they are known to interact while regulating a common biological process. As a result, abnormality in the expression level of a certain protein directly impacts only its neighbors. As evident in Fig.7.1, the expression level of a protein varies from person to person. Research in systems biology has shown that clinical outcomes, such as susceptibility to cancer, depend not only on the expression level of a single protein, but on pathways or network modules [119]. Modeling this phenomenon, therefore, requires us to have a network with dynamic node labels and a global dynamic state; the node labels indicate the protein expression levels in a human and the global state indicates the presence or absence of the disease. To predict the biological outcome, we need to find the sub-networks whose local states accurately predict the global network state.

As illustrated above, GS-networks can model aspects of data that are beyond the scope of static networks. A line of work that closely resembles GS-networks is the idea of time-evolving dynamic networks. A dynamic network consists of a series of networks whose properties change with time. However, dynamic networks lack a global network state and existing techniques on analyzing dynamic networks primarily focus on studying time-evolving recurrent patterns [117, 121, 122]. In contrast, the goal of our problem is to learn the network-encoded logic functions from the local states, and then predict the global network state. For that purpose, we develop a technique called *MINDS (MINing Discriminative Subgraphs)* to mine discriminative subgraphs from large GS-networks.

The problem of mining discriminative subgraphs has been studied. However, all existing techniques [123–126] target small graph databases such as chemical compound repositories. Consequently, the following question arises: *Are their challenges that are unique to GS-networks?* The answer to the question lies in the basic fundamentals of the processes being modeled. In contrast to GS-networks, small graph databases contain multiple graphs, each of which models a database object (such as a chemical compound) and is associated with a class label. A subgraph is considered discriminative if it is statistically "over-represented" in graphs belonging to any one of the classes. Consequently, existing techniques focus on minimizing the cost of subgraph isomorphism and efficient computation of subgraph frequencies. On the other hand, a GS-network models the evolution of a process through a network and its impact on the local network entities as well as the global network state. Thus, the focus is on mining discriminative subgraphs whose local states jointly encode the global network states.

Learning discriminative subgraphs from GS-networks is key to understanding the complex relationship that exists between the local and the global states. Consider the problem of monitoring environmental sensor networks and learning regression models to predict the intensity of climatic conditions. In an environmental network, each sensor represents a node and measures environmental properties such as pressure, temperature, etc. Two sensors are connected if changes in environmental factors directly influence each other. Now, research in meteorological science has established that climatic conditions in a region depend not only on local factors, but also on environmental conditions across the globe. For example, the intensity of Indian monsoon is linked to El Niño [127]. While limited success has been achieved in making short-term forecasts based on local environmental factors, long-term forecasts based on global factors remain a challenge. Mining discriminative subgraphs from environmental sensor networks would help us identify such global factors and forecast onsets of extreme conditions to minimize the resulting damage.

While optimizing prediction quality is important, it is also essential to learn local models that are consistent with the network structure. Additionally, the mined sub-networks should regularize the GS-network by applying a bias of network constraint towards shrinking the hypotheses space. Compactness of discriminative sub-networks is key to both network regularization as well as real time monitoring. Consider the scenario in sentiment analysis on social networks to predict stock market momentum. [118]. The global behavior of users in the network is shaped through their individual opinions and the resulting cascading effects within their social circles. Given the scale of social networks such as Facebook or Twitter, monitoring the entire user base to provide real-time updates on the global consensus is not feasible. Mining the most compact discriminative subgraphs promises to penetrate this scalability bottleneck by identifying smaller groups of influential users that maximally predict the global behavior.

Clearly, mining discriminative subgraphs from GS-networks is a powerful mechanism for identifying network components that are influential in determining the global state. However, given the fact that decades of research has already been performed on learning classification models, an obvious question arises: *How is the problem of mining discriminative subgraphs different from training classifiers?* To answer this question, we highlight the key aspects of our problem that are beyond the scope of a traditional classifier. **1. Semantics:** Learn local prediction models that are sensitive to the underlying network structure. In our problem, each feature (or node) is constrained within a structure and the network event being modeled evolves through that structure. On the contrary, a traditional classifier operates on unstructured data where each feature represents an axis in a high-dimensional space. Consequently, any model learned lacks semantic meaning.

2. Level of abstraction: Mine discriminative subgraphs, each of which is self-sufficient in explaining the evolution of the global network state and modeling a coherent event. For example, in PPI, such a subgraph corresponds to a biological process, whereas in environmental networks, a subnetwork represents a region. The local models can further be combined to design ensemble learning algorithms. On the other hand, a traditional classifier is only capable of mining discriminative nodes with the sole focus on prediction quality. Consequently, the learned patterns are of a low-level and do not capture the higher-level structure.

3. Beyond Classification: Mine discriminative subgraphs that not only provide the platform for learning classification models, but also network regularization, regression and monitoring.

To achieve the properties highlighted above, we design a technique called *MINDS* to mine *minimally discriminative* subgraphs from large GS-networks without compromising the underlying network structure. To summarize:

• We formulate the problem of mining minimally discriminative subgraphs from large GS-networks. To learn local prediction models and quantify the discriminative potential of a subgraph, we introduce the concept of *network-constrained decision tree* that learns *network-encoded logic functions* to predict the global network state.

• To tackle the exponential subgraph search space, we formulate the idea of an *Edit Map*, on which we perform Metropolis-Hastings sampling algorithm and leverage the *memoryless* property of Markov chains to drastically reduce the computation cost.

• We perform extensive experiments on real GS-networks to evaluate the efficiency and effectiveness of MINDS. Our results show that the proposed algorithm achieves an accurate approximation of the optimal answer set. Furthermore, MINDS outperforms the current state-of-the-art classifiers developed for PPIs.

7.2 Problem Formulation

A network/graph $G = (V, E)$ is composed of a set of nodes $V = \{v_1, v_2, \cdots, v_n\}$ modeling the entities of the network and a set of edges $E = \{(v_i, v_j) \mid v_i, v_j \in V\}$ modeling the relationships between these entities. A *network snapshot* $N = (V, E, L, S)$ contains two additional parameters: a labeling function $L : V \to \mathbb{R}$ and the global network state S. While L operates on the node IDs and models the local states, the global state function S quantifies the success of the event being modeled. For simplicity, we assume edges to be *undirected* and $S \in \{-1, 1\}$. However, all of the theory developed in this paper is easily generalizable to variants such as edge-weighted graphs, directed edges, multi-class states, or continuous valued states.

Definition 7.2.1 GLOBAL-STATE NETWORK: *A GS-network is a set of network snapshots $\mathbb{N} = \{N_1, \cdots, N_n \mid N_i = (V_i, E_i, L_i, S_i)\}$. We alternatively use the notation $\mathbb{N} = (V_N, E_N, L_i, S_i)$ to denote a GS-network where $V_N = \bigcup_{\forall N_i \in \mathbb{N}} V_i$ and $E_N = \bigcup_{\forall N_i \in \mathbb{N}} E_i$.*

Example 7.2.1 *Fig.7.1 demonstrates a hypothesized GS-network modeling the occurrence of cancer. The global state encodes the presence or absence of cancer and the local states indicate the protein expression levels. All snapshots in this GS-network share the same structure. For snapshots with different structures, the null value is used to denote the state of a missing node.*

A graph $G = (V, E)$ is a subgraph of a GS-network $\mathbb{N} = (V_N, E_N, L_i, S_i)$, denoted by $G \subseteq \mathbb{N}$, if $V \subseteq V_N$ and $E \subseteq E_N$. A stronger constraint is enforced by the relationship of *induced* subgraphs.

Definition 7.2.2 INDUCED SUBGRAPH: *$G = (V_G, E_G)$ is an induced subgraph of GS-network $\mathbb{N} = (V_N, E_N, L_i, S_i)$, denoted as $G \subseteq \mathbb{N}$, if and only if $V_G \subseteq V_N$, $E_G \subseteq E_N$, and $\forall (u, v) \in E_G$ where $u \in V_G$ and $v \in V_G$, $(u, v) \in E_N$.*

A *supergraph* is defined analogously.

In this paper, we focus on mining only connected induced discriminative subgraphs of a GS-network. Consequently, any reference to a subgraph is assumed to be a connected induced subgraph.

Given a training dataset, our goal is to mine subgraphs that accurately predict the global state S of any snapshot $N \in \mathbb{N}$. Furthermore, the mined subgraphs should be as compact as possible to ensure network regularization. Towards that goal, we first define the notion of *discriminative subgraphs*.

Definition 7.2.3 DISCRIMINATIVE SUBGRAPHS: *Given a GS-network* $\mathbb{N} = (V_N, E_N, L_i, S_i)$*, let* $f(G, L)$ *be a structure-sensitive prediction function that predicts the global state of a network. If* $\mathbb{C} = \{N_i = (V_N, E_N, L_i, S_i) | N_i \in \mathbb{N}, f(G, L_i) = S_i\}$ *is the set of correctly predicted networks, then the discriminative potential of subgraph* $G \subseteq \mathbb{N}$ *is:*

$$\phi(G) = \frac{|\mathbb{C}|}{|\mathbb{N}|} \tag{7.1}$$

G is discriminative if $\phi(G) \geq \theta$ for a user-provided threshold θ.

Due to our assumption of binary valued global states, $f(G, L)$ is essentially a classification model. For continuous valued global states, $f(G, L)$ would be a regression function. We elaborate on how to learn the prediction function $f(G, L)$ in Secs. 7.3 and 7.4.

While one could mine all discriminative subgraphs in the network for a given threshold θ, such an answer set is likely to be informationally sparse. More specifically, given a subgraph G that is discriminative, all of G's supergraphs are discriminative as well. This result follows from the fact that any prediction function $f(G, L)$ learned from $G = (V_G, E_G)$ can be learned from a supergraph $G' = (V_{G'}, E_{G'}) \supseteq G$ as well since the feature set $V_{G'} \supseteq V_G$ contains all the information embedded in G. Therefore , to mitigate this potential issue of information sparsity, our goal is to extract the set of *minimally discriminative* subgraphs.

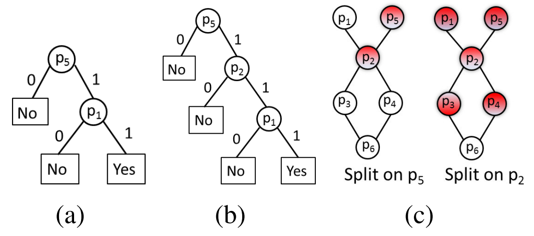


Figure 7.2: The optimal traditional decision tree (a) and the optimal network-constrained decision tree (b) for the GS-network shown in Fig.7.1. (c) Demonstrates how infection spreads in the network. The infected nodes are highlighted in red.

Definition 7.2.4 MINIMALLY DISCRIMINATIVE SUBGRAPHS: *A subgraph G is minimally discriminative if $\phi(G) \geq \theta$ and the set $\{G'|G' \subseteq G, \phi(G') \geq \phi(G)\} = \emptyset$.*

As can be seen, minimally discriminative subgraphs correspond to the smallest possible subnetworks within a GS-network that are influential enough to determine its global state. Consequently, mining minimally discriminative subgraphs allows us to maximize the information density in the answer set and avoid overfitting.

7.3 Network-constrained decision trees

Sec. 7.2 formalizes the discriminative potential of any graph G. However, we still need to learn a structure-sensitive prediction function $f(G, L)$ so that $\phi(G)$ can be quantified. From Defn. 7.2.3, $\phi(G)$ is directly proportional to the probability $P(f(G, L_i) = S_i)$ for any network $N_i \in \mathbb{N}$. Without the constraints of the structure, the problem is essentially that of learning a classification/regression model on the GS-network $\mathbb{N}$ using only nodes in G as features. However, as already discussed in Sec. 7.1, such an approach lacks semantic meaning and the level of abstraction required to gain meaningful insights from the mined network features.

To concretize the importance of structure in our problem further, consider the hypothesized GS-network in Fig.7.1 and the local events where protein p_1 over-expresses (i.e., samples where p_1 has node label 1). From the network structure, it is evident that an abnormality in p_1

has a direct impact only on p_2. As a result, out of the five human samples where $p_1 = 1$, p_2 behaves abnormally on four of them. Now, through p_2, the abnormality in p_1 has a cascading effect on the expression levels of p_3, p_4 and p_5. A deeper analysis of the example reveals that whenever both p_1 and p_5 behave abnormally, the corresponding human samples are susceptible to cancer. Clearly, p_1 and p_5 are statistically the most informative nodes and this fact is reflected in Fig.7.2(a) which shows the optimal decision tree (DT) for Fig.7.1. Notice that the learned model is completely oblivious to the fact that the process evolved from p_1 to p_5 through p_2. Even though p_2 is not statistically informative, structurally, it is the "bridge" between p_1 and p_5, and thus, plays a key role in determining the global network state. From a biological viewpoint, if p_2 can somehow be shielded from abnormal behaviors in p_1, then the risk of the disease is greatly diminished. Clearly, the importance of p_2 must be recognized and the failure of traditional local learners in capturing this key structural aspect highlights their limitations in mining structured data such as graphs. To capture the importance of network structure within the framework of a traditional classifier, we introduce the concept of *network-constrained decision trees (NCDT).*

Similar to the goals of a DT, an NCDT also learns the optimal boolean function that best predicts the global states using the local node labels. However, an NCDT also models the evolution of a process through the network and imposes additional constraints on nodes that can be used to split the training dataset. For the first split, an NCDT is free to choose any node. After the first split node n_1 is selected, an NCDT considers n_1 as "infected". Furthermore, the infection spreads from n_1 to all of n_1's neighbors, and an NCDT can select only one of the infected nodes to decide the next split. Based on this constraint, once the second split node n_2 is selected, n_2's neighbors in turn get affected and this process repeats recursively, like in a DT, till leaf nodes are reached. The additional constraint of splitting only through infected nodes ensures that "structural bridges" are captured and we do not overfit the learned models. Note that the proposed NCDT can easily be employed for learning a regression function as well by incorporating the same strategies used for learning regression DTs. Formally, an NCDT is defined as the following:

Definition 7.3.1 NETWORK-CONSTRAINED DECISION TREES: *A decision tree is also an NCDT if all nodes in the tree form a connected component in the GS-network.*

Example 7.3.1 *Fig.7.2(b) shows the optimal NCDT for the GS-network in Fig.7.1. Fig.7.2(c) demonstrates how the "infection" spreads in the network. In the optimal NCDT,* p_5 *is selected as the first split node. As a result,* p_5 *and* p_2 *get infected. Among the infected nodes,* p_2 *is selected for the next split, which results in the infection spreading to* p_1*,* p_3 *and* p_4*.* p_1 *is selected for the third split to produce the optimal NCDT. On the other hand, the DT in 7.2(a) is not an NCDT since* p_5 *and* p_1 *do not form a connected component.*

Certainly, DT is not the only classifier than can be adapted to learn a structure sensitive prediction function. We choose NCDTs since it forms a natural and intuitive extension to DT. Additionally, as shown later in Sec. 8.6, the linear construction cost of NCDTs make it highly efficient when compared to other state-of-the-art classification techniques.

7.3.1 Computational Challenges

With the formalization of NCDTs, we now have a mechanism to quantify the discriminative potential of any graph. In this section, we analyze the computational challenges faced while mining minimally discriminative subgraphs.

Claim 2 *Computing the optimal NCDT is NP-hard.*

PROOF: Learning the optimal DT is known to be NP-complete [128]. Given any dataset $\mathbb{D} = \{d_1, \cdots, d_n\}$ where $d_i = (x_1, \cdots,$
$x_k, y_i)$ with y_i being the class label, the decision tree problem is to determine whether there exists a decision tree of size (i.e., number of nodes in the tree) less than s that classifies each

d_i correctly. Given an arbitrary instance of the problem, we construct a clique with all features (or nodes) $1, \cdots, k$ connected to each other. It is easy to see that a DT of size less than s exists if and only if an NCDT of size less than s exists. In other words, learning an NCDT on a clique is equivalent to learning a DT. $\square$

NP-hardness of computing the optimal NCDT is not the only computational challenge. To mine minimally discriminative subgraphs, we need to first enumerate all possible subgraphs of the GS-network, and then compute their discriminative potentials. Unfortunately, the number of subgraphs in a network grows exponentially with its size and as a result, enumerating all possible subgraphs is not feasible. Consequently, the proposed problem presents us with a unique challenge: *how can we mine minimally discriminative subgraphs even without enumerating the entire search space?*

7.4 Mining Discriminative Subgraphs

Sec. 7.3.1 outlines the two computational challenges in mining discriminative subgraphs from large GS-networks. In this section, we address these two challenges. First, we devise a strategy to compute NCDTs *greedily*. Next, to combat the exponential search space, we impart an ordering on the candidate subgraphs in the form of an *edit map*, and then perform *Metropolis-Hastings* [129] sampling on the map to compute an accurate approximation.

7.4.1 Greedy computation of NCDT

As in greedy learning of traditional DTs, the first node to split the training dataset is selected greedily by choosing the one with the highest statistical importance. The statistical importance can be quantified using any of the existing attribute value tests such as *information gain* or *gini index*. For our implementation, we use information gain which is defined as the following

$$IG(\mathbb{N}, u) = E(\mathbb{N}) - \sum_{l \in L^*(u)} \frac{|\mathbb{N}_l|}{|\mathbb{N}|} E(\mathbb{N}_l) \tag{7.2}$$

where $\mathbb{N}$ is a GS-network, $L^*(u)$ is the set of all possible labels for node u, $\mathbb{N}_l = \{N = (V_N, E_N, L, S) | N \in \mathbb{N}, L(u) = l\}$ is the set of networks where node u has label l and $E(\cdot)$ is the *entropy* of a set. The first split divides $\mathbb{N}$ into $|L^*(u)|$ subsets. Next, the set of infected nodes is computed, and each of the subsets is split recursively by choosing the infected node with the highest information gain for that subset. As in a DT, this process completes when leaf nodes are reached where either the global states of all snapshots belong to a single class or no feature exists to split further.

7.4.2 Searching greedily in the subgraph space

A greedy learning of the NCDT tackles the NP-hardness challenge outlined earlier. Now, we focus on the second challenge of exploring the exponential search space,, which cannot be computed or stored due to its sheer size. Our current capabilities only allow us to compute the discriminative potential of a given subgraph and evaluate local modifications to further improve its discriminative power. Thus, our only hope to reach the globally optimal solution is through locally optimal choices. Towards that goal, a greedy approach could be adopted. First, the node with the highest information gain can be identified as the seed, and an NCDT can be constructed greedily around that seed node. The corresponding subgraph would therefore be the nodes spanning the NCDT. If the subgraph is discriminative, then it is added to the candidate set, otherwise discarded. To continue populating the candidate subgraph set, the process is restarted from the seed node with the second highest information gain. Once all nodes have been explored, the answer set can be computed by identifying only the minimally discriminative subgraphs from the candidate set.

While a greedy strategy is computationally efficient, the discriminative potential of the

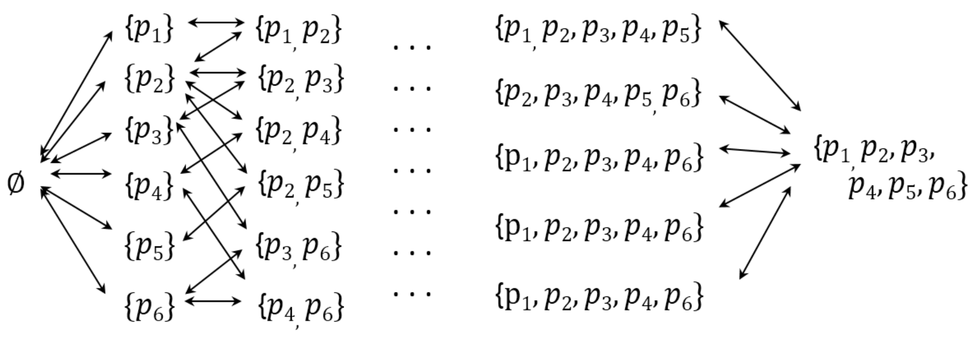


Figure 7.3: The top three levels and the bottom two levels of the edit map of the GS-network in Fig.7.1. We just show the set of nodes in each vertex of the edit map since the edges can be concluded from the definition of an induced subgraph.

subgraph is highly restricted by the choice of the initial seed node. If the seed lies in a neighborhood where the rest of the nodes provide low information gain, then the resultant subgraph will be non-discriminative as well. More importantly, a greedy algorithm is constrained to continue expanding the NCDT in the low informative region even after realizing that the initial seed is an informative outlier. What is therefore critical to the success of any local optimization based approach is being sensitive to back-tracking and negating any of the wrong choices already made. MINDS builds upon this intuition by converting the subgraph search space into an *Edit Map*, and then performing MH sampling on the map to mine minimally discriminative subgraphs.

7.4.3 Edit Map

The edit map (*EM*) of a GS-network represents all possible *edits* that can be performed on any subgraph $G \subseteq \mathbb{N}$ in the form of an edge-weighted *partial-order* graph.

Definition 7.4.1 EDIT MAP: *The edit map of a GS-network* $\mathbb{N} = (V_N, E_N, L_i, S_i)$ *is a directed edge-weighted graph* $M = (V_M, E_M)$*, where* $V_M = \{G | G \subseteq \mathbb{N}\}$*,* $E_M = \{(G = (V, E), G' = (V', E')) \mid \textit{either } G' \supseteq G,\ V' = V \cup \{u\}, u \notin V,\ u \in V_N \textit{ or } G' \subseteq G,\ V' = V \backslash \{u\}, u \in V\}$*, and* $f_M : E_M \rightarrow \mathbb{R}$ *is a function that assigns a weight to each edge in* E_M*.*

As can be seen, the EM structures the search space into an edge-weighted graph where each vertex corresponds to a distinct subgraph $G \subseteq \mathbb{N}$. Besides, G is connected to all of its subgraphs with one less node, denoted as $G \to u$, and supergraphs with one additional node, denoted as $G \leftarrow u$. Each edge in the EM, incident on some vertex G, $G \subseteq \mathbb{N}$, corresponds to an *edit* which either inserts or deletes a node from G. By performing a series of edits, G can be transformed to any subgraph $G' \subseteq \mathbb{N}$. The edge weights quantify the impact of the edits on the discriminative potential. We elaborate on how to compute these edge weights in Sec. 7.4.5. Hereon, we use the term node to denote an entity in the GS-network, and vertex to denote a candidate subgraph in the EM. The topmost vertex in the EM represents the *null graph*, and the bottom vertex represents the entire network structure. Fig.7.3 shows the EM of the GS-network in Fig.7.1. The EM is always connected.

As noted earlier, the size of the EM is exponential with respect to the GS-network size and thus cannot be computed or stored in its entirety. However, given any subgraph, we can make local edits to enhance our chances of finding the minimally discriminative subgraphs. We formalize this idea by initiating a Metropolis-Hastings sampling on the EM to guide us towards discriminative subgraphs.

7.4.4 Metropolis-Hastings Sampling

The Metropolis-Hasting (MH) algorithm is a Monte Carlo Markov Chain sampling algorithm whose goal is to sample from a *target distribution* τ. Given a state space $\Omega = \{s_1, \cdots, s_n\}$, let $v_i \geq 0$ be the value of item $s_i \in \Omega$. Our goal is to draw state s_i from τ, where

$$\tau_i = \frac{v_i}{C} \tag{7.3}$$

$C = \sum_{i=1}^{n} v_i$ is a normalizing constant. For large n, C is difficult to compute and thus, computing τ directly is not feasible. MH allows us to simulate τ by converting the state space into an n-state Markov chain with an arbitrary transition matrix Q. Let the current state, X_t, at time step t be i. The MH algorithm performs the following three steps to determine X_{t+1}:

- Draw a random state j with probability Q_{ij}
- Compute the acceptance probability α_{ij}, where

$$\alpha_{ij} = min\left\{1, \frac{\tau_j Q_{ji}}{\tau_i Q_{ij}}\right\} = min\left\{1, \frac{v_j Q_{ji}}{v_i Q_{ij}}\right\} \tag{7.4}$$

- $$X_{t+1} = \begin{cases} j, & \text{with probability } \alpha_{ij} \\ i & \text{with probability } 1 - \alpha_{ij} \end{cases} \tag{7.5}$$

In this formulation, the transition matrix Q is called the *proposal distribution matrix* and α_{ij} is termed as the *acceptance probability*. The optimal proposal distribution is the one that best approximates the target distribution, and the acceptance probability should model how precise the approximation is. The proposal distribution and the acceptance probability can be combined to define the following *one-step transition matrix* T:

$$T_{ij} = \begin{cases} Q_{ij}\alpha_{ij} & \text{if } i \neq j \\ 1 - \sum_{k \neq i} Q_{ij}\alpha_{ij} & \text{if } i = j \end{cases} \tag{7.6}$$

The Markov chain with transition matrix T is *reversible* and *ergodic*. Additionally, the stationary distribution π of the Markov chain converges to the target distribution τ.

7.4.5 MH sampling on the Edit Map

Sec. 7.4.4 describes how the MH algorithm can be used to sample from a target distribution. In this section, we utilize the MH algorithm to sample discriminative subgraphs from the exponential subgraphs search space. In our problem, each subgraph (or vertex) in the EM is a state. The target is to approximate the answer set by sampling only a small subset of highly discriminative subgraphs from the entire search space. Clearly, the quality of the sampled set is critical to the accuracy of our approximation. In MH algorithm, the quality of the stationary distribution depends on two key parameters: the proposal distribution and the acceptance probability. We thus focus on defining these parameters to best approximate the answer set of minimally discriminative subgraphs.

The proposal distribution matrix Q is a function of the edge weights in the EM. The edge weights reflect the quality of the edits on a given subgraph G. An edit is "good" if the newly constructed subgraph increases our chances of finding a minimally discriminative subgraph. While an increase in the discriminative potential can be observed only when nodes are added, Q should allow node deletions so that the sampler does not converge to local optimums. Furthermore, since our goal is to mine minimally discriminative subgraphs and maximize information density, deletions should be preferred over "bad" additions that do not increase the discriminative potential. Thus, to summarize, given a subgraph G, we group all possible edits on G into three classes:

1. **Good addition:** $\phi(G)$ increases due to addition of a node.
2. **Bad addition:** $\phi(G)$ does not increase.
3. **Delete:** Delete nodes to avoid converging to local optimums.

We next formalize these intuitions.

First, we focus on quantifying the edge-weights corresponding to additions. As can be seen, the impact of a node addition on the discriminative potential can be computed only after the NCDT is constructed on the new subgraph following the edit. Consequently, if G has m supergraph neighbors, we need to build m NCDTs. On dense networks, m can be significantly large. Furthermore, this operation needs to be repeated for each graph that we sample in the EM. As a result, an accurate computation of the edge weights is computationally expensive. To reduce this computational burden, we compute an approximation of the actual edge weight based on information gain. Assuming the current state $X_t = G = (V_G, E_G)$, first, the NCDT on G is built. Next, we construct the set $\mathbb{M} = \{N_i = (V_N, E_N, L_i, S_i) | N_i \in \mathbb{N}, f(G, L_i) \neq S_i\}$ of misclassified networks, where $f(G, L)$ is the prediction function. For each node $u \in V_N$ that can be added to G to construct a supergraph $G' = G \leftarrow u \in G_{sup}$, we group them into two sets based on their information gain:

$$A_- = \{u | IG(\mathbb{M}, u) \leq 0 \mid G' = G \leftarrow u \in G_{sup}\}$$
$$A_+ = \{u | IG(\mathbb{M}, u) > 0 \mid G' = G \leftarrow u \in G_{sup}\}$$

A_- and A_+ represent the “bad” and “good” additions respectively. The quality of performing an addition is now quantified as follows:

$$A(u) = \begin{cases} \frac{\Delta}{|A_-|} & \text{if, } u \in A_- \\ (1 - \Delta)\frac{IG(\mathbb{M}, u)}{\sum_{v \in A_+} IG(\mathbb{M}, v)} & \text{if, } u \in A_+ \end{cases} \tag{7.7}$$

where Δ is a small probability distributed evenly among the “bad” additions. As can be seen, the sampler is most likely to select one of the “good” additions based on its information gain. However, since information gain is only an approximation of the actual increase in discriminative potential, with a small probability Δ, the sampler would explore “bad” additions as well. Furthermore, as in the case of deletions, being open to “bad” additions avoids convergence to local optimums. We discuss how to select Δ in Sec. 7.4.5.

Next, we focus on quantifying the utility of deletions. As discussed earlier, deletions are necessary to maximize the information density in the sampled subgraphs, and ensure that the sampler does not explore non-minimally discriminative subgraphs. To achieve this property, the need for deletions on a subgraph G should be dependent on $\phi(G)$. If $\phi(G)$ is low, additions are preferred so that the sampler adds more information and moves to discriminative subgraphs.

On the other hand, if $\phi(G)$ is high, it is preferable to delete nodes and explore other regions of the subgraph search space. We model these requirements using the following proposal distribution:

$$Q_{GG'} = \begin{cases} \frac{\beta}{|G_{sub}|} & \text{if } G' = G \to u \in G_{sub} \\ (1-\beta)\frac{A(u)}{\sum_{G \leftarrow v \in G_{sup}} A(v)} & \text{if } G' = G \leftarrow u \in G_{sup} \end{cases} \tag{7.8}$$

where β models the need for deletions based on $\phi(G)$ and is quantified as the following:

$$\beta = \frac{e^{K\phi(G)}}{e^{K}} \tag{7.9}$$

and K is some large constant. As $\phi(G)$ increases, most of the probability is distributed among deletes, whereas at a low $\phi(G)$, additions are preferred.

The definition of β completes the formalization of the proposal distribution matrix Q. We next focus on defining the acceptance probability $\alpha_{GG'}$. Since our goal is to sample discriminative subgraphs, v_G in Eq. 7.4 can be set to $\phi(G)$. However, with such a score assignment, any supergraphs of $G' \supseteq G$ where $\phi(G') = \phi(G) \geq \theta$ will be considered as a "good" state even though from Definition 7.2.4, G' will never be part of the answer set. Therefore, to model this property, we compute v_G as follows:

$$v_G = \begin{cases} \epsilon \approx 0 & \text{if } \exists G' = G \to u \in G_{sub}, \phi(G') = \phi(G) \\ \epsilon \approx 0 & \text{if } \exists G' = G \leftarrow u \in G_{sup}, \phi(G') > \phi(G) \\ \phi(G) & \text{otherwise} \end{cases} \tag{7.10}$$

Eq. 7.10 ensures that transitions from non-minimally discriminative states are always accepted (cases 1 and 2). If no such conclusion can be drawn from the current state and its neighbors, then transitions are accepted based on their discriminative potentials.

Parameters

Although the proposed model contains two parameters, Δ and K, none of them have a profound impact on the results as long as the parameters are set within an appropriate range. Δ is a small probability that allows exploration of locally "bad" node additions in hope of a globally optimal solution. The results are consistent for any values in the range [0.001,0.005]. In our experiments, we set $\Delta = \frac{|A_-|}{|V_N|}$. For K in Eq. 7.9, any large value in the range [100, 300] would produce consistent results.

7.4.6 Implementation details

Algorithm 8 MINDS($\mathbb{N}, \theta$)

```
1:  $\mathbb{A} := \emptyset$
2:  $t := 0$
3:  $X_t$ := A randomly selected subgraph $G = (V_{X_t}, E_{X_t}) \subseteq \mathbb{N}$
4:  Build NCDT on $X_t$
5:  while $t < maxiter$ do
6:      if $X_t$ is minimally discriminative then
7:          $\mathbb{A} := \mathbb{A} \cup \{X_t\}$
8:          $\mathbb{A} := \mathbb{A} \backslash \{G' \in \mathbb{A} | G' \supseteq X_t,\ \phi(X_t) = \phi(G')\}$
9:      end if
10:     $X_{t_{sub}} := \{X_t \rightarrow u | u \in V_{X_t}, u$ is not a cut-vertex$\}$
11:     $X_{t_{sup}} := \{X_t \leftarrow u | u \notin V_{X_t}, \exists v \in V_{X_t}, (u, v) \in E_N\}$
12:     Compute $Q_{X_t G'}, \forall G' \in X_{t_{sub}} \cup X_{t_{sup}}$
13:     Choose neighbor $G'$ from proposal distribution $Q_{X_t G'}$
14:     Update NCDT for $G'$
15:     $\alpha := \frac{v_{G'} Q_{G' X_t}}{v_{X_t} Q_{X_t G'}}$
16:     $t := t + 1$
17:     if $uniform(0, 1) \leq \alpha$ then
18:         $X_t := G'$
19:     end if
20: end while
21: return $\mathbb{A}$
```

In this section, we discuss the implementation details of MINDS. Alg. 8 presents the pseudocode. At time step $t = 0$, a random subgraph of the GS-network $\mathbb{N}$ is selected as state

X_t and the NCDT on X_t is built (lines 2-4). Since the EM is guaranteed to be connected, the random selection of the graph does not impact the algorithm performance. Next, X_t is checked for minimal discriminativeness, and the answer set $\mathbb{A}$ is updated accordingly (lines 6-8). By leveraging the memoryless property of the MH sampling algorithm, Alg. 8 constructs only the local neighborhood of X_t in the EM (lines 9-10). To further reduce computation cost, only those entries of Q that involve graph X_t are computed (line 11). A neighbor G' is then selected from this proposal distribution and the NCDT is updated accordingly to incorporate the edit operation (lines 12-13). After that, based on α, the state X_{t+1} is selected and the process is repeated for time step $t+1$ (lines 14-17). Finally, after a high number of iterations, the approximated answer set of minimally discriminative subgraphs is returned.

As can be seen in Alg. 8, MINDS is extremely efficient in both storage and computation. Even though the search space is exponential, at any time step, MINDS maintains only two copies of NCDTs in memory: one for the current state and the other for the proposed state. Similarly, the EM or the probability distribution matrix Q is never computed in its entirety. Only the local neighborhood of the current state is computed and stored. As a result, MINDS achieves both of the desired goals: accurate simulation of the target distribution through MH sampling, and computational efficiency through memoryless property of Markov chains.

7.5 Experiments

The objectives of our evaluation procedure are the following:

• Evaluate the sampling quality and scalability of MINDS.

• Investigate the importance of network structure. Towards that goal, we perform MH sampling with Support Vector Machines (SVM) instead of NCDTs and compare the performance.

• Study the impact of noise in GS-networks on the mined patterns. Furthermore, based on the observed results, quantify the statistical significance of the results obtained in the cleaned

Table 7.1: Summary of the GS-networks used. The 'Event' column denotes the event being modeled.

Dataset	#Nodes	#Edges	#Events	Event
D_1	11203	57235	371	Breast Cancer
D_2	9673	39240	183	Liver Metastasis
D_3	1321	5227	35	Embryonic Origin

datasets.

• Analyze the power of minimally discriminative subgraphs on predicting network states.

7.5.1 Datasets

To benchmark MINDS on real GS-networks, we use three different PPIs. Each of the PPIs represents the human protein interaction network. Although all three networks are drawn from the same species, they are curated by three different agencies and differ in the various cellular processes being modeled. Consequently, no mapping exists between nodes across networks. Table 8.4 summarizes the GS-networks. D_2 is obtained from [130], whereas D_1 and D_3 are obtained from [119]. Fig.7.4(a) shows the degree distribution of each of these networks. As expected, they display a scale-free behavior. D_1, D_2 and D_3 model the events of breast cancer, liver metastasis, and embryonic origin of human tissues respectively. The "#Events" column denotes the number of network events/snapshots observed. Each event is associated with local node labels and a global state. The local node labels represent the protein expression levels and the global state indicates the clinical outcome of the event being modeled. To discretize the protein expression levels, we follow the standard procedure from systems biology [130]. First, the expression levels of each protein are standard normalized so that the mean and the standard deviation is 0 and 1 respectively. Next, the expression levels of all proteins across all events are sorted and the values in the top 25% are set to 1. The remaining values are set to 0. A node label 1 therefore indicates the corresponding protein to over-express and 0 indicates normal behavior.

7.5.2 Experimental Setup

For experiments evaluating quality of MINDS, we select the maximum possible subset of network events from each dataset such that the distribution of the global states is balanced. A balanced set ensures that a majority-class classifier can only achieve an accuracy of 0.5. Otherwise, we use the entire datasets. Unless specifically mentioned, we iterate the sampler for $100,000$ time steps. We set the default threshold for discriminative potential to 0.8. The value of constant K in Eq. 7.10 is set to 100. Typically, K has minimal impact on the results as long as $K > 100$.

MH with SVM

To highlight the importance of capturing the underlying network structure, we replace NCDT with SVM as the learning methodology in the MH sampling step. More specifically, at any subgraph $G \subseteq \mathbb{N}$, $\phi(G)$ is computed based on SVM with linear kernel. The SVM is not constrained by network connectivity as long as all features (nodes) are part of G. In MH sampling with SVM, only the proposal distribution matrix is altered based on the feature ranking mechanism outlined in [131]. Instead of information gain, the importance of a node u is quantified based on its absolute weight value $w(u)$ in the learned SVM model. Thus, at each state with graph G, two SVM models are learned: SVM model M_G on G, and SVM model $M_{G_{sup}}$ that uses all nodes in G in addition to the nodes that can be added to G on the EM. $w(u) \in M_G$ quantifies the importance of deleting node u, and $w(u) \in M_{G_{sup}}$ quantifies the importance of adding node u to G. Thus,

$$Q_{GG'} = \begin{cases} (\beta)\frac{|w(u)|}{\sum_{G \rightarrow v \in G_{sub}} |w(v)|} & \text{if } G' = G \rightarrow u \in G_{sub} \\ (1-\beta)\frac{|w(u)|}{\sum_{G \leftarrow v \in G_{sup}} |w(v)|} & \text{if } G' = G \leftarrow u \in G_{sup} \end{cases}$$

The formulation of β (7.9) and α (7.10) remains the same.

7.5.3 Performance analysis of sampling

First, we evaluate the quality of the subgraphs sampled from the EM. The quality of the sampling procedure is the single most important aspect that affects the accuracy of the computed answer set. To obtain an accurate approximation, the sampler should often visit graphs that have high discriminative potential. Therefore, to analyze the desired correlation between visit count and the discriminative potential of a subgraph, we plot the likelihood of a subgraph being visited given its discriminative potential. Figs.7.4(b)-7.4(c) demonstrate the results on two of the largest datasets D_1 and D_2. To set the baseline, we perform random sampling of subgraphs. As can be seen, majority of the subgraphs visited by MINDS have $0.9 \leq \phi(G) \leq 1$. On the other hand, if we select subgraphs randomly from the network, the visit count is uniformly distributed across all values of discriminative potential. During random selection, we ensure that the subgraph sizes are drawn from the same distribution visited by MINDS. For SVM-guided MH sampling, a trend similar to MINDS is also observed. However, the sampler spends more time in the range $0.7 \leq \phi(G) \leq 0.9$. This result shows that the proposed formulations of the proposal distribution matrix and the acceptance probability, for both SVM and NCDT, are effective in separating out the discriminative subgraphs from those that are non-discriminative.

The second important aspect of the sampling procedure that affects the quality of the answer set is the size of the visited subgraphs. Recall that for subgraph G to be minimally discriminative, none of G's subgraphs can have a higher discriminative potential than G. Clearly, that reduces to sampling small subgraphs but with high discriminative potentials. Thus, to analyze how well the proposed technique conforms to this desired sampling property, we analyze the distribution of the subgraph sizes that are sampled. First, we plot the distribution of the subgraph sizes against the discriminative potential. As can be seen in Figs. 7.4(d)-7.4(e), the information density in subgraphs sampled by MINDS is significant higher than in SVM for

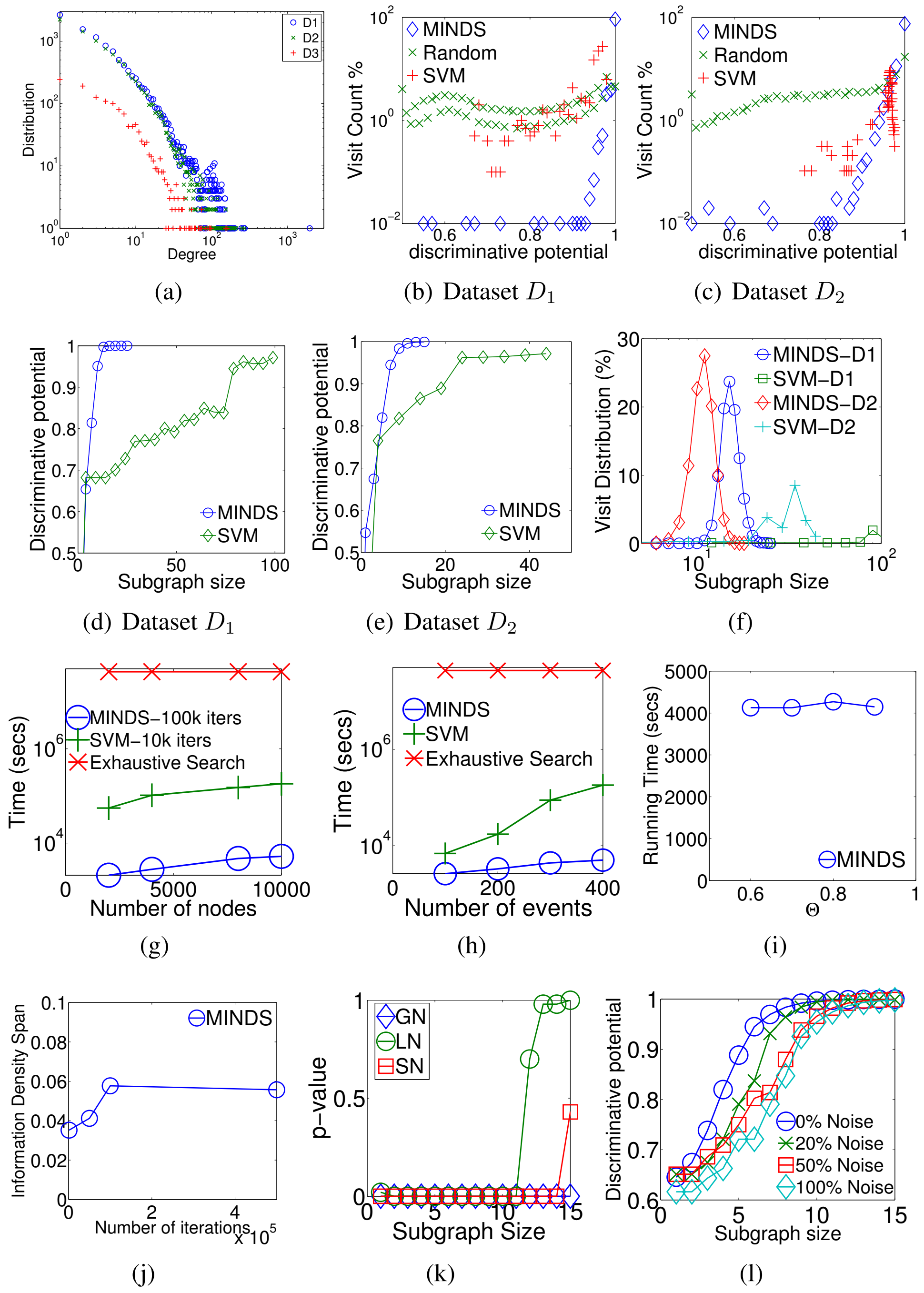


Figure 7.4: (a) Degree distribution in datasets D_1, D_2 and D_3. (b-c) Growth rate of visit count with discriminative potential for dataset D_1 and D_2. (d-e) Growth rate of subgraph size with discriminative potential. (f) Distribution of sampled subgraph sizes. Growth rate of the running time with (g) network size, (h) number of events, and (i) θ. (j) Quality of the answer set against number of iterations. (k) Statistical significance of the patterns mined by MINDS. (l) Impact of structural noise on discriminative subgraphs.

subgraph sizes above 5. While the discriminative potential of subgraphs sampled by MINDS saturates at sizes around 20, to achieve the same potential, SVM requires significantly larger subgraphs. This result highlights the importance of capturing the network structure through NCDTs. Since SVM is oblivious to network connectivity, it only utilizes the information encoded in the network nodes. On the other hand, the structural constraint in NCDT captures the network through which the process being modeled evolves, and consequently, utilizes the information encoded in both the nodes as well as edges. The importance of capturing the structure is further established in Fig.7.4(f). Fig. 7.4(f) analyzes the sampled subgraph sizes and plots their distribution against visit count. Since the information densities in SVM sampled subgraphs are significantly lower than MINDS, much of the SVM sampling is restricted on large subgraphs to achieve a high discriminative potential. Consequently, most of the subgraphs sampled by SVM are not minimally discriminative. We further analyze the importance of structure in Sec. 7.5.4.

Next, we focus on analyzing the scalability of MINDS. First, we evaluate the growth rate of the running time against network size in Fig. 7.4(g). To construct GS-networks of varying sizes, we select subgraphs from D_1. To set the baseline, we first attempt an exhaustive subgraph exploration on a network containing only 60 nodes. However, due to the exponential subgraphs search space, the exhaustive search failed to complete even after 12 hours. Given this context, even on a network containing $10,000$ nodes, MINDS is more than three orders of magnitude faster than an exhaustive exploration on a network of size 60. Compared to SVM, MINDS is more than a magnitude faster than SVM. Fig. 7.4(h) analyzes scalability against the number of events in the network. As can be seen, the running time of MINDS grows linearly and is more than 10 times faster than SVM. Finally, we evaluate the growth rate of the running time against the discriminative potential threshold θ. As expected from the formulation in Sec. 7.4.5, the running time is constant since other than the randomness in the sampling procedure, θ does not change the number of computations performed.

Fig. 7.4(j) analyzes the quality of the answer set with the number of iterations. To quantify quality, we verify whether the answer set captures the entire spectrum of the minimally discriminative subgraphs. For that purpose, we use the metric of *information density span*. First, we define *information density* of a graph $G = (V_G, E_G)$ as $\frac{\phi(G)}{|V_G|}$. The information density span of the answer set is the difference between the highest and the lowest information densities among all graphs in the answer set. The highest and the lowest information densities define the boundaries of the answer set, and we consider the answer set to converge once the density span stops expanding. As can be seen in Fig.7.4(j), after $100,000$ iterations, the increase in the span is minimal. We use information density instead of discriminative potential, since graphs in the answer set should be both discriminative and compact.

Although the above experiments indicate an excellent performance, an important question remains to be answered: *How accurate is our approximation?* To answer this question, we selected a sub-network of D_1 containing 60 nodes and tried computing the ground truth answer set. Unfortunately, the process could not be completed even after 12 hours during which it analyzed more than 100×10^6 subgraphs. The projected time based on the number of subgraphs that were left unprocessed was 200 hours. Due to this huge computational cost, computing the ground truth even on miniature networks is not feasible. Thus, we use an alternative strategy for constructing the ground truth. We synthetically implant NCDTs on the network structure of D_1 and generate a balanced set of network events ensuring that the implanted NCDTs have an accuracy of 1. While generating the network events and the accompanying node labels, we set the node labels according to the functions encoded by the implanted NCDTs. For nodes that are not used by the NCDTs, we set the labels arbitrarily. Due to this controlled construction, subgraphs spanning the implanted NCDTs have discriminative potentials of 1.0. Now, to evaluate the accuracy, we execute MINDS on the constructed GS-network and verify whether the discriminative subgraphs are extracted.

Table 7.2 presents the results as the sizes of the implanted subgraphs are varied (we explain the results for graph $N = (V_N, E_N)$ in Sec. 7.5.4. In all of our experiments, MINDS is able to identify a subgraph with a discriminative potential of 1. Interestingly, for $|V_I| \geq 5$, MINDS is able to identify a smaller subgraph G and still achieve an accuracy of 1. Due to the human-mediated construction of the implanted NCDTs, the trees are not always optimal. MINDS is able to identify that non-optimality and construct a more complex and compact NCDT while retaining the same accuracy. This result establishes that MINDS is efficient in both accurately approximating the answer set and regularizing the network.

7.5.4 Impact of noise

What happens when the GS-network is noisy? How much of the signal is lost due to inaccuracies in the network structure? In this section, we answer these questions. There are three sources of noise: the network structure (SN), the local expression levels at nodes (LN), and the global network class (GN). To understand the impact of noise, we perform permutation tests [132]. More specifically, first, we create a null hypotheses by introducing noise in the PPI and run MINDS on the noisy network. Due to the addition of noise, if the original network contains any prediction signal, we expect to lose it. This process of introducing noise is repeated one million times to compute the distribution of discriminative potentials for a subgraph of a given size. Based on the null hypothesis, we compute the p-value for the distribution observed in Fig.7.4(e). For example, in Fig.7.4(e), a subgraph of size 6 has an average discriminative

Table 7.2: Accuracy of MINDS against ground-truth answer set and the impact of noise on network structure. $I = (V_I, E_I)$ denotes the implanted discriminative subgraph, $G = (V_G, E_G)$ and $N = (V_N, E_N)$ denote the best subgraph discovered in the original and noisy GS-networks respectively.

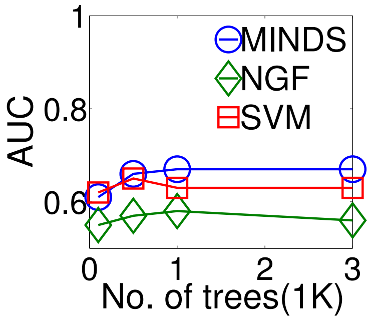
$\lvert V_I\rvert$	$\lvert V_G\rvert$	$\lvert V_N\rvert$	$\lvert V_G \cap V_I\rvert$	$\lvert V_N \cap V_I\rvert$	$\frac{\lvert V_G \cap V_I\rvert}{\lvert V_G \cup V_I\rvert}$	$\frac{\lvert V_N \cap V_I\rvert}{\lvert V_N \cup V_I\rvert}$	$\phi(G)$	$\phi(N)$
3	4.93	10.02	2.59	2.31	0.64	0.28	1	1
5	5.08	8.53	3.42	3.41	0.57	0.38	1	1
8	6.52	8.02	4.61	4.61	0.48	0.42	1	1
10	8.02	8.47	5.38	5.26	0.43	0.4	1	1

potential of 0.94 and our goal is to compute the statistical significance of this event. To introduce structural noise (SN), we randomly construct edges between nodes while keeping the total number edges in the original network, and the local and global labels intact. For LN and GN, we adopt similar strategies by permuting the local node labels and global snapshot labels respectively. As in SN, we ensure that the distributions of the noisy local and global states are the same as in the original datasets.

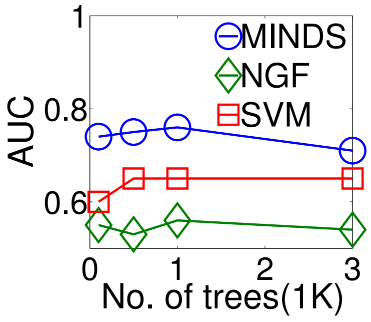
Fig.7.4(k) shows the results for all subgraph sizes sampled by MINDS in dataset D_2. As can be seen, regardless of the noise introduction policy, the p-values for the majority of the subgraph sizes are 0. The significance of the results decreases for sizes above 12 in the SN and LN methods due to diminishing return of marginal gains. More specifically, in the original dataset, the discriminative potential saturates for subgraphs above sizes 8. In the noisy dataset, even after perturbing the local states, MINDS is able to identify discriminative subgraphs when their sizes are above 12. However, due to the permutation, the saturation happens from size 12 onwards instead of 8.

To further understand the impact of noise, we analyze the information density of the discriminative subgraphs as the amount of SN is varied. A noise level of 20% indicates that 20% of the edges are randomly constructed; the remaining edges are the same as in the original network. Fig. 7.4(l) demonstrates the results. As expected, with increase in noise level the structural signal is lost, and consequently, the average discriminative potential for a given subgraph size decreases. Similar results are observed for LN and GN as well.

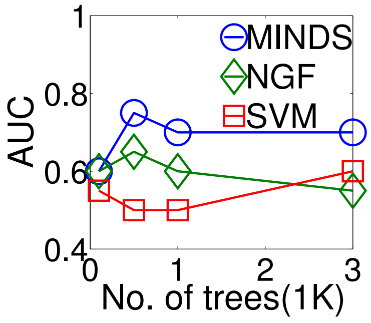
Finally, we investigate the impact of SN on discovery of the ground-truth answer set. As in the verification procedure earlier, synthetic NCDTs are implanted on the GS-network. Next, SN is introduced and we compare the discriminative subgraphs $N = (V_N, E_N)$ identified in the noisy network with the implanted ones. As can be seen in Table 7.2, although discriminative subgraphs are still identified, they are significantly larger in size. This is in sharp contrast to the results in original network where MINDS is able to identify subgraphs that are actually smaller



(a) Breast Cancer



(b) Liver Metastasis



(c) Embryonic Origin

Figure 7.5: (a-c) Growth rate of the AUC with number of trees for the events.

than the implanted ones. This is a direct consequence of the structural signal getting lost due to shuffling of edges.

Overall, the analysis reveals the following:

1. The network structure contains discriminative information that should be captured in the classifier for optimum performance.

2. If the network is noisy, the sizes of the discriminative subgraphs grow to compensate for the missing information.

7.5.5 Analysis of prediction quality

In this section, we verify the predictive power of the mined subgraphs for network state classification. To evaluate prediction performance, we perform 5-fold cross validation. First, we compute the answer set on the training dataset. Then, from each of the minimally discriminative subgraphs, we extract the learned NCDT and the state predicted on the testing set by the majority of the NCDTs is considered as the collective predicted state. The accuracy is quantified by the area under the ROC curve (AUC).

To benchmark our technique, we use the state-of-the-art classifier *Network Guided Forests (NGF)* [119] designed specifically for PPIs, SVM. NGF employs a greedy sampling strategy similar to the algorithm outlined in Sec. 7.4.2. By sampling multiple times, NGF constructs a random forest. Furthermore, NGF incorporates domain specific information, such as favoring high-degree proteins, and a second round of clustering to identify decision modules to boost the performance. In contrast, MINDS incorporates no domain specific information. Figs. 7.5(a)-7.5(c) demonstrate the classification accuracy as the number of trees is varied for MINDS and NGF. As can be seen, MINDS achieves a higher AUC across all network events that is up to 65% higher than NGF and SVM.

7.6 Related Work

While the idea of a GS-network has not been formalized before, the problem of mining protein modules from PPIs has been studied. Prior work in systems biology has indicated that the network structure is critical towards identifying discriminative protein modules and have focused on network regularization [133, 134] and classification [119, 130]. However, with the exception of [119], existing techniques assume homogeneous activity on entire protein modules and are only capable of identifying simple logic functions such as the sum or the multiplication of the expression levels. On the other hand, [119] employs a greedy strategy by starting from the most informative node in the network and then building a tree within that neighborhood. As illustrated in Sec. 7.4.2, a greedy strategy is susceptible to converging to a local optima. Furthermore, as opposed to solving the problem only in the context of PPIs, our work proposes a generalized algorithm for GS-networks. The features mined by MINDS can not only be employed for classification, but also for regression, network regularization and real-time monitoring.

As discussed earlier in Sec. 7.1, dynamic networks [117, 121, 122] and mining discriminative subgraphs from graph databases [120, 123–126] are the two closest lines of work from the computer science community. However, both fail to model the problem being proposed here. Dynamic networks do not contain global states and each snapshot is ordered temporally. Mining discriminative subgraphs from graph databases, on the other hand, assume a database of multiple graphs and mine subgraphs that are statistically "over-represented" in one of the classes.

7.7 Conclusion

In this paper, we formalize the concept of a global-state network and design a technique called *MINDS* to learn the network-encoded evolution rules that determine the global network state. MINDS learns local prediction models by constructing network-constrained decision trees on minimally discriminative subgraphs. The mined patterns regularize the network and provide the platform for an array of higher level tasks such as classification, regression and network monitoring. To tackle the exponential subgraph search space, MINDS structures the space in the form of an Edit Map and performs MH sampling on the map to mine discriminative subgraphs. MINDS aggressively leverages the memoryless property of Markov chains and drastically scales up the mining procedure. Extensive experiments performed on real GS-networks demonstrate MINDS to be efficient in mining patterns that are accurate and statistically significant. MINDS is up to 4 orders of magnitudes faster than baseline techniques. Overall, MINDS unleashes the potential of minimally discriminative subgraphs towards a myriad of applications such as real time recommendation engines, cost-effective clinical treatments and efficient designing of marketing campaigns.

Part III

Predicting network processes

Understanding and mining patterns from network data are important. However, the final goal is to model and forecast networked behaviors in the future using the insights we have gained. Thus, I next propose a number of models for predicting different network processes, including traffic flow and online information spreads.

In Chapter 8, I predict the movement of crowds in a city, which is strategically important for traffic management, risk assessment, and public safety. I propose predicting two types of flows of crowds in every region of a city based on big data, including human mobility data, weather conditions, and road network data. To develop a practical solution for citywide traffic prediction, I first partition the map of a city into regions using both its road network and historical records of human mobility. Our problem is different than the predictions of each individual's movements and each road segment's traffic conditions, which are computationally costly and not necessary from the perspective of public safety on a citywide scale. To model the multiple complex factors affecting crowd flows, I decompose flows into three components: seasonal (periodic patterns), trend (changes in periodic patterns), and residual flows (instantaneous changes). The seasonal and trend models are built as intrinsic Gaussian Markov random fields which can cope with noisy and missing data, whereas a residual model exploits the spatio-temporal dependence among different flows and regions, as well as the effect of weather. Experiment results on three real-world datasets show that our method is scalable and outperforms all baselines significantly in terms of accuracy.

In Chapter 9, I predict the popularity of online content in social networks, which is important in many applications, ranging from ad campaign design, web content caching and prefetching, to web-search result ranking. Earlier studies target this problem by learning models that either generalize behaviors of the entire network population or capture behaviors of each individual user. In this paper, I claim that a novel approach based on group-level popularity is necessary and more practical, given that users naturally organize themselves into clusters and that users within a cluster react to online content in a uniform manner. I develop a novel framework by first grouping users into cohesive clusters, and then adopt tensor decomposition to make predictions. In order to minimize the impact of noisy data and be more flexible in capturing changes in users' interests, our framework exploits both the network topology and interaction among users in learning a robust user clustering. The PARAFAC tensor decomposition is adapted to work with hierarchical constraint over user groups, and I show that optimizing this constrained function via gradient descent achieves faster convergence and leads to more stable solutions. Extensive experimental results over two social networks demonstrate that our framework is scalable, finds meaningful user groups, and significantly outperforms eight baseline methods in terms of prediction accuracy.

Finally, in Chapter 10, I propose a novel model-free approach to forecasting complex network phenomena – such as information cascades in online social networks. In forecasting, a recent trend has been to forgo the use of parsimonious models in favor of models with increasingly large degrees of freedom that are trained to learn the behavior of a process from historical data. Extrapolating this trend into the future, eventually I would renounce models all together. *But is it possible to forecast the evolution of a complex stochastic process directly from the data without a model?* In this work I show that model-free forecasting is possible. I present SED, an algorithm that forecasts process statistics based on relationships of statistical equivalence using two general axioms and historical data. To the best of our knowledge, SED is the first method that can perform axiomatic, model-free forecasts of complex stochastic processes. Our simulations using simple and complex evolving processes and tests performed on a large real-world dataset show promising results.

Chapter 8

FCCF: Forecasting Citywide Crowd Flows Based on Big Data

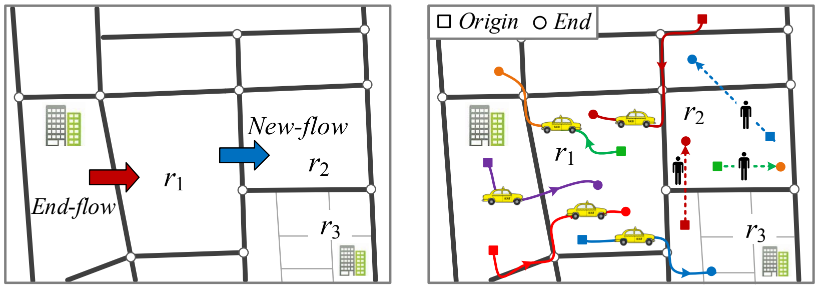
8.1 Introduction

Predicting the movement of crowds in a city is strategically important for traffic management, risk assessment, and public safety. For example, 36 people died and 47 others were injured during the stampede in the Shanghai Bund in 2015, turning a New Year Celebration into a catastrophic accident. Massive flows of people streamed into a strip region which was not designed to hold them to watch the New Year's Eve Light Show, making the region overloaded and difficult for police to control. A similar stampede happened in the 2010 German Love Parade. If we can predict the arrival of crowds in a region and know the crowd flows would exceed the region's safe capacity, we can launch emergency mechanisms (e.g., sending warnings to people and conducting traffic controls) or evacuate people in advance.

Prior research on crowd movements has focused on the prediction of each individual's movement (e.g., [135, 136]), and traffic conditions on road segments (e.g., [137, 138]). While these problems provide a *detailed view* of city traffic, they may have heavy computational costs due to the huge number of roads, vehicles, and people in a big city, and are also not necessary from the perspective of public safety at a citywide scale. Furthermore, predicting each individual's movement is difficult to do given the diversity of individual life patterns and the randomness of human behavior.

Given the above limitations, in this paper we investigate a *macro-level view* of crowd movements by predicting two types of *flows of crowds* in every region of a city based on big data, including human mobility data, weather conditions, and road network data. As shown in Figure 8.1a, a region (such as r_1) is bound by major roads, and the two flows are: 1) *new-flow*, the traffic of crowds originating from a region at a given time interval (e.g., people start driving from a parking spot); and 2) *end-flow*, the traffic of crowds terminated in a region (e.g., people stop driving and park their cars). Intuitively, new-flow and end-flow track the origins and final destinations of the crowds. These two flows thus summarize the movements of crowds and are enough for traffic management and risk assessment.

The two crowd flows can be measured *individually* by the number of vehicles driven on roads, or the number of people traveling in public transportation systems, or the number of pedestrians, or *all together* if data is available. The data representing human mobility can be the GPS trajectories of vehicles, or the mobile phone signals of users, or card swiping data in public transportation systems such as the subway or bike sharing systems. For example, in Figure 8.1b, according to the GPS trajectories and measured by the number of vehicles, the



(a) Two types of flow to be predicted (b) Illustration of measurement of flow

Figure 8.1: Crowd flows in a region

new-flow and end-flow of r_1 over the outlined 30 minutes are (2, 3) respectively. Likewise, the two types of flows are (2, 2) in r_2 in terms of mobile phone signals, measured by the number of pedestrians. If both the GPS and phone signals are tracked, we can consider the crowd flows of region r_3 to be (1, 1). During a time interval, if a person starts and ends his/her trajectory in the same region, s/he will be counted in both the new-flow and end-flow of that region. Note that our proposed framework can also be applied as is to other definitions of crowd flows.

The challenges of our research are three-fold. 1) *Multiple complex factors*: There are multiple complex factors affecting crowd flows, which can be captured thanks to the advent of big data. For instance, the crowd flows in a region usually have a daily and weekly periodic pattern, which might change over time, as well as instantaneous changes due to noise, weather conditions, and other social events. 2) *Flow dependencies*: There are dependencies between different types of flows in a region (intra-region dependence) and those among different regions (inter-region dependence) over time. For example, the increase of end-flow in a region in the current hour may raise its new-flow over the next hour. Similarly, the end-flow of a region is influenced by the new-flows of its neighbors. 3) *City-scale prediction*: While we need the prediction instantly, a city-scale prediction is computationally intensive. Therefore, an efficient predictive model is needed. In addition, different regions could have different scales of crowd flows. Sparse flow data in some regions will prevent us from learning a stable periodic pattern inherent in crowd flows, thus reducing prediction accuracy.

To tackle these challenges, we decompose each type of flow in a region into *three ingredients*: seasonal, trend, and residual flows, proposing a three-step predictive method to capture each of them. The contributions of our research:

- To deal with data sparsity and construct a practical citywide solution, we first divide a city into low-level regions using its road network, and then group adjacent low-level regions with similar crowd flow patterns using graph clustering. The obtained high-level regions have more stable (thus easier to predict) crowd flows, and also provide a meaningful and more manageable representation of the citywide crowd flows.

- Based on the Intrinsic Gaussian Markov Random Field (IGMRF), we propose a seasonal model to predict the periodic flow, and a trend model to predict the change of the seasonal pattern over time. Our IGMRF models are robust to noisy and missing data, and scalable to big data.
- We propose a spatio-temporal residual model to predict the instantaneous deviations from the periodic patterns of flows, based on the historical flow data of a region and those of its neighbors as well as weather information. The model uses a Bayesian network to capture the transition probability among the regions. We combine the seasonal, trend, and residual models to obtain our **FCCF** model (**F**orecasting **C**itywide **C**rowd **F**lows).
- Experiments[1] on three real-world datasets (taxi and bike data) show that **FCCF** is scalable and outperforms baseline approaches significantly in terms of accuracy.

The rest of this paper is as follows: Section 8.2 overviews our framework. Section 8.3 discusses the division of a city into regions. Section 8.4 proposes the seasonal and trend models. Section 8.5 proposes the spatio-temporal residual model. Section 8.6 reports our experimental results. Section 8.7 discusses related works and Section 8.8 concludes the section.

8.2 Overview

8.2.1 Preliminaries

Regions: There are many definitions of a location in terms of different granularities and semantic meanings. In this study, we first partition a city into a number of low-level regions by city roads, using a map segmentation method [139]. Consequently, each region is bound

[1]Our data and code are available at `https://www.microsoft.com/en-us/research/publication/forecasting-citywide-crowd-flows-based-big-data/`

by roads, carrying a semantic meaning of neighborhoods or communities, as illustrated in Figure 8.1. These regions are low-level, that is, they can be very small and have very little data for prediction. Therefore, we propose grouping adjacent low-level regions with similar crowd flow patterns into high-level regions using a graph clustering approach. We will discuss the clustering step in Section 8.3. We denote the set of high-level regions as $\mathcal{R} = \{u_1, u_2, ..., u_m\}$, where m is the number of high-level regions. We then use the high-level regions as the minimal unit of location in the following study, though a region can be a uniform grid or defined by the governments in other applications.

Definition 8.2.1 (Crowd flows) *The movement of an individual can be recorded as a spatial trajectory $\mathcal{T}$, which is a sequence of time-ordered points, $\mathcal{T} : p_1 \rightarrow p_2 \rightarrow ... \rightarrow p_{|\mathcal{T}|}$, where each point $p_i = (a_i, b_i, t_i)$ has a geospatial coordinate position (a_i, b_i) and a timestamp t_i, and $|\mathcal{T}|$ is the number of points in $\mathcal{T}$. Likewise, the movement of crowds can be represented by a collection of trajectories $\boldsymbol{P}$. Specifically, for a region u, the two types of flows of crowd (crowd flows) at timestamp t, namely new-flow and end-flow, are defined respectively as*

$$x_{u,t}^{new} = |\{\mathcal{T} \in \boldsymbol{P} : (a_1, b_1) \in u, t_1 = t\}|$$
$$x_{u,t}^{end} = \left|\{\mathcal{T} \in \boldsymbol{P} : (a_{|\mathcal{T}|}, b_{|\mathcal{T}|}) \in u, t_{|\mathcal{T}|} = t\}\right|$$

where $(a_i, b_i) \in u$ means that point p_i lies within region u.

Problem 8.2.1 (Forecast Citywide Crowd Flows) *For $\forall u \in \mathcal{R}$ and $\forall \theta \in \{new, end\}$, given the historical crowd flows $x_{u,t}^{\theta}$ for $t = 0, ..., n-1$, predict $x_{u,n}^{\theta}$.*

8.2.2 A Case Study of Taxi Trajectories

We now analyze a case study of Beijing taxi GPS dataset **BJ** (detailed in Section 8.6.1). First, we partition Beijing into 372 low-level regions based on its road network as done in [139]

(Figure 8.4a). Since 372 regions are too many to monitor at city scale, we further cluster the low-level regions into 26 high-level regions with comparable crowd flow volumes (Figure 8.4b). The region IDs are also provided in this figure.

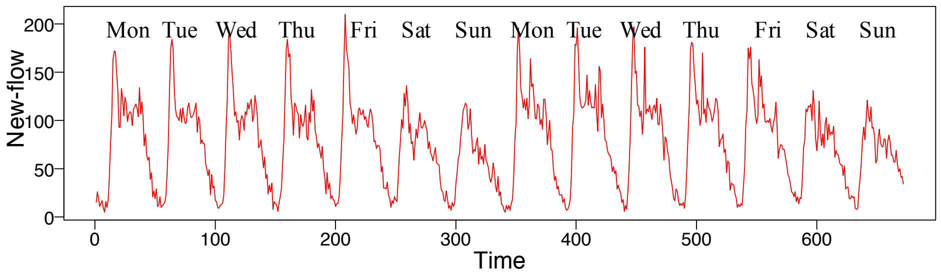
We obtain crowd flows by tracking the trajectories of taxis using their GPS signals. For example, Figures 8.2a shows the new-flow in region 22 during May 4^{th}-17^{th}, 2015, where each timestamp is 30 minutes. Clearly, the flows have a periodicity of day and week—a seasonal effect. Further, we can see a trend of change in this seasonal pattern over time, which may differ per region and per time of day. For example, as the weather got warmer, new-flows at 6am of region 22 in Figure 8.2b clearly got bigger on average. Whereas, new-flows at 3pm in this region (Figure 8.2c) got smaller possibly because it is less comfortable to travel outside when the temperature is too high.

Neighboring regions can affect each other due to crowd flows among them. Figures 8.2d and 8.2e show an example of two neighboring regions 1 and 3 at the top right corner of Figure 8.4a. The new-flow of region 3 and the end-flow of region 1 deviate from their seasonal patterns at the same time and in the same direction (as marked by the blue and red arrows), suggesting their dependence on each other.

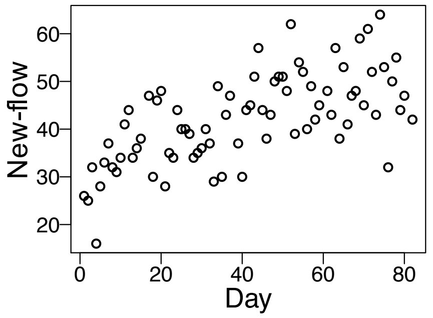
8.2.3 Prediction Framework

We discuss the segmentation of a city map into regions in Section 8.3. We track and predict crowd flows in these regions. Based on the observations in Section 8.2.2, we propose a prediction framework as shown in Figure 8.3.

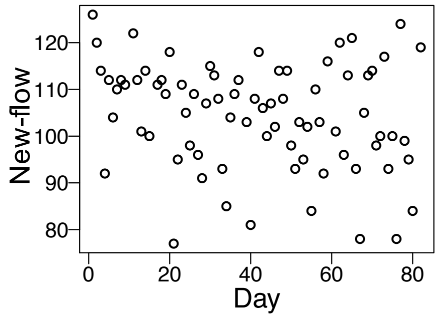
Figure 8.3a shows the modelling framework for the crowd flows of a region. Specifically, we decompose a crowd flow time series $\boldsymbol{x} = (x_0, x_1, ..., x_{n-1})$ over n timestamps into three components: a seasonal component $\boldsymbol{s}$ capturing the periodic pattern, a trend component $\boldsymbol{y}$ capturing the offset from the periodic pattern for each timestamp in a period, and a spatio-



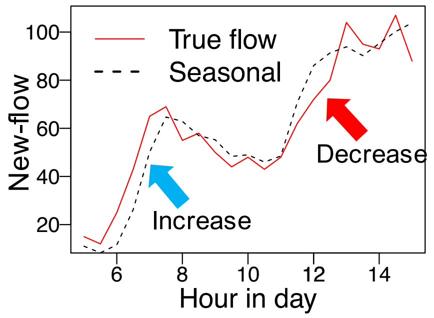
(a) New-flow in region 22, May 04-17, 2015



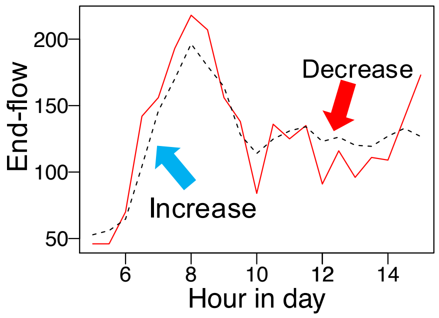
(b) Trend at 6 am (region 22)



(c) Trend at 3 pm (region 22)



(d) Region 3, June 3rd, 2015



(e) Region 1, June 3rd, 2015

Figure 8.2: Beijing data (the regions are shown in Figure 8.4b). One timestamp is 30 minutes. (a) new-flow of region 22 during two weeks of May, 2015. (b, c) Trend of new-flow at 6am and 3pm for region 22 from March to June, 2015. (c, d) new-flow and end-flow of two neighboring regions (regions 3 and 1) during June 03, 2015.

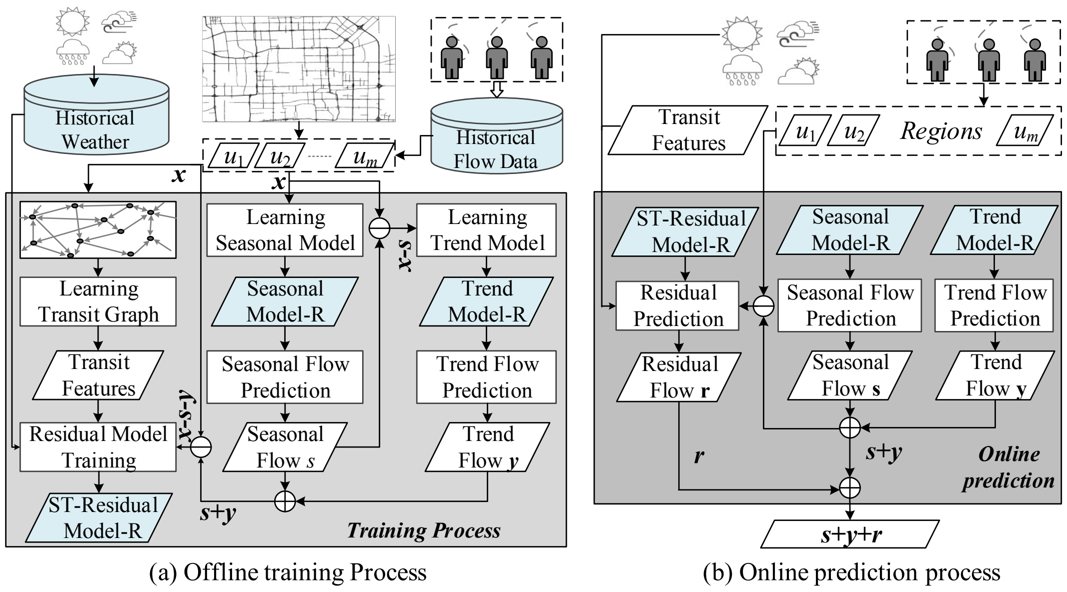


Figure 8.3: Framework overview.

temporal residual component $\boldsymbol{r}$ capturing the instantaneous changes. Thus,

$$\boldsymbol{x} = \boldsymbol{s} + \boldsymbol{y} + \boldsymbol{r} \tag{8.1}$$

We use only temporal information to model $\boldsymbol{s}$ and $\boldsymbol{y}$. The seasonal model $\boldsymbol{s}$ is learned on the original flow $\boldsymbol{x}$, and the trend model $\boldsymbol{y}$ is learned on the residual $\boldsymbol{x} - \boldsymbol{s}$. After that, the residual $\boldsymbol{r}$ is learned for $\boldsymbol{x} - \boldsymbol{s} - \boldsymbol{y}$.

Assume that the periodicity in $\boldsymbol{x}$ has a length of period F, that is, $s_t = s_{t \bmod F}$ $\forall t = 0, ..., n-1$, then we can divide $\boldsymbol{x}$ into a sequence of periods as shown in the first two lines of Table 8.1 for $n = 10$ and $F = 4$. In general, $\boldsymbol{x}$ will contain $n_y = \lfloor n/F \rfloor + 1$ periods, where period j contains timestamps in the range $[jF, (j+1)F - 1]$, for $j = 0, ..., n_y - 1$.

Since the timestamps within a period may have different evolutionary trends over time (see Figures 8.2b and 8.2c), we build a seperate trend model for each of them. Therefore, we decompose $\boldsymbol{x}$ into three components as in Table 8.1:

$$x_t = s_{t \bmod F} + y_{t \bmod F, \lfloor t/F \rfloor} + r_t \tag{8.2}$$

where:

- $s_{t \bmod F}$ is the seasonal flow at the $(t \bmod F)$-th timestamp within a period.
- $y_{t \bmod F, \lfloor t/F \rfloor}$ is the offset from the seasonal flow of the $(t \bmod F)$-th timestamp in period $\lfloor t/F \rfloor$.
- r_t is the residual flow at time t.

In particular, we model $\boldsymbol{s}$ as a time series of length F: $\boldsymbol{s} = (s_0, s_1, ..., s_{F-1})$. For each i-th timestamp of a period ($i = 0, ..., F-1$), we model its trend across different periods as a time series $\boldsymbol{y_i}$ of length n_y: $\boldsymbol{y_i} = (y_{i0}, y_{i1}, ..., y_{i,n_y-1})$, where y_{ij} is the offset from the seasonal pattern of the i-th timestamp in period j. Finally, the residual $\boldsymbol{r}$ is modeled as a time series of length n: $r = (r_0, r_1, ..., r_{n-1})$.

Period 0				Period 1				Period 2	
x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
s_0	s_1	s_2	s_3	s_0	s_1	s_2	s_3	s_0	s_1
y_{00}	y_{10}	y_{20}	y_{30}	y_{01}	y_{11}	y_{21}	y_{31}	y_{02}	y_{12}
r_0	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9

Table 8.1: Decomposed flow $\boldsymbol{x} = \boldsymbol{s} + \boldsymbol{y} + \boldsymbol{r}$ for $n = 10$, $F = 4$.

Both $\boldsymbol{s}$ and $\boldsymbol{y}$ are built based on IGMRFs (Section 8.4), scalable to big data, and robust to noisy and missing data.

For the residual $\boldsymbol{r}$ (Section 8.5), we first propose a Bayesian network to model the transition probability of crowds among regions. By applying this transition probability into $\boldsymbol{r}$, we obtain the residual transit flows among regions—the transit features in Figure 8.3a—which capture the dependence among neighboring regions. Finally, we combine the transit features (inter-region dependence), the history of all types of flows of a region (intra-region dependence), and the weather data into a spatio-temporal residual model to predict $\boldsymbol{r}$.

We use the trained models to make online predictions for crowd flows of regions as described in Figure 8.3b.

For clarity, Table 8.2 lists the notations used in this paper.

$\mathcal{R} = \{u_1, u_2, ..., u_m\}$	The set of all high-level regions
m, n	Number of regions, and number of timestamps
$x^{\theta}_{u,t}$	Crowd flow of region u at time t; $\theta \in \{new, end\}$
$\boldsymbol{x} = (x_0, ..., x_{n-1})$	Vector representing a flow time series
F	Length of a period in $\boldsymbol{x}$
$n_y = \lfloor n/F \rfloor + 1$	Number of periods in $\boldsymbol{x}$
$\boldsymbol{s} = (s_0, ..., s_{F-1})$	Seasonal component of $\boldsymbol{x}$
$\boldsymbol{y_i} = (y_{i0}, ..., y_{i,n_y-1})$	Trend component for the i-th timestamp in a period
$\boldsymbol{r} = (r_0, ..., r_{n-1})$	Residual component of $\boldsymbol{x}$
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	The graph of an IGMRF
$\boldsymbol{Q}$	$n \times n$ precision matrix for an IGMRF of size n
κ	Precision parameter of an IGMRF
d_{max}	Maximum transit duration between two regions
L	History length

Table 8.2: Symbols and notations.

8.3 Finding regions

We aim to divide a city into regions with two goals so that they are useful for high level traffic management: (i) the regions are semantically meaningful, and (ii) the regions have comparable traffic volumes.

For the first goal, we use the map segmentation method in [139] to partition the map of a city based on its road network. For example, the map of Beijing city can be divided into 372 low-level regions as shown in Figure 8.4a. Such regions are bound by the roads and thus naturally capture the division of human activities, making them semantically meaningful. However, the number of low-level regions can be high, making it difficult to monitor all of them. In addition, these regions have highly varying areas and traffic volumes. On one hand, it is not straightforward for city managers to decide how to distribute their work force across the city. On the other hand, it is hard to predict the crowd flows of a tiny region due to the sparsity of data for such a small area. Many small regions are simply roundabounds, making their existence less meaningful. As a result, we propose to further group the low-level regions into bigger high-level regions that have comparable traffic volumes and contain low-level regions with similar crowd flow patterns. To do this, we cluster the region graph as defined below.

Definition 8.3.1 (Region graph) *A region graph is denoted as $G = (V, E, N, W)$, where*

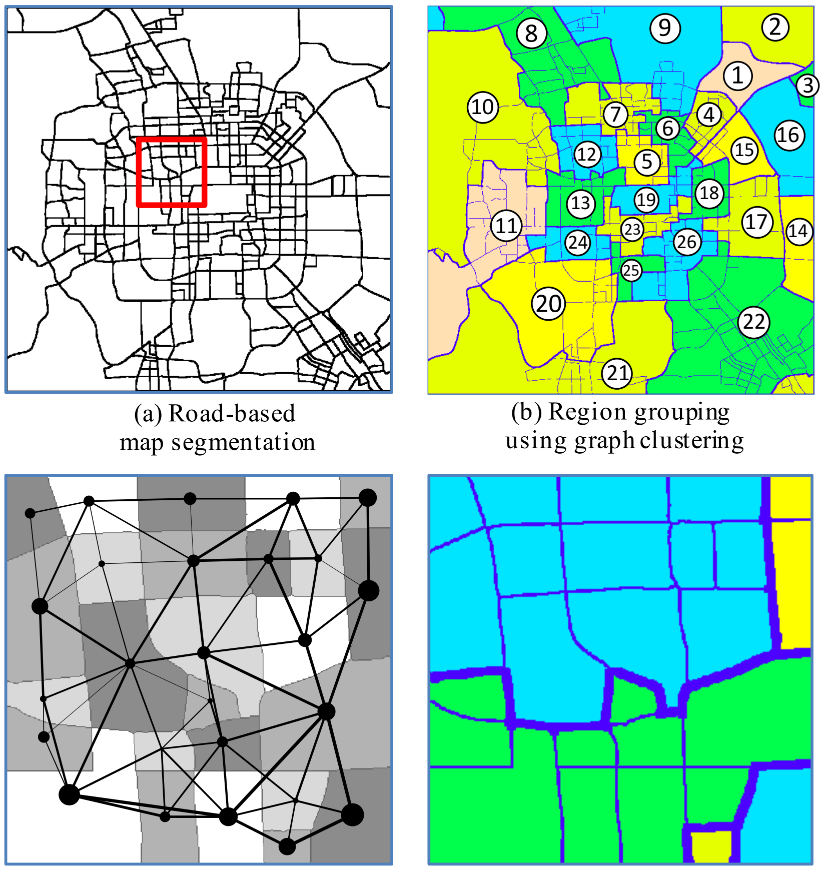
- *Node set $V = \{v_1, v_2, ...\}$ is the set of low-level regions obtained using the map segmentation method in [139]*
- *Edge set $E = \{(v_i, v_j) | v_i$ and v_j are adjacent on the city map$\}$.*
- *Node weights N, where $N_{v_i} = \sum_{t=0}^{n-1} (x_{v_i,t}^{new} + x_{v_i,t}^{end})$ is the sum of crowd flows in region v_i during the historical time period $[0, n-1]$.*
- *Edge weights W, where W_{v_i,v_j} is the similarity of crowd flow patterns between regions v_i and v_j.*

We want to merge low-level regions with similar rise-and-fall crowd flow patterns, i.e., the plots of their crowd flows over time have similar shapes. Thus, we define the edge weight between two low-level regions v_i and v_j as the Spearman's rank correlation coefficient between their crowd flows during a historical time period. Specifically, each region can be represented as a vector $v_i = (x_{v_i,0}^{new}, ..., x_{v_i,n-1}^{new}, x_{v_i,0}^{end}, ..., x_{v_i,n-1}^{end})$. The correlation coefficients among the regions are computed on these vectors.

Figure 8.4c shows a subgraph of the region graph for our Beijing dataset. Here, each node represents a low-level region while its size represents the node weight. There is an edge between two regions if they share a boundary road. The edge widths are proportional to the edge weights.

Next, we cluster the region graph into m high-level regions $\mathcal{R} = \{u_1, u_2, ..., u_m\}$, where each high-level region u_i is a set of adjacent low-level regions, with two goals:

- Edge cut minimization: $\min_{\mathcal{R}} \sum_{\substack{v_i \in u_k; v_j \in u_l \\ u_k \neq u_l}} W_{v_i,v_j}$
- Cluster balancing: $\frac{\sum_{v_i \in u_j} N_{v_i}}{\sum_{v_i \in V} N_{v_i}/m} < 1 + \zeta \; \forall u_j \in \mathcal{R}$, where $\zeta > 0$ is a predefined imbalance factor.



(a) Road-based map segmentation

(b) Region grouping using graph clustering

(c) Low-level region graph for the area in the red box in Fig. 4a

(d) Region grouping for the graph in Fig. 4c

Figure 8.4: Finding regions in Beijing: (a) low-level regions based on city roads, (b) high--level regions based on crowd flow patterns, (c,d) the adjacency graph and region grouping for the red box in Figure 8.4a.

The first goal helps us group highly similar low-level regions together. The second goal constrains the sum of node weights in each cluster to be close to the attainable average. In other words, we want to balance the total traffic volumes among the clusters, which would be helpful for city planning and traffic management.

We use the graph clustering algorithm in [140] to cluster the region graph since it supports our two goals (we set $\zeta = 0.1$). To choose the number of clusters m, we use the elbow method [6] on the edge cut: $m = 26$ for the Beijing dataset and $m = 15$ for the NYC taxi dataset. The NYC bike dataset has only 23 regions, thus there is no need to futher reduce the number of regions. Figure 8.4d shows the resulting high-level regions for the corresponding

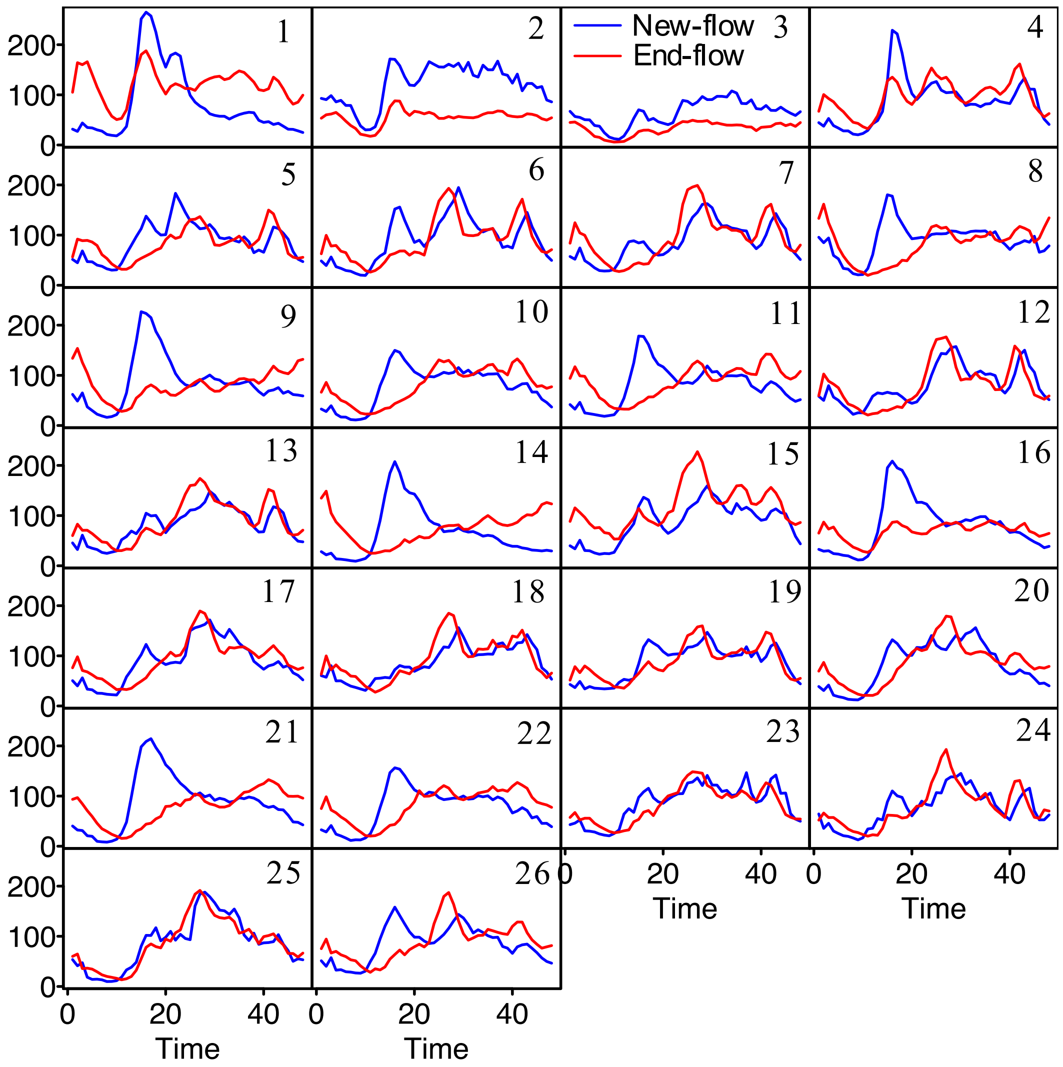


Figure 8.5: Average daily crowd flows of 26 regions in Fig. 5b. The region IDs are in the top right corners of the sub-figures.

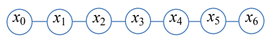
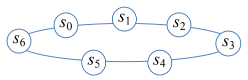
Model	Trend	Seasonal
Graph $\mathcal{G}$		
Probability density π	$\pi(\boldsymbol{x}\|\kappa) \propto \kappa^{(n-1)/2} exp\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}\right)$	$\pi(\boldsymbol{s}\|\kappa) \propto \kappa^{(F-1)/2} exp\left(-\frac{1}{2}\boldsymbol{s}^T\boldsymbol{Q}\boldsymbol{s}\right)$
Precision matrix $\boldsymbol{Q}$	$\kappa \times \begin{pmatrix} 1 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -1 & 2 & -1 & \\ & & & & -1 & 2 & -1 \\ & & & & & -1 & 1 \end{pmatrix}$	$\kappa \times \begin{pmatrix} 2 & -1 & & & & & -1 \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -1 & 2 & -1 & \\ & & & & -1 & 2 & -1 \\ -1 & & & & & -1 & 2 \end{pmatrix}$

Table 8.3: Temporal IGMRF models: $\mathcal{G}$ is the graph ($n = 7$) and $\boldsymbol{Q}$ is the precision matrix; $\kappa \in \mathbb{R}$ is the precision parameter to be learned.

subgraph in Figure 8.4c. The whole high-level region map for Beijing is shown in Figure 8.4b, with the obtained region IDs. Further, Figure 8.5 summarizes the average daily crowd flows of the 26 high-level regions in Figure 8.4b. Clearly, these regions have comparable total traffic volumes. We can also see some distinctive traffic patterns, suggesting that the obtained clusters are meaningful. Neighboring regions may have similar patterns due to their geographhical proximity.

8.4 Temporal models

In this section, we build the seasonal model $\boldsymbol{s}$ and trend model $\boldsymbol{y}$ based on Intrinsic Gaussian Markov Random Fields (IGMRF) by capturing the temporal information in the crowd flows. We first give a brief introduction to IGMRF.

8.4.1 Temporal IGMRF Models

To model a time series $\boldsymbol{x} = (x_0, x_1, ..., x_{n-1})$ over n timestamps, we treat $\boldsymbol{x}$ as a temporal IGMRF, that is, a random vector $\boldsymbol{x}$ having an improper Gaussian density. The IGMRF model fits our city-scale prediction problem well since it is robust to noise and missing data and scalable to big data.

The temporal IGMRF is specified by its precision matrix $\boldsymbol{Q}$ and undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ and $\mathcal{E}$ are the node set and edge set respectively. In essence, the structure of $\mathcal{G}$ visually summarizes the conditional dependence among timestamps, while the value of matrix $\boldsymbol{Q}$ decides the specific probability density $\pi(\boldsymbol{x})$ of the distribution of $\boldsymbol{x}$. The second column in Table 8.3 shows an example IGMRF with its $\mathcal{G}$, $\boldsymbol{Q}$, and $\pi(\boldsymbol{x}|\kappa)$, where κ is a parameter to be learned. In particular, each timestamp is represented by a row and a column in $\boldsymbol{Q}$, as well as a node in $\mathcal{V}$. Non-zero entries in $\boldsymbol{Q}$ correspond to edges between corresponding nodes in $\mathcal{E}$. Each zero entry in $\boldsymbol{Q}$—or equivalently, the absence of an edge in $\mathcal{G}$—signifies that two corresponding timestamps are conditionally independent given the other timestamps. While the general form of $\boldsymbol{Q}$ is given in Table 8.3, we still need to learn the parameter $\kappa \in \mathbb{R}$ from data to set a specific value for $\boldsymbol{Q}$.

The formal definition of IGMRF [141] is given below. First, let us define a symmetric matrix $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ as symmetric positive semi-definite (SPSD) iff $\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} \geq 0 \, \forall \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{x} \neq \boldsymbol{0}$.

Definition 8.4.1 (IGMRF) *Let $\boldsymbol{Q}$ be a SPSD precision matrix with rank $n-k>0$. A random vector $\boldsymbol{x} = (x_0, ..., x_{n-1})$ is an IGMRF of rank $n-k$ with parameters $(\boldsymbol{\mu}, \boldsymbol{Q})$ iff $\boldsymbol{x}$ follows an improper Gaussian distribution, that is, its probability density function $\pi(\boldsymbol{x})$ has the form:*

$$\pi(\boldsymbol{x}) = (2\pi)^{-\frac{n-k}{2}}(|\boldsymbol{Q}|^*)^{1/2}exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{Q}(\boldsymbol{x}-\boldsymbol{\mu})\right) \tag{8.3}$$

Further, $\boldsymbol{x}$ is an IGMRF wrt a labelled graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{0, ..., n-1\}$ and

$$Q_{ij} \neq 0 \Leftrightarrow (i,j) \in \mathcal{E} \, \forall i \neq j$$

Here, $|\boldsymbol{Q}|^*$ denotes the generalized determinant equal to the product of the $n-k$ non-zero eigenvalues of $\boldsymbol{Q}$. The first condition states that $\boldsymbol{x}$ follows an improper Gaussian distribution, the probabilistic nature of which makes the IGMRF robust to noise and missing data. Whereas, the second condition ($Q_{ij} \neq 0$ only if i and j are neighbors in $\mathcal{G}$) is the Markov property that makes $\boldsymbol{Q}$ and $\mathcal{G}$ sparse, hence easier to store and faster to compute as shown in [141]. We next design $\boldsymbol{Q}$ and $\mathcal{G}$ based on the forward differences:

Definition 8.4.2 (Forward differences) *Given a time series $\boldsymbol{x} = (x_0, ..., x_{n-1})$, the first-order forward difference at time t is defined as*

$$\Delta x_t = x_{t+1} - x_t, \quad t = 0, ..., n-2$$

Gaussian assumption: *To make $\boldsymbol{Q}$ and $\mathcal{G}$ sparse, we impose the following assumption on $\boldsymbol{x}$:*

$$\Delta x_t \overset{iid}{\sim} \mathcal{N}(0, \kappa^{-1}), \quad t = 0, ..., n-2 \tag{8.4}$$

where $\kappa \in \mathbb{R}$ is the precision parameter learned from data.

The graphs $\mathcal{G}$ for the Gaussian assumption is shown in the second column of Table 8.3 in case $n = 7$. There is an edge in $\mathcal{G}$ for and only for pairs of consecutive timestamps.

The Gaussian assumption reduces the numbers of edges in $\mathcal{G}$ and non-zero entries in $\boldsymbol{Q}$ to $O(n)$, making them very sparse and our solution scalable. Further, it imposes a smooth change between consecutive timestamps in the time series, making the IGMRF robust to noisy and missing data. To find $\boldsymbol{Q}$, we form the probability density $\pi(\boldsymbol{x})$ as in Equation 8.3:

$$
\begin{aligned}
\pi(\boldsymbol{x}|\kappa) &\propto \kappa^{(n-1)/2} exp\left(-\frac{\kappa}{2}\sum_{t=0}^{n-2}(\Delta x_t)^2\right) \\
&= \kappa^{(n-1)/2} exp\left(-\frac{\kappa}{2}\sum_{t=0}^{n-2}(x_{t+1} - x_t)^2\right) \\
&= \kappa^{(n-1)/2} exp\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}\right) \qquad (8.5)
\end{aligned}
$$

where the $n \times n$ precision matrix $\boldsymbol{Q}$ is shown in the second column of Table 8.3 (zero entries are not shown), and κ is the parameter to be learned.

Learning IGMRF: We learn the IGMRFs using the integrated nested Laplace approximations approach [142]. In essence, we find the parameter κ using maximum a posterior (MAP) estimation given some prior distribution $\pi(\kappa)$ of κ:

$$
\arg\max_{\kappa} \pi(\kappa|\boldsymbol{x}) = \arg\max_{\kappa} \pi(\boldsymbol{x}|\kappa)\pi(\kappa) \qquad (8.6)
$$

Note that the computation of an IGMRF can be sped up using the Cholesky factorization $\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^T$, where $\boldsymbol{L}$ is a lower triangular matrix. With the Gaussian assumption, $\boldsymbol{Q}$ becomes sparse with $O(n)$ non-zero entries, reducing the factorization cost from $O(n^3)$ in general to $O(n)$ [141].

8.4.2 IGMRF Seasonal Models

Gaussian properties of crowd flows: Figure 8.6a shows the histograms and the fitted normal distributions using maximum-likelihood estimation for the forward differences of the square root of new-flow in a region in **BJ** and **BIKE** (see Section 8.6.1 for dataset descriptions). Visually, the fitted normal distributions closely match the histograms. Thus, we can use our Gaussian assumption for **BJ** and **BIKE** to model the square root of new-flow. We note that the square root of flows follow Gaussian distribution but the raw flows do not. We obtain similar results for end-flows and other datasets and report the complete results in Section 8.6.1. Therefore, we propose building an IGMRF seasonal model for the square-root of crowd flows.

Seasonal model: For a periodic time series with period length F, we design an IGMRF $\boldsymbol{s} = (s_0, s_1, ..., s_{F-1})$ as a seasonal model, with an additional assumption on the smooth change between s_{F-1} and s_0. The graph $\mathcal{G}$ thus becomes circular: there is an additional edge between the last timestamp and the first timestamp. Specifically, the circular graph $\mathcal{G}$ is shown in the last column of Table 8.3. To impose the circular property of $\mathcal{G}$, we modify the forward differences for $i = 0, ..., F-1$ as follows:

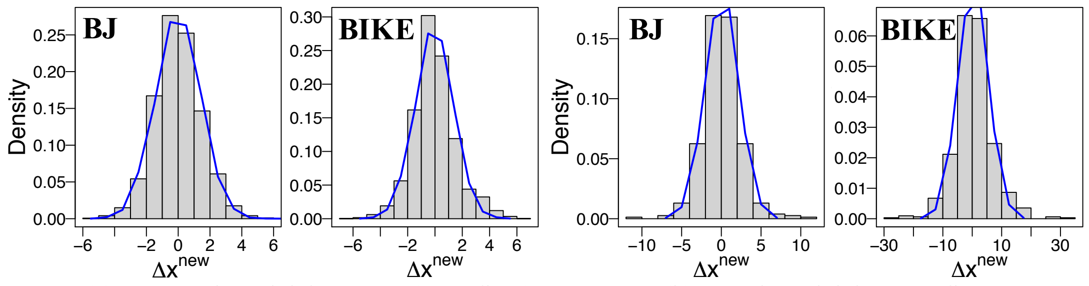
$$\Delta s_i = s_{(i+1) \bmod F} - s_i$$

Here the Gaussian assumption is $\Delta s_i \overset{iid}{\sim} \mathcal{N}(0, \kappa_s^{-1})\ \forall i = 0, ..., F-1$, and $\kappa_s \in \mathbb{R}$ is the only parameter we need to learn. The corresponding circular precision matrix $\boldsymbol{Q}$ can be derived similar to the case without the seasonal assumption, and is given in Table 8.3.

8.4.3 IGMRF Trend Models

Gaussian properties of trends: After the seasonal pattern is removed from the flows, we obtain the raw residual $\boldsymbol{x} - \boldsymbol{s}$. Figure 8.6b shows the histograms and the fitted normal distributions for the forward differences of these residuals for the same flows in Figure 8.6a. Again, it is clear that Gaussian distributions can be used to approximate these histograms. We have the same observations for all flow types and datasets, as statistically shown in Section 8.6.1. Thus, we can build an IGMRF trend model as follows.

Trend model: As discussed in Section 8.2.3, we propose adding a trend $\boldsymbol{y_i}$ to capture the change over time in the seasonal pattern of the i-th timestamp in a period. In particular, we want to model a time series $\boldsymbol{y_i} = (y_{i0}, y_{i1}, ..., y_{i,n_y-1})$, where n_y is the number of periods. The temporal IGMRF model in Section 8.4.1 can be used directly for this purpose.



(a) Seasonal model for SQRT(new-flow) (b) Trend model for new-flow
Figure 8.6: Gaussian properties of new-flow of one region.

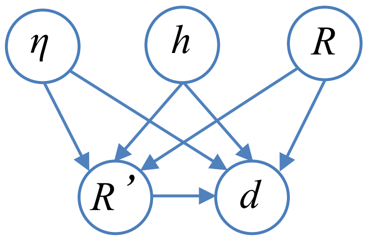
8.5 Spatio-temporal residual model

In this section, we utilize the (intra-region and inter-region) dependencies among different flows and weather information to predict the residuals $\boldsymbol{r} = \boldsymbol{x} - \boldsymbol{s} - \boldsymbol{y}$. We denote $r^{\theta}_{u,t}$ as the residual flow of type θ at time t in region u, where $\theta \in \{new, end\}$. We design our final spatio-temporal residual model to predict $r^{\theta}_{u,t}$ as a regression problem:

$$r^{\theta}_{u,t} = \boldsymbol{\alpha}^{T}_{u,\theta}\boldsymbol{\delta}_{u,t} + \boldsymbol{\beta}^{T}_{u,\theta}\boldsymbol{z}_{u,t} + \phi^{\theta}_{u,w_t} + \sigma^{\theta}_{u,h_t} + \gamma^{\theta}_{u} \tag{8.7}$$

where the inputs are:

- $\boldsymbol{\delta}_{u,t}$: Transit features of region u at time t, capturing the *inter-region dependence* among flows (Section 8.5.1).



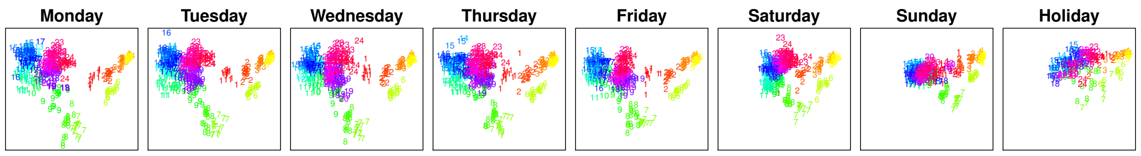


Figure 8.7: (a) Bayesian network transit model: At hour h of day type η, an object takes d timestamps to move from region R to region R' on average. (b) Mapping timestamps into 2D-space based on transit patterns using PARAFAC. Each point is for one hour in a day.

- $\boldsymbol{z}_{u,t} = (r^{new}_{u,t-i}, r^{end}_{u,t-i} | i = 1, 2, ..., L)^T$: Historical residual flows of region u at time t, capturing *intra-region dependence* among different flow types, where L is a chosen history length.

- w_t: The weather condition w_t at time t.

Model parameters (for region u and flow type θ): $\boldsymbol{\alpha}_{u,\theta}$ and $\boldsymbol{\beta}_{u,\theta}$ are coefficient vectors for $\boldsymbol{\delta}_{u,t}$ and $\boldsymbol{z}_{u,t}$; ϕ^{θ}_{u,w_t} is a coefficient for weather condition w_t; σ^{θ}_{u,h_t} is a coefficient for the hour in day h_t of time t; and γ^{θ}_{u} is an intercept.

Other factors: While we only consider weather and public holidays here, if more data is available (e.g., local social events in each region, traffic accidents, or traffic jams), we can easily and similarly include it in the residual model $\boldsymbol{r}$.

We next explain in detail how to capture the inter-region dependence and the effects of weather and holidays.

8.5.1 Capturing Inter-region Dependence

Bayesian Network Transit Model

The flows of neighboring regions can affect each other due to the transition of objects among them. This inter-region dependence can be naturally captured by a Bayesian network transit model as shown in Figure 8.7(a). For an individual that is moving between regions, when s/he gets out of a region R, we predict the next region R' s/he will move to and the time duration d that s/he will take to complete the transition. We assume that the next region R' depends on the current region R, different hours in a day h, and different types of days η. Similarly, the transit duration d is dependent on R, h, η as well as R', where she is traveling to. Hour of day h can take any integer value between 1 and 24. Days with similar transit patterns are grouped into one day type. We discuss how day types η are determined in Section 8.5.1.

Our Bayesian Network can be learned easily and fast by counting since it has known structure and full observability (R, R', h, and η are all pre-defined). Denote h_t and η_t as the hour in day and the day type for timestamp t respectively. Once we have learned the conditional probability functions $p(R'|R, h_t, \eta_t)$ and $p(d|R', R, h_t, \eta_t)$, we can compute the probability that an individual getting out from R at time t will transit to R' after d timestamps as:

$$g_{R,R',t,d} = p(R'|R, h_t, \eta_t) \times p(d|R', R, h_t, \eta_t) \tag{8.8}$$

Transit Features

To capture the influences among regions with regard to their deviations from the expected flows $(\boldsymbol{s} + \boldsymbol{y})$, we apply g from Equation 8.8 to the residual flow $\boldsymbol{r} = \boldsymbol{x} - \boldsymbol{s} - \boldsymbol{y}$. Here, we assume that the transition probabilities g are the same for both the temporal components $(\boldsymbol{s} + \boldsymbol{t})$ and the residual component $\boldsymbol{r}$. While this is a strong assumption, it eliminates the need to model the flows between every pair of regions, which is costly and more susceptible to noise.

With this assumption, we define the ***transit features*** in Equation 8.7 as $\boldsymbol{\delta}_{u,t} = (\delta_{u,t}^{new}, \delta_{u,t}^{end})^T$, where $\delta_{u,t}^{new}$ is the sum of crowd flows from other regions to u, and $\delta_{u,t}^{end}$ is the sum of crowd flows from u to other regions in the last d_{max} timestamps:

$$\delta_{u,t}^{new} = \sum_{d=1}^{d_{max}} \sum_{v \in \mathcal{R}} (r_{u,t-d}^{new} \times g_{u,v,t-d,d}) \tag{8.9}$$

$$\delta_{u,t}^{end} = \sum_{d=1}^{d_{max}} \sum_{v \in \mathcal{R}} (r_{v,t-d}^{new} \times g_{v,u,t-d,d}) \tag{8.10}$$

We define a maximum transit duration d_{max} since most transitions take a bounded amount of time in real life.

Choice of Day Type η

To decide the day type η, we group days with similar transit patterns using a data-driven approach. We denote $\mathcal{Y} \in \mathbb{N}^{n \times m \times m \times d_{max}}$ as the (timestamp, source region, destination region, transit duration) tensor, where $\mathcal{Y}_{tuvd}$ is the number of trajectories that leave region u at timestamp t, and arrive at region v at timestamps $t + d$. To compare the transit patterns at two timestamps t_1 and t_2, we compare the two sub-tensors $\mathcal{Y}_{t_1 \dots}$ and $\mathcal{Y}_{t_2 \dots}$, which are high-dimensional and sparse. To avoid the curse of dimensionality, we perform dimension reduction using PARAFAC tensor factorization [5]. In particular, $\mathcal{Y}$ can be factorized into four matrices $\boldsymbol{M} \in \mathbb{R}^{n \times q}$, $\boldsymbol{H} \in \mathbb{R}^{m \times q}$, $\boldsymbol{I} \in \mathbb{R}^{m \times q}$, and $\boldsymbol{J} \in \mathbb{R}^{d_{max} \times q}$, where q is the number of lower dimensions, such that:

$$\mathcal{Y}_{tuvd} = \sum_{i=1}^{q} M_{ti} H_{ui} I_{vi} J_{di} \tag{8.11}$$

Matrix $\boldsymbol{M}$ is the low-dimension representation of the timestamps (each row is one timestamp, and each column is one dimension). We can use this low-dimension representation to compare the transit patterns at different timestamps.

Figure 8.7(b) shows the mapping result of timestamps into a 2D-space ($q = 2$) for dataset

BJ-B (Section 8.6.1). Timestamps are separated into different weekdays and holidays. Each point in a subfigure is for one hour in a day (marked as 1 to 24). From these figures, we can visually put the daily transit patterns into five groups: (i) Monday to Friday, (ii) Saturday, (iii) Sunday, and (iv) holidays. We thus use these four day types for **BJ-B**. Similar results are obtained for other datasets but not shown due to space limitation.

8.5.2 Effects of Weather and Holidays

Weather and holidays can affect crowd flows. For example, Figure 8.8a shows that thunderstorms may increase the use of taxis while Figure 8.8b shows that heavy rain may reduce crowd flows at a region compared to its seasonal pattern. Figure 8.8b also shows that crowd flows during a holiday can be significantly different from the flows during normal days.

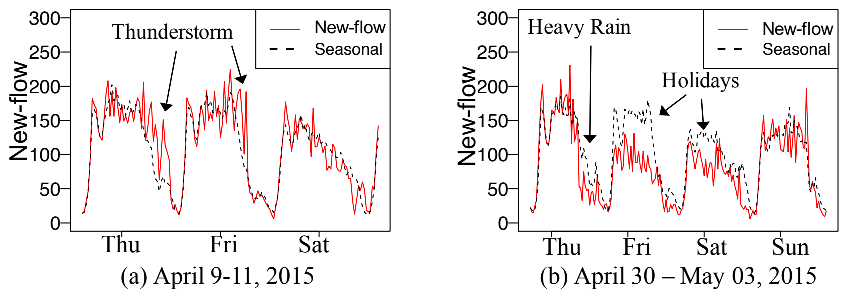


Figure 8.8: Effects of weather and holiday in region 2 in Beijing.

We include weather and holidays into our models in two ways. First, we build a separate seasonal model for holidays with the period of a day ($F_{holiday} = 1$ day). Second, we add a coefficient ϕ^{θ}_{u,w_t} for each region u, flow type θ, and weather condition w_t at time t in the regression (Equation 8.7).

8.6 Experiments

8.6.1 Settings

Datasets: We use three different sets of data as summarized in Table 8.4. Each dataset contains three sub-datasets: trajectory, region map, and weather data, as detailed below.

BJ: The trajectory data is taxi GPS data for Beijing in 2015. We categorize weather data into good weather (sunny, cloudy) and bad weather (rainy, storm, dusty). With our clustering framework in Section 8.3, we partition Beijing into 26 high-level regions (Figure 8.4a). Using Definition 8.2.1, we obtain two types of crowd flows. Data for 350 timestamps were missing for all regions due to system glitches. We choose data from the last three weeks as testing data, and all data before that as training data.

NYC: We partition NYC into 15 high-level regions using its road network and traffic data. The trajectory data is generated by taxis in NYC in 2013. Trip data includes: taxi ID, pick-up and drop-off locations and times. The new-flow and end-flow are thus the number of pick-ups and drop-offs in a region respectively. Weather conditions include good weather (sunny, or no available data) and bad weather (foggy, rainy, snowy). We pick the last-week data for testing, and data before that for training.

BIKE: The trajectory data is taken from the New York City bike system in 2014. Trip data includes: trip duration, start and end station IDs, start and end times. Following [143], we group bike stations into clusters using their bipartitie clustering method, and treat each obtained station cluster as a region, instead of using our own clustering framework in Section 8.3. For each region, the new-flow is the number of checked-out bikes, and the end-flow is the number of checked-in bikes. We use data from Apr. 1^{st} to Sep. 10^{th} for training and Sep. 10^{th} to 30^{th} for testing.

Gaussian properties of crowd flows: To statistically verify if a data sample follows a Gaussian distribution, we perform the Kolmogorov-Smirnov test (KS-test). If the KS-test returns a p-

Dataset	**BJ**	**NYC**	**BIKE**
Data type	Taxi GPS	Taxi pickup	Bike rent
Location	Beijing	New York	New York
Start time	3/1/2015	1/1/2013	4/1/2014
End time	6/28/2015	9/8/2013	9/30/2014
#holidays	10	20	9
Timestamp bin size	30 minutes	1 hour	1 hour
	Trajectory data		
#taxis/bikes 34K	33.6K	16K	6.8K
#trips/records	11.7M	206.6M	5.4M
#effective timestamps	2,753	5,880	4,392
#missing timestamps	350	0	0
	Region map data		
#roads/bike stations 193,663	193,663	32,210	344
#low-level regions	372	215	-
#high-level regions	26	15	23
	Weather data		
#good-weather timestamps	3,812	4,081	5,398
#bad-weather timestamps	1,208	358	448
#missing timestamps	486	53	34
Temperature	[-7,34]°C	[32,90]°F	[10,97]°F

Table 8.4: Datasets (holidays include adjacent weekends).

Dataset	**BJ**		**NYC**		**BIKE**	
	Seasonal	Trend	Seasonal	Trend	Seasonal	Trend
new-flow	0.96	1	0.93	1	1	1
end-flow	1	1	1	1	1	1

Table 8.5: Proportions of regions whose crowd flows satisfy the Gaussian assumption.

value greater than 0.01, the Gaussian hypothesis is acceptable. Table 8.5 reports the proportion of regions whose crowd flows follow Gaussian distributions for each dataset. Note that seasonal models are tested on the square root of flows, while trend models are tested on the residual of the corresponding seasonal models. We emphasize that, for the seasonal models, the Gaussian assumptions are poorly satisfied by the raw crowd flows.

Parameter settings: Each timestamp corresponds to 1 hour for **NYC** and **BIKE**, and 30 minutes for **BJ**. Due to the differences in the crowd flows between weekdays and weekends, and between different weekdays (e.g., see Figure 8.2), we choose $F = 1$ week for the seasonal model $\boldsymbol{s}^w$ during normal days, and $F = 1$ day for the seasonal model $\boldsymbol{s}^h$ during holidays. Log-gamma prior is used for the precision parameter κ in Equation 8.6 as suggested in [141].

Evaluation metric: For evaluation, we use the Root Mean Squared Error (RMSE), as defined below:

$$RMSE(\theta) = \frac{1}{n}\sum_{t=1}^{n}\sqrt{\frac{1}{m}\sum_{u=1}^{m}(x_{u,t}^{\theta} - \hat{x}_{u,t}^{\theta})^2} \tag{8.12}$$

where n is the number of regions, m is the number of timestamps, $x_{u,t}^{\theta}$ and $\hat{x}_{u,t}^{\theta}$ are respectively the true and predicted values of flow type θ in region u at time t.

Experiments are run on a Debian machine with Intel i7, 3.50GHz CPU and 15GB RAM. The IGMRF models are implemented using the R-inla package [142].

Baselines: Table 8.6 lists all compared methods. **FCCF** is our final spatio-temporal model. **SARIMA** is the seasonal ARIMA model, using only temporal data. We choose the best parameters for the SARIMA models using the "forecast" package in R language [144]. **LmNei** is a naive linear regression spatio-temporal model. **VAR** (vector auto-regressive model) and **STARMA** (space-time auto-regressive moving average model) are more advanced spatio-temporal models. **VAR** captures the pairwise relationships among all flows, and has heavy computational costs due to the large number of parameters. **STARMA** [145] has fewer parameters thanks to the spatial constraints, but requires an ad hoc definition of the weight matrices capturing the relationships among flows from the same or neighboring regions. **HP-BC-MSI** [143], the state-of-the-art prediction framework for bike-sharing systems, is the most similar to our problem. **HP-BC-MSI** predicts the in/new-flows for clusters of bike stations instead of the noisy individual station flows. Further, it first predicts the aggregated flow for the whole city, and then distributes this flow into each cluster (hierarchical prediction). However, it does not decompose the flows into three components as we do. To compare with **HP-BC-MSI**, we also predict the flows for the same clusters of stations by treating them as high-level regions. Finally, we break down our spatio-temporal model to investigate the contributions of each component.

8.6.2 Results

Complete framework: Table 8.7 shows the RMSE of all methods. Our complete framework consistently and significantly outperforms all baselines. Specifically, **FCCF** is 22% to 52% better than **LmNei**, 25% to 50% better than **SARIMA**, 10% to 30% better than **VAR**, and 27% to 70% better than **STARMA**. **VAR** exploits the relationship among flows and is clearly better than other baseline methods. While both **LmNei** and **STARMA** use spatial information, they are far worse than **VAR**, and even worse than **SARIMA**, suggesting that the ad hoc assignments of the weight matrices in **STARMA** or the naive way of incorprating spatial information in **LmNei** can actually hurt performance. Moreover, this observation also hints that the prediction of a future crowd flow depends heavily on its own history. Finally, **FCCF** decreases the error by 26% for new-flow and 37% for end-flow in **BIKE** compared to **HP-BC-MSI**, the state of the art for prediction in bike-sharing systems, showing the clear benefits of our decomposing flows into three components.

Temporal components: As seen in Table 8.7, the seasonal model **SH** that considers holidays is clearly better than the one without holidays (**S**). The accuracy is further increased when

Method	**Description**
Temporal models	
SARIMA	Seasonal ARIMA model, frequency = 24.
S	$\boldsymbol{x} = \boldsymbol{s}^w$, weekly seasonal model, where $\boldsymbol{s}^w$ follows Section 8.4.2 with $F = 168$.
SH	$\boldsymbol{x} = \boldsymbol{s} = \boldsymbol{s}^w + \boldsymbol{s}^h$, where $\boldsymbol{s}^w$ is a seasonal model ($F = 128$) for normal days, $\boldsymbol{s}^h$ is a daily seasonal model ($F = 24$) for holidays (Section 8.4.2).
SHT	$\boldsymbol{x} = \boldsymbol{s} + \boldsymbol{y}$, **SH** and trend model.
Spatio-temporal models	
LmNei	Linear regression of the historical flows of a region and its neighbors, as well as weather information.
VAR(p)	Vector Auto-Regressive model with lag p.
STARMA(p,q)	Space Time Auto-Regressive Moving Average model [145], p and q are the AR and MA lags.
HP-BC-MSI	(**BIKE** only [143]) Hierarchical prediction + bipartite clustering + multi-similarity-based inference.
SHT+intra	$\boldsymbol{x} = \boldsymbol{s} + \boldsymbol{y} + \boldsymbol{r}$; $\boldsymbol{r}$ follows Equation 8.7 without transit features $\boldsymbol{\alpha}^T \boldsymbol{\delta}_{u,i}$ and weather ϕ^θ_{u,w_t}.
FCCFnoWea	$\boldsymbol{x} = \boldsymbol{s} + \boldsymbol{y} + \boldsymbol{r}$; $\boldsymbol{r}$ follows Equation 8.7, without weather ϕ^θ_{u,w_t}.
FCCF	$\boldsymbol{x} = \boldsymbol{s} + \boldsymbol{y} + \boldsymbol{r}$; $\boldsymbol{r}$ follows Equation 8.7.

Table 8.6: Baselines

trend is added (**SHT**). Our seasonal models **S** and **SH** are better than **LmNei** and **SARIMA** for the **BJ** and **BIKE** datasets, while worse for the **NYC** dataset. This is possibly due to the different levels of noise in different datasets. Specifically, the region crowd flows in the **BJ** and **BIKE** datasets are significantly smaller than those in **BIKE**, leading to noisier data. Thanks to its probabilistic nature, our IGMRF models are robust to noise, and thus give better prediction in the two more noisy datasets.

Spatial-temporal components: **SHT+intra** combines the intra-region dependence into **SHT**, leading to an outstanding improvement in accuracy. The addition of transit features (**FCCFnoWea**) further reduces RMSE, which is more significant for **BIKE** than for the other three datasets, since bike trips are generally longer than taxi trips, and often take more than one 1-hour timestamp to complete. In other words, the bigger the ratio between the average

Model	**BJ**		**NYC**		**BIKE**	
	New	End	New	End	New	End
SARIMA	21.20	18.85	132.82	142.27	20.50	19.38
lmNei	19.17	18.18	154.38	146.92	22.26	20.62
STARMA(3,1)	42.46	19.57	287.34	161.20	26.94	21.01
VAR(5)	15.83	15.83	106.81	101.32	15.36	13.05
HP-MSI-BC					14.70	15.60
S	17.54	16.38	192.77	190.56	18.27	18.09
SH	17.18	16.34	159.90	155.10	17.56	17.33
SHT	16.60	15.80	156.06	153.84	15.24	14.84
SHTIntra	14.63	14.28	89.04	84.84	11.55	10.92
FCFCNoWea	14.19	**14.14**	87.93	84.45	10.83	9.83
FCFC	**14.17**	**14.14**	**87.18**	**83.89**	**10.79**	**9.80**

Table 8.7: RMSE.

trip duration and the timestamp duration, the bigger the impact of inter-region dependence for short-term flow predictions.

Weather effect: The addition of weather (**FCCF**) improves the accuracy for all datasets except the end-flow of **BJ**, possibly due to the high number of timestamps with missing weather data in **BJ**.

Multi-step-ahead prediction: To predict the crowd flows for multiple steps ahead, we change the left hand side of Equation 8.7 from $r^{\theta}_{u,t}$ to $r^{\theta}_{u,t+\Delta}$, for $\Delta \in \{1, 2, 3, 4\}$. Figure 8.9(a) shows the results. Clearly, the farther in the future, the harder the prediction and the higher the error.

Training period: Figure 8.9(b) shows the errors as we vary the length of the training period from 1 month to all available months (4 months for **BJ**, 8+ months for **NYC**, and 5 months for **BIKE**). While more training data generally leads to higher accuracy, the addition of more training data after 3 months does not improve the results significantly.

Missing data: We evaluate the robustness of our framework against missing and noisy data by making predictions when a proportion (20%, 50%, and 70%) of the timestamps is randomly removed from the training datasets for all regions. As shown in Figure 8.9(c), even when 50% of the training data is missing, the performance of **FCCF** is still exceptionally good. Specifically, **FCCF** with 70% missing data is still better than **VAR** with complete data, and significantly outperforms **SARIMA**, **LmNei**, **STARMA**, and **HP-BC-MSI** with complete data (as shown in Table 8.7).

Efficiency: Figure 8.9(d) shows the running times without any parallelization. As can be seen, the total time for offline training is less than 10 minutes for all three datasets. More importantly, online prediction takes less than 1 minute for all datasets, showing that our framework is practical for real-time prediction of citywide crowd flows. In practice, we can train the temporal models $\boldsymbol{x}$ and $\boldsymbol{y}$ in parallel for all flows.

8.6.3 Case Studies

Figure 8.10a shows the region map for lower Manhattan and Brooklyn, New York, for **BIKE** in 2014. To show that our framework can capture the sudden deviations of crowd flows from their usual patterns, we investigate two anomalous case studies: suddenly decreased and suddenly increased flows.

During a rainy day (Figure 8.10b, region $R9$, Sept. 13^{th}, 2014), the flows of bikers were significantly reduced. From 1pm to 3pm, the weather turned from sunny to foggy with strong wind, making people wary of traveling by bike. Thus, the true end-flow (red solid line) into region $R9$ became smaller than its seasonal pattern (**SHT**, brown circles). From 4pm, it started to rain, leading to a big decrease of flows and deviation from **SHT**. By including the recent history of $R9$, **SHT+intra** (blue empty circles) better tracks the true flow but is still far from

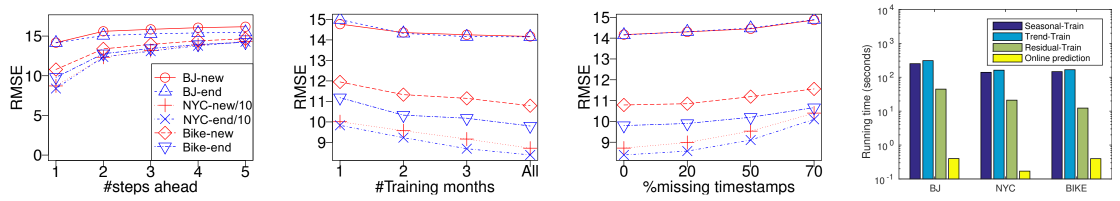
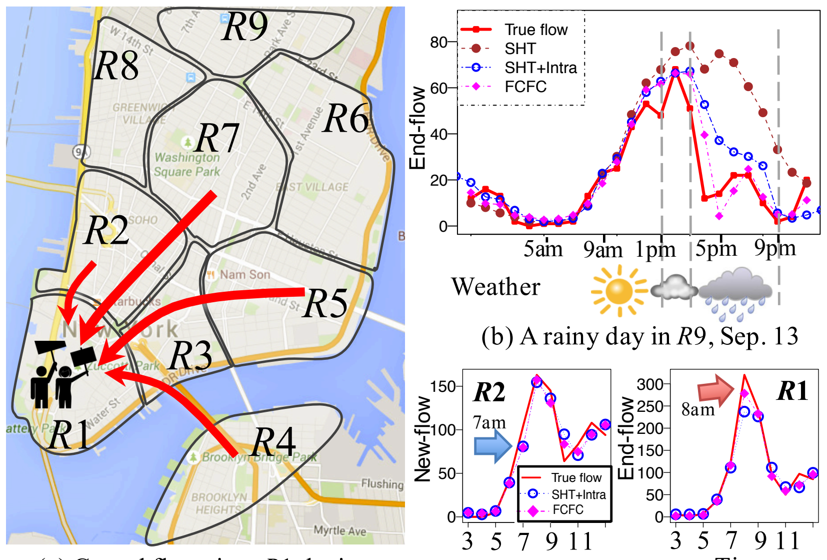


Figure 8.9: **FCFC**: (a) Multi-step ahead prediction; Effects of (b) fewer training data and (c) randomly missing data. (We plot RMSE/10 for NYC dataset for clearer figures). (d) Running time for training and predicting.

the truth. **FCCF** (diamonds) further notices the reduction of crowd flows from other regions into $R9$ due to bad weather conditions, and improves the prediction significantly compared to **SHT+intra**.

On Sept. 17^{th}, 2014, an enormous flow of people traveled to Zuccotti Park in region $R1$ to celebrate the three-year anniversary of Occupy Wall Street protest at 8am. Figure 8.10a shows the large crowd flows from different origins traveling to $R1$ before 8am. As a result, the end-flow of $R1$ at 8am, as well as the new-flows of these origins (e.g., $R2$, $R3$, $R5$, $R7$) during the previous hours were anomalously higher. For example, an increase in the new-flow of region $R2$ at 7am (Figure 8.10c, red solid line) led to a later increase in end-flow of $R1$ at 8am (Figure 8.10d, red solid line), as annotated by the arrows. Model **SHT+intra** (blue circles), which only considers the history of region $R1$, fails to predict this sudden increase at 8am, as pointed out by the red arrow in Figure 8.10d. Whereas, **FCCF** (diamonds) captures the sudden increase in new-flows from other regions in the previous hours and thus much better tracks the ground truth of $R1$. Note that there is still room for improvement; for example, if we had known that Occupy Wall Street would happen, we might have included one more coefficient



(a) Crowd flows into $R1$ during Occupy Wall Street (8am, Sep. 17)
(b) A rainy day in $R9$, Sep. 13
(c) $R2$, Sep. 17
(d) $R1$, Sep. 17

Figure 8.10: Crowd flow prediction for **BIKE**, in NYC, 2014.

for local social events in the residual model $\boldsymbol{r}$ to further improve prediction.

8.7 Related Work

Human Mobility Prediction: Prior research [136, 146–148] has been done to predict an individual's movement based on their location history, in order to enable context-aware computing that can facilitate the individual's daily life, such as suggesting driving directions, pushing promotion coupons, or predicting human mobility under disaster scenarios. Unlike such research, we forecast the aggregated crowd flows in a region rather than millions of individuals' mobility traces. The latter is very difficult, computationally expensive, and not necessary for the application scenario of public safety.

Traffic Condition Prediction on Roads: Another branch of research has been conducted to predict travel speeds and traffic volume on road networks. The majority of such research [149–152] focuses on the prediction on a single (or a few) road segment(s), rather than a city-scale prediction. A number of works also use Bayesian network approach [153, 154] and Markov Random Fields [155] for road traffic forecasting. Some recent studies [137, 138] try to scale up the prediction throughout an entire city, with a diversity of models, such as matrix factorization and tensor decomposition. [156] presents research on developing models that forecast traffic flow and congestion in a deployed traffic forecasting service.

Our method differs from the above problems in the following ways. First, we study the crowd flows in a region rather than traffic conditions on a road segment. The region-based flows provide a macro-level view of city traffic, which is important not only for traffic management but also for public safety. The four types of flows we consider are only meaningful within a region setting. In addition, people can cross regions without being constrained by road networks, for example, by walking or subway systems. Second, given the four types of flows, our problem becomes more difficult, as there are dependencies between different types of flows in a region and dependencies between flows of different regions. Third, we cluster regions into groups based on flow patterns, predicting the crowd flows in a cluster. The latter can deal with the data sparsity and help adjust the flow in each individual region belonging to the cluster. That is, we have flow prediction at both fine and coarse granularities.

Urban Computing: Recently, the proliferation of big data in cities has fostered new research on urban computing [157], which aims to tackle urban challenges (such as traffic congestion and air pollution) by using data science and computing technology. A branch of research also partitions a city by major roads [139], and then studies the traffic flow between regions, for example, detecting traffic anomalies [158] and problematic urban design [159], or understanding the latent function of a region [160]. Our research is also a step towards urban computing, but different in terms of problem setting. To the best of our knowledge, in the field of urban computing, forecasting crowd flows has never been done at the scale of a city and in a data-driven way.

8.8 Conclusion

In this paper, we propose predicting the flows of crowds in a city using big data, which is strategically important for traffic management and public safety. We propose a scalable prediction framework that exploits multiple complex factors affecting the crowds and decomposes crowd flows into three components: seasonal, trend, and residual flows. Thanks to the IGMRF models and the cluster-based adjustment, our framework is robust to both noise and missing data. Experiments show that our approach is scalable and outperforms baselines significantly.

While we treat each type of trajectory data separately in our experiments due to the restriction of available data, the crowd flows can be measured as an aggregation of all types of trajectories if available (e.g., phone signals, GPS data, and subway card swiping data). Our framework still applies to such cases as is. Last but not least, if more information is available (e.g., local social events, traffic jams, or traffic accidents in each region), it can be easily incorporated into our residual model $\boldsymbol{r}$ to further improve prediction accuracy.

8.9 Acknowledgement

The work was supported by the National Natural Science Foundation of China (Grant No. 61672399 and No. U1401258), the China National Basic Research Program (973 Program, No. 2015CB352400), and the US NSF grant IIS-1219254.

Chapter 9

GPOP: Scalable Group-level Popularity Prediction for Online Content in Social Networks

9.1 Introduction

With the enormous amount of data generated on the Internet today, predicting the popularity of online content, such as videos, news articles, or posts on social networks, is increasingly important in many different applications. In particular, *given the set of users who have reacted (i.e., commented, liked, shared or retweeted) to a content from time t_0 (when it was created) to time t_1, can we predict how many and which users will react to it until time $t_2 > t_1$?* If this question can be efficiently answered, we can filter information for users to cope with data overload, or prefetch web content to reduce latency and improve user experience. We can also design more effective ad campaigns to increase product popularity and maximize profit.

Several studies have focused on predicting the popularity of various online network-contents, and they can be generally grouped into two categories: (i) *user-level popularity* [3, 161–164] that predicts (at a low level) which users will react to a content; (ii) *population-level popularity* [4, 165–168] that predicts (at a high level) how many users in total will react to a content. While each approach is reasonable to use in certain situations, we claim that a *group-level popularity* approach, which predicts the popularity within user groups, is more practical given the noise and the intrinsic heterogeneity in the network data. The user-level information cascades, on one hand, are often susceptible to missing data, sensitive to users' emotions, and also often costly to learn [169]. The population-level popularity, on the other hand, is only able to provide a very coarse view, losing most essential information on user behaviors, and thus lacks flexibility in tailoring information for different users' interests. We observe that in many social networks (specifically Twitter.com and Behance.net in this paper) that users naturally organize themselves into groups, reflecting their interests, communities or locations. Within a group, users are fairly consistent in how they react to content. Thus, group-level prediction provides a great trade-off between the cost in model learning and prediction quality. Compared to the user-level, a group-level popularity is much less noisy and more compact, while it is more detailed and cohesive than that at the population level. In addition, a group-level incurs a significantly smaller computational cost than the user-level predictions.

Example 1. Behance.net is a social network where users share their creative projects for others to see. The popularity of a project at a timestamp t can be defined as the total number of users who have pressed the "appreciate" button on this project. In Fig. 9.1a, we show the number of new reactions (appreciations) at each timestamp (a period of 4 hours each) for one project, while Fig. 9.1b shows its cumulative form. Each color stripe corresponds to one cohesive group of users (based on our later proposed solution). It can be seen that most users who appreciated this project are from one group corresponding to the yellow stripe in Fig. 9.1a-b, suggesting that the interests and behaviors of users are similar within each group and different across different groups. Finally, given the observation over this project from time 0 to 18 (first 3 days), we predict its popularity from time 19 to 60 (the next 7 days). Fig. 9.1c-e

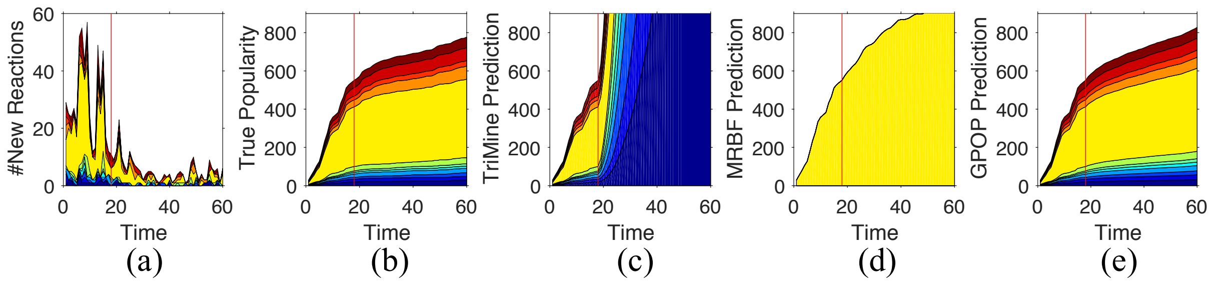


Figure 9.1: An example project from Behance.net where each color stripe represents one user group: (a) Number of new reactions per 4-hour time periods; (b) Cumulative number of reactions; Predictions of popularity made at time $t_1 = 18$ (marked by the vertical red line) for (c) user level [3] (then aggregated by groups), (d) population level [4], and (e) group level (our solution).

respectively show the prediction results of user-level [3], population-level [4], and group-level popularity. Clearly, our group-level solution makes the best predictions for both the number of appreciating users in total and within each individual group.

We develop in this paper a novel framework for predicting the group-level of online content: *Given the set of users reacting to a content from time t_0 to time t_1, how many users in each group will react to that content until time $t_2 > t_1$?* We first group users into robust and cohesive clusters, and then perform tensor decomposition coupled with a hierarchical structure among groups to make predictions. Improvement in either of these two steps will lead to an improvement in the overall prediction accuracy. The proposed framework not only ensures scalability in dealing with large-scale social networks but also promises a high prediction accuracy. In order to minimize the impact of noise and be more flexible in capturing the changes in user interests, we exploit both the network topology among users and their interaction activities in learning a robust partition over all users. The PARAFAC tensor decomposition is further adapted to work with the hierarchical constraint over user groups, and we show that optimizing this constrained function via gradient descent achieves faster convergence and leads to more stable solution, as compared to other matrix factorizations. Here are our contributions:

- We propose to combine users' historical activities and the network structure into a network-constrained popularity graph. By clustering this graph, we put users into robust and meaningful groups that capture the evolution of online content's popularity over time.

- We add a novel hierarchical constraint to coupled tensor decomposition in order to simultaneously predict popularity at the group and population levels. We further prune data using top-k similarity queries to improve accuracy and reduce computational cost.
- We evaluate our framework, which we name GPOP (Group-level POpularity Prediction), on two real-world datasets collected from Twitter and Behance social networks: GPOP scales linearly with its parameters, and outperforms all baseline methods significantly in terms of accuracy. Our code and data are available online[1].

9.2 Problem definitions

Let us denote $G = (V, E)$ a network where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes, each representing a user, and $E \subseteq V \times V$ is the set of edges representing the (undirected) connections among users. Let p_i be a content (e.g. a hashtag in Twitter) being broadcast in the network.

Definition 9.2.1 (User-level popularity) *A user-level popularity of a content p_i at time t is defined by the vector $s_{it} = (\mathcal{S}_{it1}, \ldots, \mathcal{S}_{itn})$, where $\mathcal{S}_{itj} \in [0, +\infty)$ is the number of times user v_j has reacted to content p_i after the first t timestamps since p_i was created. We call $\mathcal{S}_{itj}$ the state of user v_j at timestamp t w.r.t. content p_i.*

In Definition 9.2.1, the popularity is a *non-decreasing quantity* over time: $\mathcal{S}_{itj} \leq \mathcal{S}_{it'j}$ $\forall i, j$ and $t' > t$. If users are no longer interested in that content, its popularity will stay the same. We define the popularity to be the cumulative number of reactions instead of the number of new reactions at each timestamp since the latter in practice is very noisy, as visualized in Fig. 9.1a. Additionally, all timestamps are *relative* to the creation time of each content.

[1]Code and data: `http://cs.ucsb.edu/~mhoang/gpop.tar.gz`

Problem 9.2.1 (User Clustering) *Given a network G, a set of contents $P = \{p_1, p_2, \ldots, p_m\}$, and the number of clusters l, find a partition of the users $C = \{C_1, C_2, \ldots, C_l\}$ that reflects the spread of the popularity of contents in P, where $C_j \cap C_{j'} = \emptyset$, $\cup_{j=1}^{l} C_j = V$.*

Definition 9.2.2 (Group-level popularity) *Given C as an optimal solution for Problem 9.2.1, the group-level popularity of a content p_i at timestamp t is the vector $x_{it} = (\mathcal{X}_{it1}, \ldots, \mathcal{X}_{itl})$, where $\mathcal{X}_{itj}$ is the total number of times users in group C_j have reacted to p_i after the first t timestamps since p_i was created, that is, $\mathcal{X}_{itj} = \sum_{v_h \in C_j} \mathcal{S}_{ith}$. For brevity, we call $\mathcal{X}_{itj}$ the state of group C_j at timestamp t w.r.t. p_i. Finally, in case $C = \{V\}$, we have the population-level popularity.*

Problem 9.2.2 (Group-level Popularity Prediction) *Given a network G, a set of historical contents $P = \{p_1, p_2, \ldots, p_m\}$ (each content was observed over q timestamps), a set of user groups $C = \{C_1, C_2, \ldots, C_l\}$, and the group-level popularity of a new content p_{m+1} during its first t_1 timestamps ($t_1 < q$), predict the group-level popularity of p_{m+1} during time period $[t_1 + 1, q]$, that is, given $\{x_{m+1,1}, x_{m+1,2}, \ldots, x_{m+1,t_1}\}$, predict $\{x_{m+1,t_1+1}, \ldots, x_{m+1,q}\}$.*

We solve Problem 9.2.1 in Section 9.3 and Problem 9.2.2 in Section 9.4.

Example 2. For a project in Behance, we want to predict how many users in total and in each user group would appreciate it. Fig. 9.2a shows the group-level popularity of the same project in Fig. 9.1b (normalized by the total number of appreciating users at time t_1). Fig. 9.2c shows our popularity prediction for this project using its top-3 similar projects in Fig 9.2d. Our prediction is very close to the ground truth.

9.3 User Clustering

For Problem 9.2.1, we first discuss the four goals, G1-G4, of clustering users, and then propose how to achieve the goals.

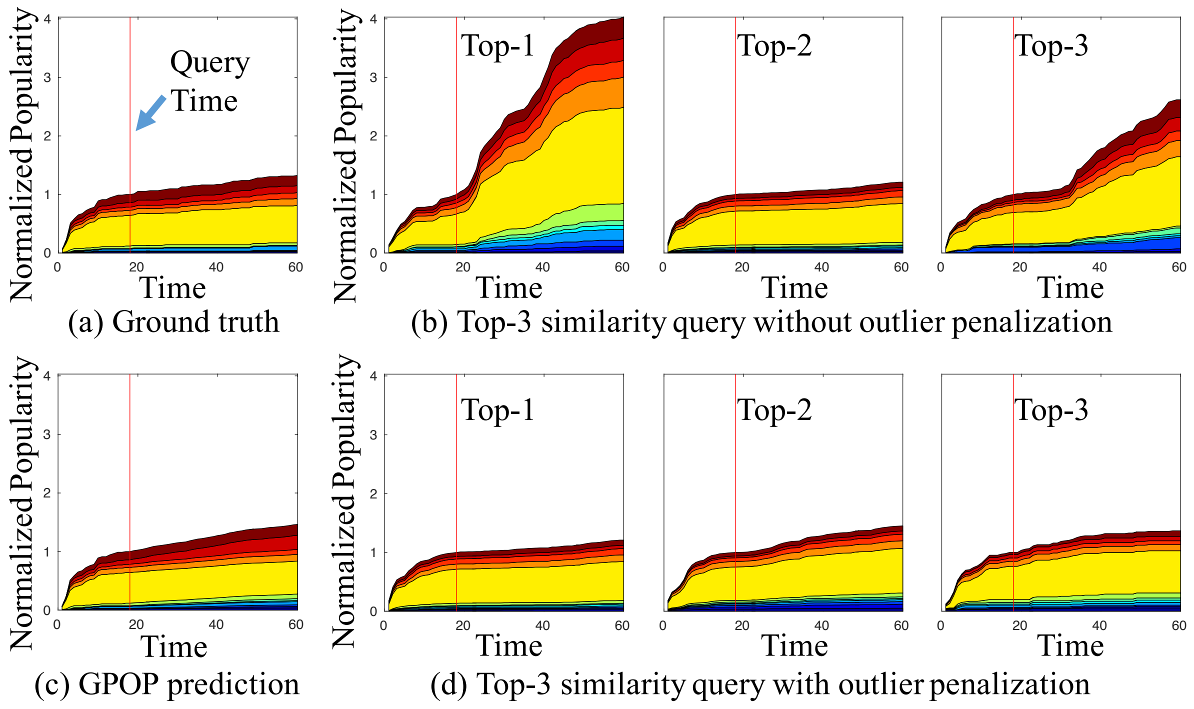


Figure 9.2: Example top-3 similarity query and prediction after the first 3 days (marked by the vertical line at $t_1 = 18$) for Behance. Popularity is normalized by the population-level popularity at t_1.

9.3.1 Clustering goals

G1: Group users with similar interests and behaviors. First, the behaviors and interests of users should be similar within the same group and different across different groups, leading to meaningful and useful groups for real-world applications. For example, an ad campaign may choose to target only a few relevant groups, knowing that the users in these groups will react to the advertised products instead of wasting money on other irrelevant groups. Similarly, web content can be prefetched in batches to only groups of users that are more likely to react to that content, avoiding unnecessary bandwidth and storage cost.

G2: Capture future changes in user interests. Using past user behaviors to cluster users is prone to overfitting: the obtained groups are good for historical contents, but may fail to capture a change of user interests on unobserved future content for two reasons. First, the spread of a content in a network is highly random and noisy, especially at the user level [169]. Fitting the clusters too tightly to the historical data would thus capture this noise. Second, many users are not active and react to few contents. Such users can be put in any group without incurring a significant cost in the clustering objective, making the cluster membership more random and less powerful in modeling future events.

G3: Capture the paths of information spread. There are two main ways a user gets exposed to a new content, which may in turn trigger him/her to react to that content: (i) via the network structure, or (ii) via an external source. For example, on Twitter, a user can learn about a new hashtag either (i) by reading his/her friends' tweets, or (ii) via other websites. Similarly, in Behance.net, a user will be exposed to a new project either (i) if his/her friends have performed some actions on that project, or (ii) if s/he actively searches for the project using some side information.

G4: Avoid imbalanced user groups. In clustering users, we may obtain groups of largely varying sizes while still optimizing some clustering objective. For example, simply minimizing the edge cut in a graph among users may lead to one group with almost all users and many tiny groups with only a few users. Such a partition of the users is hardly useful for reducing cost in a targeted ad campaign or group-based web-content caching/prefetching. Thus, we propose to find groups with comparable sizes.

9.3.2 Clustering network-constrained popularity graph

For goal G1, users v_i and $v_{i'}$ should be more likely to be in the same group if they tend to react to the same contents at the same times, i.e., $\exists t, p_j$ s.t. $\mathcal{S}_{itj} > 0$ and $\mathcal{S}_{i'tj} > 0$. This is equivalent to clustering the following popularity graph $G^{\mathcal{S}}$ into separate groups of vertices to minimize the edge cut.

Definition 9.3.1 (Popularity graph) Given a user set $V = \{v_1, \ldots, v_n\}$, and the user-level popularity $\mathcal{S}$ over q timestamps of m historical contents $P = \{p_1, \ldots, p_m\}$, denote the popularity vertex set $S = \{s_{11}, \ldots, s_{it}, \ldots, s_{mq}\}$ as the set of all combinations of m contents and q timestamps. The popularity graph $G^{\mathcal{S}} = (V^{\mathcal{S}}, E^{\mathcal{S}}, N^{\mathcal{S}}, W^{\mathcal{S}})$ is then defined as a weighted undirected bipartite graph (see Fig. 9.3a) with vertex set $V^{\mathcal{S}} = V \cup S$, edge set $E^{\mathcal{S}} = \{(v_j, s_{it}) | \mathcal{S}_{itj} > 0\}$, vertex weights $N^{\mathcal{S}}$, and edge weights $W^{\mathcal{S}}$, such that $\forall v_j, v_{j'} \in V; s_{it}, s_{i't'} \in S$: $N^{\mathcal{S}}_{v_j} = 1; N^{\mathcal{S}}_{s_{it}} = 0; W^{\mathcal{S}}_{v_j, s_{it}} = W^{\mathcal{S}}_{s_{it}, v_j} = \mathcal{S}_{itj}; W^{\mathcal{S}}_{v_j, v_{j'}} = 0$; and $W^{\mathcal{S}}_{s_{it}, s_{i't'}} = 0$.

$G^{\mathcal{S}}$ captures past user behaviors but it does not help us with goals G2 and G3. Fortunately, even if future user interests are different from those obtained from historical data, a new content must still spread via the network structure G unless users actively approach this content via some external sources. Thus, the network structure can be included in the clustering framework to deal with this change. In other words, we claim that clustering the following network-constrained popularity graph G^*, which is the union of G and $G^{\mathcal{S}}$, will help us satisfy both goals G2 and G3.

Definition 9.3.2 (Network-constrained popularity graph) Given a graph $G = (V, E)$ and its popularity graph $G^{\mathcal{S}} = (V^{\mathcal{S}}, E^{\mathcal{S}}, N^{\mathcal{S}}, W^{\mathcal{S}})$, we define the network-constrained popularity graph (Fig. 9.3b) as $G^* = (V^*, E^*, N^*, W^*)$ with vertex set $V^* = V^{\mathcal{S}}$, edge set $E^* = E^{\mathcal{S}} \cup E$, vertex weights $N^* = N^{\mathcal{S}}$, and edge weights W^*, such that $\forall v_j, v_{j'} \in V$; $s_{it}, s_{i't'} \in S$: $W^*_{v_j, s_{it}} = W^*_{s_{it}, v_j} = \mathcal{S}_{itj}$; $W^*_{v_j, v_{j'}} = \max(A_{jj'}, A_{j'j})$; and $W^*_{s_{it}, s_{i't'}} = 0$, where A is the adjacency matrix of G.

By using G^*, the cluster membership of an inactive user can be decided more effectively: s/he is more likely to be in the same cluster with her/his friends, rather than some random users that are very far away in G but coincidentally active at the same time.

Finally, goal G4 will be satisfied if we cluster G^* with a balancing criteria: we would like to obtain a partition $C = \{C_1, \ldots, C_l\}$ of V, such that $|C_j| \approx |C_{j'}| \; \forall C_j, C_{j'} \in C$. However, in clustering G^*, we actually obtain a partition $C^* = \{C^*_1, \ldots, C^*_l\}$ of V^*, such that $C^*_j \cap C^*_{j'} = \emptyset$,

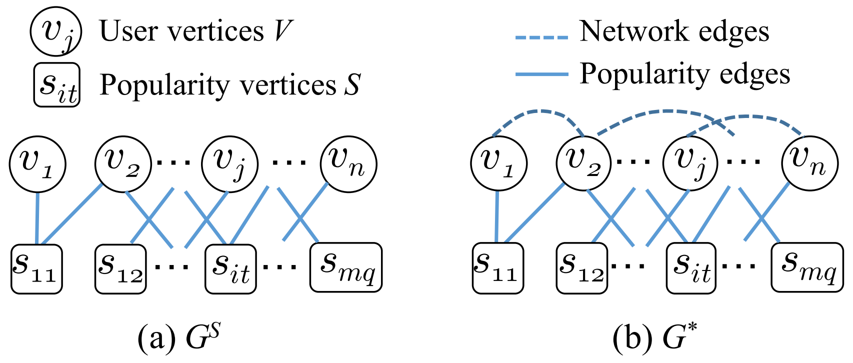


Figure 9.3: (a) Popularity graph $G^{\mathcal{S}}$ and (b) network-constrained popularity graph G^*

$\cup_j C_j^* = V^*$. We can easily convert C^* into C by choosing $C_j = C_j^* \cap V \; \forall 1 \leq j \leq l$. Moreover, define the weight $w(C_j^*)$ of each group C_j^* as the sum of all vertex weights in C_j^*, then:

$$w(C_j^*) = \textstyle\sum_{v_i \in C_j^* \cap V} N_{v_i}^* + \textstyle\sum_{s_{it} \in C_j^* \cap S} N_{s_{it}}^* = |C_j|$$

Thus, the balancing criteria on C can be translated into a balancing criteria on C^*, i.e., $w(C_i^*) \approx w(C_j^*) \; \forall C_i^*, C_j^* \in C^*$.

Clustering objectives: Based on the above intuitions, we cluster G^* with two objectives to satisfy all goals G1-G4:

1. Weighted edge cut minimization:

$$\min_{C^*} \textstyle\sum_{u \in C_i^*, v \in C_j^*, i \neq j} W_{u,v}^*$$

2. Group balancing:

$$\frac{w(C_j^*)}{n/l} \leq 1 + \beta, \forall C_j^* \in C^* \tag{9.1}$$

where $\beta > 0$ is a predefined imbalance factor.

The first objective assures that each user group is homogeneous since it tries to minimize the amount of weighted edges between different groups. The second objective guarantees that group sizes do not deviate too far from the average size n/l, where l is the desired number of

Algorithm 9 Find-User-Groups

```
Input: Network $G = (V, E)$. Number of user groups $l$
    Historical contents $P = \{p_1, \dots, p_m\}$. Imbalance factor $\beta$
Output: $C = \{C_1, \dots, C_l\}$
1: $\mathcal{S} \leftarrow$ Network state tensor of $P$
2: $G^* \leftarrow$ Network-constrained popularity graph$(G, \mathcal{S})$
3: $C^* \leftarrow$ Part-Graph-K-way$(G^*, l, \beta)$
4: $C \leftarrow \{C_j^* \cap V | j = 1, \dots, l\}$
5: return $C$
```

groups.

Clustering algorithm: We use the multilevel k-way partitioning algorithm [140] (Part-Graph-K-way) for graph clustering since it is scalable and supports our two objectives. The final clustering procedure is summarized in Algorithm 9. **Example 3.** Fig. 9.4a shows the average group sizes over 5-fold cross validation clustering for our Behance data: the groups clearly have similar sizes. Fig. 9.4b shows the group-level popularity of one example project in the testing set (one fold) given three different partitions of users obtained by clustering G^*, G, and $G^{\mathcal{S}}$ on the training data (the other four folds). Clearly, combining G and $G^{\mathcal{S}}$ to get G^* makes the obtained user groups more homogeneous on testing data.

9.4 Hierarchical Prediction

For Problem 9.2.2, we find the top-k similar contents at group and population levels for p_{m+1}, and then perform coupled tensor decomposition on these top-k contents with a novel hierarchical constraint to predict p_{m+1}'s future popularity.

9.4.1 A baseline approach

Once the groups $C = \{C_1, \dots, C_l\}$ are defined, we can create a *group-level popularity tensor* $\mathcal{X}$ for $P \cup \{p_{m+1}\}$ as shown in the left hand side of Fig. 9.5, where $\mathcal{X}_{itj}$ is defined as in

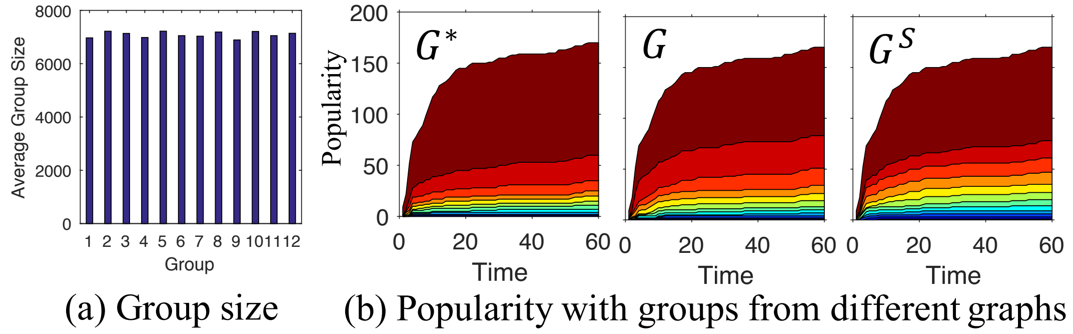


Figure 9.4: Behance data: (a) Group sizes (for G^*, 5-fold cross validation); (b) Group-level popularity of an example project.

Definition 9.2.2, and $\mathcal{X}_{m+1,t,j}$ is missing for all $t > t_1$. We can perform tensor completion to fill in these missing values and predict p_{m+1}. In particular, we decompose $\mathcal{X}$ into three matrices $D \in \mathbb{R}^{(m+1)\times R}$, $J \in \mathbb{R}^{q\times R}$, $F \in \mathbb{R}^{l\times R}$ using PARAFAC [5] such that, for all observed entries:

$$\mathcal{X}_{itj} = \textstyle\sum_{r=1}^{R} D_{ir} J_{tr} F_{jr} \tag{9.2}$$

or equivalently, $\mathcal{X} = [[D, J, F]]$, where D is the factor matrix for contents in $P \cup \{p_{m+1}\}$, J is the factor matrix for q timestamps, F is the factor matrix for l groups, and R is the number of latent dimensions. To learn D, J, and F, we minimize this objective function using gradient descent [78]:

$$\mathcal{L} = \tfrac{1}{2}\|\mathcal{M} * (\mathcal{X} - [[D, J, F]])\|_F^2 + \tfrac{\lambda}{2}(\|D\|_F^2 + \|J\|_F^2 + \|F\|_F^2) \tag{9.3}$$

where "$*$" is the element-wise tensor product, $\|.\|_F$ is the Frobenius norm, $\lambda > 0$ is a regularization factor to avoid overfitting, and $\mathcal{M}$ is a mask tensor of the same size as $\mathcal{X}$, indicating observed entries in $\mathcal{X}$, i.e., $\mathcal{M}_{itj} = 0$ iff $i = m+1, t_1 < t \leq q$, and $\mathcal{M}_{itj} = 1$ otherwise.

Drawbacks: (i) P can be large and contains contents vastly different from p_{m+1}, causing unnecessary computational cost and degrading accuracy. (ii) $\mathcal{M}$ is a dense tensor, leading to huge memory and computational costs if $\mathcal{X}$ is large.

9.4.2 Finding Top-k Similar Contents

Due to the drawbacks in Section 9.4.1, we claim that including only the top-k contents in P that are similar to p_{m+1} in tensor $\mathcal{X}$ makes $\mathcal{X}$ smaller and more relevant.

In Equation 9.2, predicting p_{m+1} is equivalent to learning the $(m+1)$th row of D. If J, F and the first m rows of D are fixed, this task is equivalent to representing row $(m+1)$th as a linear combination of the first m rows in D. Thus, the best candidate for predicting the $(m+1)$th row is some row i_1 in D, if it exists, such that $D_{m+1,r} = \beta D_{i_1,r}\ \forall 1 \leq r \leq R$ for some $\beta \in \mathbb{R}$. This is because we then need to learn only a single parameter β to make a perfect prediction for p_{m+1}:

$$\mathcal{X}_{m+1,t,j} = \sum_{r=1}^{R} \beta D_{i_1 r} J_{tr} F_{jr} = \beta \mathcal{X}_{i_1 t j}$$

for $\forall 1 \leq t \leq q; 1 \leq j \leq l$. Therefore, we propose to find top-k similar contents for p_{m+1} in a normalized space: given the training time period $[1, t_1]$, we normalize each content in $\mathcal{X}$ by its population-level popularity at time t_1 as follows:

$$\tilde{\mathcal{X}}_{itj} = \mathcal{X}_{itj} / \sum_j \mathcal{X}_{it_1 j} \quad \forall i, t, j \tag{9.4}$$

We then define the distance at timestamp T between p_{m+1} and another content p_i as the Eu-

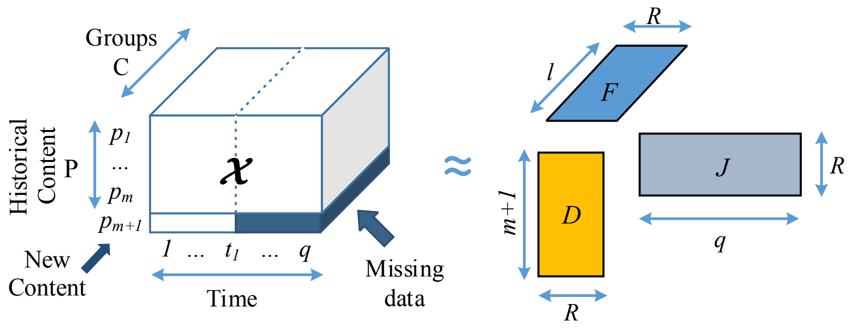


Figure 9.5: Predict missing values in group-level popularity tensor $\mathcal{X}$ using PARAFAC [5]. All timestamps are relative to the creation time of each content.

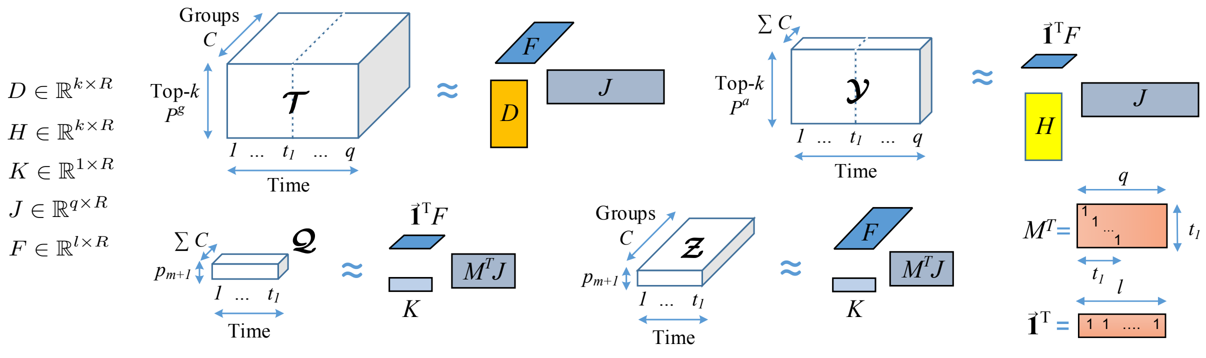


Figure 9.6: Hierarchical prediction using coupled tensor factorization. P^g (and P^a) are the top-k contents with similar group-level (and population-level) popularity as p_{m+1} over the time period $[1, t_1]$. $M^T J$ contains the first t_1 rows of J; $\vec{1}^T F$ is the sum of rows of F.

clidean distance:

$$\delta_T(p_i, p_{m+1}) = \sqrt{\sum_{t=1,\ldots,T;j=1,\ldots,l} (\tilde{\mathcal{X}}_{itj} - \tilde{\mathcal{X}}_{m+1,t,j})^2} \tag{9.5}$$

The top-k similar contents are then the contents with the smallest distance to p_{m+1} at time t_1.

Outliers: The top-k contents may be similar to p_{m+1} at time t_1, but very different from p_{m+1} in the future due to some unforeseeable events after t_1. For example, on Behance.net, a project could be promoted by the website and becomes popular even though it was barely noticed before. Similarly, in Twitter, some real-world events outside the social network may boost the usage of some hashtags suddenly. Therefore, we reduce the impact of such a historical outlier by including an outlierness score defined as the average distance at time q between it and the rest of the historical contents. The new distance δ_T^{out} at time T is defined as:

$$\delta_T^{out}(p_i, p_{m+1}) = \frac{\delta_T(p_i,p_{m+1})}{m-1} \times \sum_{j=1,\ldots,m;j\neq i} \delta_q(p_i, p_j) \tag{9.6}$$

Example 4. Fig. 9.2(b, d) show an example top-k query using δ and δ^{out}. Since δ does not penalize outliers, it returns the top-k contents (Fig. 9.2b) that are very different from the querying content (Fig. 9.2a) after time t_1. Whereas, δ^{out} gives us significantly better top-k results (Fig. 9.2d).

Algorithm 10 summarizes the procedure for finding top-k similar contents given historical data and user groups.

9.4.3 Tensor-based Hierarchical Prediction

Since the population level is less noisy, we borrow its strength to make better predictions for the group level in a hierarchical prediction framework.

Algorithm 11 summarizes how to hierarchically predict a new content p_{m+1}. First, in Lines 1-2, we use Algorithm 10 to find its top-k similar contents in P during time $[1, t_1]$ at group level (P^g) and population level (P^a). The latter is a special case of the former, where $C = \{V\}$. In line 3, we next build four tensors $\mathcal{T}$, $\mathcal{Y}$, $\mathcal{Z}$, $\mathcal{Q}$ as shown in Fig. 9.6:

$$\mathcal{T} \in \mathbb{R}^{k \times q \times l}; \mathcal{Y} \in \mathbb{R}^{k \times q \times 1}; \mathcal{Z} \in \mathbb{R}^{1 \times t_1 \times l}; \mathcal{Q} \in \mathbb{R}^{1 \times t_1 \times 1}$$

Algorithm 10 Top-k

```
Input: Historical contents $P = \{p_1, p_2, \dots, p_m\}$
    Partially observed content $p_{m+1}$
    Maximum observed timestamp $t_1 < q$
    User groups $C = \{C_1, C_2, ...\}$. Number of top items $k$
Output: Top-$k$ content IDs
1: Construct tensor $\mathcal{X}$ for $P \cup \{p_{m+1}\}$ & $C$ (Definition 9.2.2)
2: $\tilde{\mathcal{X}} \leftarrow$ Normalize $\mathcal{X}$ at time $t_1$ using Equation 9.4
3: for $i := 1$ to $m$ do
4:     $d_i \leftarrow \delta^{out}_{t_1}(p_i, p_{m+1})$ {Equation 9.6}
5: end for
6: return Top-$k$ indices with the smallest values in $d$
```

$$\mathcal{T}_{itj} = \sum_{v_h \in C_j} \mathcal{S}_{ith} \, \forall p_i \in P^g; 1 \leq t \leq q; 1 \leq j \leq l \tag{9.7}$$

$$\mathcal{Y}_{it1} = \sum_{v \in V} \mathcal{S}_{ith} \, \forall p_i \in P^a; 1 \leq t \leq q \tag{9.8}$$

$$\mathcal{Z}_{1tj} = \sum_{v_h \in C_j} \mathcal{S}_{m+1,t,h} \, \forall 1 \leq t \leq t_1; 1 \leq j \leq l \tag{9.9}$$

$$\mathcal{Q}_{1t1} = \sum_{v_h \in V} \mathcal{S}_{m+1,t,h} \, \forall 1 \leq t \leq t_1 \tag{9.10}$$

$\mathcal{T}$ and $\mathcal{Y}$ store the group-level and population-level popularity of contents in P^g and P^a respectively. $\mathcal{Z}$ and $\mathcal{Q}$ store the group-level and population-level popularity of p_{m+1} during time $[1, t_1]$ respectively. In line 4, we decompose these tensors into five factor matrices (Fig. 9.6) as detailed here:

$D \in \mathbb{R}^{k\times R}; H \in \mathbb{R}^{k\times R}; K \in \mathbb{R}^{1\times R}; J \in \mathbb{R}^{q\times R}; F \in \mathbb{R}^{l\times R}$

$$\mathcal{T} \approx [[D, J, F]] = \mathcal{T}^* \tag{9.11}$$

$$\mathcal{Y} \approx [[H, J, \vec{\mathbf{1}}^T F]] = \mathcal{Y}^* \tag{9.12}$$

$$\mathcal{Z} \approx [[K, M^T J, F]] = \mathcal{Z}^* \tag{9.13}$$

$$\mathcal{Q} \approx [[K, M^T J, \vec{\mathbf{1}}^T F]] = \mathcal{Q}^* \tag{9.14}$$

where R is the chosen number of latent dimensions; $\vec{\mathbf{1}}$ is an all-one column vector with l elements; and M is a $q \times t_1$ mask matrix to extract the first t_1 rows of matrix J, i.e.,

$$M_{ii} = 1 \, \forall i \text{ and } M_{ij} = 0 \, \forall i \neq j \tag{9.15}$$

To predict p_{m+1} (Lines 5-10), we use K, F, and the last $q - t_1$ rows of J (corresponding to time period $[t_1 + 1, q]$).

Intuitively, the rows of D and H represent the contents in P^g and P^a respectively; the rows of J represent the q timestamps; K has only one row representing the new content p_{m+1}; and the rows of F represent l user groups in C. Additionally, $\vec{1}^T F$ is the sum of the rows in F, representing the population-level popularity; while $M^T J$ are the first t_1 rows of J, representing the observed time period $[1, t_1]$.

Equations 9.11 and 9.12 capture the latent representations at the group and population levels using the historical contents respectively; whereas Equations 9.13 and 9.14 map p_{m+1} to the same latent space as that of the historical contents by sharing the factor matrices for time J and groups F. Since data for p_{m+1} is incomplete, only the observed part $M^T J$ of J is shared. Finally, by sharing factor matrix F in these four equations, we effectively learn a *hierarchical model* at both the group and population levels simultaneously.

Why coupled tensor decomposition? While $\mathcal{Y}$ and $\mathcal{Z}$ can be represented as matrices, and $\mathcal{Q}$ can be represented as a vector instead of tensors, we choose to use tensors in our formulation because of two reasons. First, PARAFAC decompositions are often unique, leading to more stable results and faster convergence compared to other matrix factorizations (which are often not unique, except for SVD) [5]. Second, the hierarchical structure of the user groups are naturally reflected in the decomposition when $\mathcal{Y}$, $\mathcal{Z}$ and $\mathcal{Q}$ are represented as tensors. In particular, $\mathcal{Y}$ and $\mathcal{Q}$ are simply the collapsed versions of $\mathcal{T}$ and $\mathcal{Z}$ along the group (3^{rd}) dimension respectively. Thus, the factor along the 3^{rd} dimension of $\mathcal{Y}$ and $\mathcal{Q}$ is also the sum of the rows

Algorithm 11 GPOP (Group-level POpularity Prediction)

```
Input: Partially observed content $p_{m+1}$ during time $[1, t_1]$.
    Historical contents $P = \{p_1, \dots, p_m\}$. User groups $C = \{C_1, ..., C_l\}$. Number of latent
    dimensions $R$
Output: Prediction of $p_{m+1}$: $\{x_{m+1,t_1+1}, \dots, x_{m+1,q}\}$
 1: $P^g \leftarrow$ Top-$k(P, p_{m+1}, t_1, C)$\{group level\}
 2: $P^a \leftarrow$ Top-$k(P, p_{m+1}, t_1, \{V\})$\{population level\}
 3: Create $\mathcal{T}, \mathcal{Y}, \mathcal{Z}, \mathcal{Q}$ using Equations 9.7-9.10
 4: $D, H, K, J, F \leftarrow$ Factorize-Tensors$(\mathcal{T}, \mathcal{Y}, \mathcal{Z}, \mathcal{Q}, R)$
 5: $J^* \leftarrow$ Rows $(t_1 + 1)$ to $q$ of $J$
 6: $\mathcal{Q}^* \leftarrow [[K, J^*, F]]$
 7: $x_{m+1,t_1+i} \leftarrow \{\mathcal{Q}^*_{1,i,j} | j = 1, \dots, l\}\ \forall i = 1, \dots, q - t_1$
 8: return $\{x_{m+1,t_1+1}, \dots, x_{m+1,q}\}$
```

in the factor for $\mathcal{T}$ and $\mathcal{Z}$ along the 3^{rd} dimension.

Optimization solution: Algorithm 12 shows how to find the five factor matrices by using gradient descent to minimize the following objective function:

$$\begin{aligned}\mathcal{L} = &\frac{1}{2}\|\mathcal{T} - [[D, J, F]]\|_F^2 + \frac{1}{2}\|\mathcal{Y} - [[H, J, \vec{\mathbf{1}}^T F]]\|_F^2 \\ &+ \frac{1}{2}\|\mathcal{Z} - [[K, M^T J, F]]\|_F^2 + \frac{1}{2}\|\mathcal{Q} - [[K, M^T J, \vec{\mathbf{1}}^T F]]\|_F^2 \\ &+ \frac{\lambda}{2}(\|D\|_F^2 + \|J\|_F^2 + \|F\|_F^2 + \|H\|_F^2 + \|K\|_F^2) \end{aligned} \tag{9.16}$$

where $\lambda > 0$ is a regularization factor to avoid overfitting.

The gradients of $\mathcal{L}$ are:

$$\nabla_D \mathcal{L} = (\mathcal{T}^*_{(1)} - \mathcal{T}_{(1)})(F \odot J) + \lambda D \tag{9.17}$$

$$\nabla_H \mathcal{L} = (\mathcal{Y}^*_{(1)} - \mathcal{Y}_{(1)})(\vec{\mathbf{1}}^T F \odot J) + \lambda H \tag{9.18}$$

$$\begin{aligned}\nabla_K \mathcal{L} = &(\mathcal{Z}^*_{(1)} - \mathcal{Z}_{(1)})(F \odot (M^T J)) \\ &+ (\mathcal{Q}^*_{(1)} - \mathcal{Q}_{(1)})(\vec{\mathbf{1}}^T F \odot (M^T J)) + \lambda K\end{aligned} \tag{9.19}$$

$$\begin{aligned}\nabla_J \mathcal{L} = &(\mathcal{T}^*_{(2)} - \mathcal{T}_{(2)})(F \odot D) + (\mathcal{Y}^*_{(2)} - \mathcal{Y}_{(2)})(\vec{\mathbf{1}}^T F \odot H) \\ &+ M(\mathcal{Z}^*_{(2)} - \mathcal{Z}_{(2)})(F \odot K) \\ &+ M(\mathcal{Q}^*_{(2)} - \mathcal{Q}_{(2)})(\vec{\mathbf{1}}^T F \odot K) + \lambda J\end{aligned} \tag{9.20}$$

$$\begin{aligned}\nabla_F \mathcal{L} = &(\mathcal{T}^*_{(3)} - \mathcal{T}_{(3)})(J \odot D) + \vec{\mathbf{1}}(\mathcal{Y}^*_{(3)} - \mathcal{Y}_{(3)})(J \odot D) \\ &+ (\mathcal{Z}^*_{(3)} - \mathcal{Z}_{(3)})(M^T J \odot K) \\ &+ \vec{\mathbf{1}}(\mathcal{Q}^*_{(3)} - \mathcal{Q}_{(3)})(M^T J \odot K) + \lambda F\end{aligned} \tag{9.21}$$

where $\mathcal{T}_{(i)}$ denotes the mode-i matricization of $\mathcal{T}$, and "$\odot$" denotes the Khatri-Rao product as defined in [5].

Algorithm 12 Factorize-Tensors

```
Input: Tensors $\mathcal{T}, \mathcal{Y}, \mathcal{Z}, \mathcal{Q}$ as in Equations 9.7-9.10
    $R$: Number of latent dimensions
Output: Factor matrices $D, H, K, J, F$
    Compute $\mathcal{L}$ using Equation 9.16
1: while not converged do
2:     Compute step length $\alpha$
3:     Compute the gradients $\nabla_D\mathcal{L}$, $\nabla_H\mathcal{L}$, $\nabla_K\mathcal{L}$, $\nabla_J\mathcal{L}$, $\nabla_F\mathcal{L}$ using Equations 9.17-9.21
4:     $D \leftarrow D - \alpha\nabla_D\mathcal{L}$; $H \leftarrow H - \alpha\nabla_H\mathcal{L}$; $K \leftarrow K - \alpha\nabla_K\mathcal{L}$
5:     $J \leftarrow J - \alpha\nabla_J\mathcal{L}$; $F \leftarrow F - \alpha\nabla_F\mathcal{L}$
6:     Compute $\mathcal{L}$ using Equation 9.16
7: end while
8: return $D, H, K, J, F$
```

9.5 Experiments

9.5.1 Datasets

We use two real-world datasets for evaluation: **Behance** [59] and **Twitter** [57, 58] (see Table 9.1). Behance.net is a social network where users can share their creative works (projects) and appreciate each other's projects. Twitter is a micro-blogging platform where users post short messages (tweets) that may include hashtags. There is a directed following relationship among users in both these social networks. Since we only care if two neighboring users are active at the same time, we convert these networks into undirected networks. A content is a project in Behance or a hashtag in Twitter. The popularity of a project is the number of users who have appreciated it; whereas the popularity of a hashtag is the number of times it has been tweeted by users. We only use contents with at least 100 reacting users, and also remove users with less than 10 tweets on Twitter.

Comprehensive experimental analyses for both Twitter and Behance can be found in our technical report [170].

Table 9.1: Datasets

Datasets	Behance	Twitter
Data time range	June, 2014	Sep-Dec, 2009
#Users	85092	22255
#Edges (follows)	13428465	575819
#Contents	1326 projects	1015 hashtags
#Timestamps	60	24
Timestamp bin size	4 hours	4 hours
Prediction task		
History length t_1	18 (3 days)	12 (2 days)
Future length $(q - t_1)$	12 (2 days)	12 (2 days)

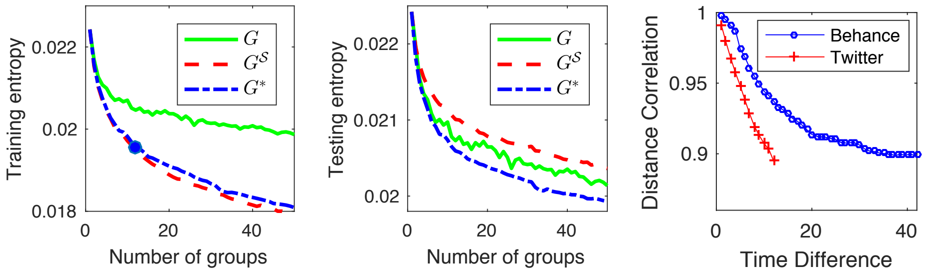
9.5.2 Quality of user clustering

We evaluate the quality of user groups in Problem 9.2.1 using entropy. A smaller entropy means more homogeneous groups. Given a content p_i, and a timestamp t, we define the active probability of users in a group C_j as the proportion of users with non-zero states, i.e., $p_{act} = |\{v_h \in C_j | \mathcal{S}_{ith} > 0)\}|/|C_j|$. Then, the entropy of $C = \{C_1, \ldots, C_l\}$ w.r.t. a set of historical contents P during a time period $[1, q]$ is:

$$h(C) = \textstyle\sum_{j=1}^{l} \frac{|C_j|}{|V|} \frac{1}{mq} \sum_{p_i \in P; 1 \le t \le q} h(C_j, p_i, t) \tag{9.22}$$

where $h(C_j, p_i, t) = -p_{act} \log p_{act} - (1 - p_{act}) \log (1 - p_{act})$

Effect of network structure: Fig. 9.7(a, b) show the average entropy over 5-fold cross



(a) Training entropy (b) Testing entropy (c) Distance correlation

Figure 9.7: (a,b) 5-fold clustering results for Behance. The best number of clusters chosen by the elbow method [6] is marked by a blue circle in figure (a). (b) Pearson correlation of distances at time t_1 and t_2 v.s. $(t_2 - t_1)$

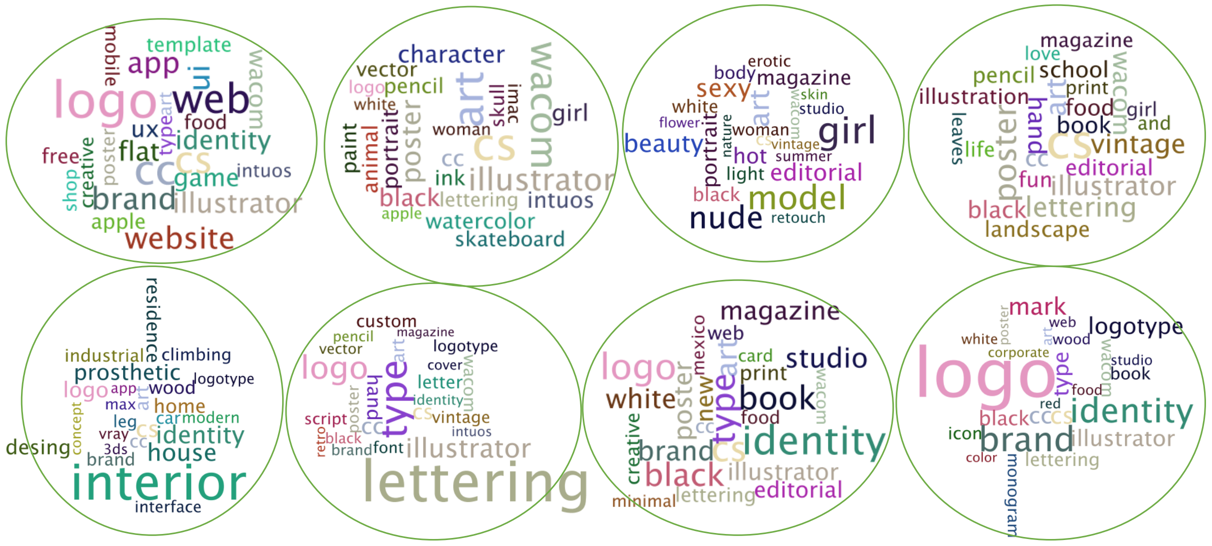


Figure 9.8: Word clouds of project tags for 8 user groups (Behance).

validation as the number of groups varies when G, $G^{\mathcal{S}}$ and G^{*} are clustered for Behance. Obviously, the higher the number of clusters, the smaller the entropy. More importantly, the effect of overfitting can be seen clearly. Clustering the tensor graph $G^{\mathcal{S}}$ provides the best groups on the training sets; but the obtained groups fit the testing sets very poorly. On the contrary, the qualities of groups obtained from the network G are similar for both the training and testing sets, suggesting that the effect of the network structure is consistent across different contents. Finally, groups obtained from G^{*} have comparable quality to those from $G^{\mathcal{S}}$ on training sets, while superior to both the groups obtained from G and $G^{\mathcal{S}}$ on the testing sets.

Number of user groups: We use the elbow method [6] to choose the best number of clusters for each dataset: 12 for Behance (see Fig. 9.7a), and 11 for Twitter. Fig. 9.8 further shows the word clouds of users' topics of interest (the tags of appreciated projects) in 8 of the 12 user groups in Behance. Clearly, users' interests are consistent within each group.

9.5.3 Usability of top-k similarity queries

For the top-k prediction strategy in Section 9.4.2 to work, the distances between two contents should be consistent over time. Fig. 9.7c shows the Pearson correlations between the distances computed at two different timestamps ($t_1 = 12$, $t_2 \in [19, 60]$ for Behance, $t_2 \in [19, 24]$ for Twitter). Though the correlations decrease as the time differences increase for both Behance and Twitter, the correlations remain high (> 0.89). Using top-k for prediction is thus a reasonable choice.

9.5.4 Quality of hierarchical prediction

Settings

Baselines: All compared methods are listed in Table 9.2 and divided into four groups: (i) GPOP and its variants, (ii) other tensor decomposition approaches for popularity prediction, (iii) hierarchical time series prediction, and (iv) population-level prediction only. We also note in Table 9.2 at which levels the predictions are performed for each method (user, group, and population levels).

Parameter setting: We set R as 50 for group-level and 100 for user-level predictions; imbalance factor $\beta = 0.03$; $k = 10$; $\lambda = 0.1$ (chosen using cross validation). t_1 and q are chosen as in Table 9.1. To cope with the instability of random initialization for gradient descent, we run each tensor decomposition three times, and choose the best results. Predictions are done for 5-fold cross validation. Experiments are run on a Debian machine with Intel i7, 3.50GHz CPU and 15GB RAM. Codes are written in Matlab, using the Tensor Toolbox [171], Poblano Toolbox [172] and METIS library [173].

Evaluation: We define the *Relative mean Error* for the group (REG) and population (REP) levels as below:

$$REG = \frac{1}{m} \sum_i \frac{\sqrt{\sum_{t>t_1,j} (\mathcal{X}_{itj} - \hat{\mathcal{X}}_{itj})^2}}{\sqrt{\sum_{t>t_1,j} \mathcal{X}_{itj}^2}} \times 100\%$$

$REP = \frac{\sum_{i,t>t-1} |\sum_j \mathcal{X}_{itj} - \sum_j \hat{\mathcal{X}}_{itj}|}{\sum_{i,t>t_1,j} \mathcal{X}_{itj}} \times 100\%$

where $\hat{\mathcal{X}}$ contains the predicted group-level popularity.

Quantitative Performance

The prediction results for all compared methods are shown in Table 9.3: our GPOP framework consistently outperforms all baselines. We next discuss the results in more details.

Choice of clustered graphs: The first three rows of Table 9.3 show the prediction errors for user groups obtained from G^*, G, and $G^{\mathcal{S}}$ respectively. Since the clusters from G^* are the most homogeneous, they produce the smallest errors for groups as well as smaller errors for population.

Variants of GPOP: We verify the effectiveness of GPOP's two components: top-k similarity query, and tensor-based hierarchical prediction. As shown in Table 9.3, GPOP outperforms GPOP-NoTop, which uses all historical contents instead of just the top-k similar contents, proving the benefit of top-k prediction. GPOP is also better than GPOP-NoNorm, i.e., computing top-k on the normalized popularity $\tilde{\mathcal{X}}$ (Equation 9.4) is better than on the raw data $\mathcal{X}$. Next, GPOP is far superior to naively taking the average of the top-k similar contents (Group-Avg). Predicting each group separately (GroupSep) and ignoring the relationship among groups is also significantly worse than GPOP, confirming the advantages of hierarchical prediction. Finally, predicting at the group level is much easier than at the noisy user level, as shown by the extremely high errors of GPOP-User.

Other tensor-based approaches: We test three different options. First, coupled matrix-tensor factorization (CMTF [78]) uses both the tensor $\mathcal{S}$ and the adjacency matrix of G to predict at user level. Here, we naively set the same weights for $\mathcal{S}$ and G in its objective function. The noise and high number of latent dimensions to be learned lead to extremely high errors. The network G thus actually makes it even more difficult for CMTF to converge to a good solution. Second, we test a probabilistic tensor decomposition method named TriMine [3],

Table 9.2: Baselines, with notes on three levels of prediction: ($\star$) Group, ($\dagger$) Population, ($\ddagger$) User.

Methods	Descriptions
GPOP $\star\dagger$	top-k + hierarchical prediction
Variants of GPOP	
GPOP-NoTop $\star\dagger$	GPOP that uses all historical contents instead of top-k similar content
GPOP-NoNorm $\star\dagger$	GPOP where top-k is computed on $\mathcal{X}$ instead of $\tilde{\mathcal{X}}$ as in Equation 9.4
Group-Avg $\star$	Weighted average of top-k similar content
GroupSep $\star$	top-k + separate predictions for each group using GPOP with $\mathcal{Y}$ and $\mathcal{Q}$
GPOP-User $\dagger\ddagger$	GPOP with each user as a group
Other tensor-based approaches	
CMTF [78] $\ddagger$	Coupled matrix-tensor factorization ($\mathcal{S}$ & G)
TriMine [3] $\ddagger$	Co-evolving time-series prediction of $\mathcal{S}$
CP-wopt [174] $\star$	PARAFAC tensor completion (Eqn. 9.3)
Hierarchical time series prediction [175]	
ARIMA-COMB $\star$	ARIMA + optimal combination
ARIMA-BU $\star$	ARIMA + bottom-up
ETS-COMB $\star$	Exponential smoothing + opt. combination
ETS-BU $\star$	Exponential smoothing + bottom-up
Population-level popularity prediction only	
MRBF [4]$\dagger$	Multivariate linear & Radial Basis Function

which is oblivious to the network G, and only uses the time period $[1, t_1]$ in tensor $\mathcal{S}$ for learning and predicting. It also predicts at the user level, and thus is prone to noise, leading to a much higher error compared to GPOP. Finally, we evaluate tensor completion for the group-based tensor $\mathcal{X}$ using PARAFAC (CP-wopt [174]), as discussed in Equation 9.3. We only test CP-wopt for $\mathcal{X}$ (group-level) because it does not scale for $\mathcal{S}$ (user-level)—the mask tensor $\mathcal{M}$ in Equation 9.3 is huge and dense for our data. Our results show that CP-wopt is much less stable than coupled tensor decomposition, and often gets stuck at sub-optimal solutions, causing high errors.

Hierarchical time series prediction: We test two classic time series prediction approaches: ARIMA, and Exponential Smoothing (ETS). We also use two different ways of combining their

Table 9.3: Relative prediction errors (%). (*) marks experiments that did not finish after 1 day.

		Behance		Twitter	
		REP	REG	REP	REG
GPOP	GPOP in G^*	6.95	10.99	11.86	15.14
	GPOP in G	6.92	11.21	12.14	16.73
	GPOP in $G^{\mathcal{S}}$	6.95	10.99	12.04	15.47
Other baselines for G^*	GPOP-NoTop	7.81	12.57	12.57	33.38
	GPOP-NoNorm	7.37	12.15	11.78	16.7
	Group-Avg	7.48	11.44	12.05	15.73
	GroupSep	12.15	68.13	12.26	15.29
	GPOP-User	(*)	(*)	324.68	297.44
	CMTF	(*)	(*)	13343	20787
	TriMine	25.00	13.28	12.65	23.10
	CP-wopt	73.47	53.92	19.51	23.26
	ARIMA+COMB	9.52	16.70	34.72	37.54
	ARIMA+BU	9.36	16.05	33.15	35.96
	ETS+COMB	9.14	16.13	25.42	30.28
	ETS+BU	8.44	15.60	24.22	29.60
	MRBF	27.04	-	24.50	-

Table 9.4: Prediction errors (%) as $q - t_1$ and k vary for Behance

$q - t_1$	6	12	18	24	36	42
REP	3.91	6.95	9.49	11.62	15.09	16.7
REG	7.07	10.99	13.8	16.03	19.03	20.45

k	1	5	10	15	30	50
REP	7.75	7.07	6.95	6.91	6.94	6.97
REG	11.46	11.04	10.99	11.01	11.12	11.08

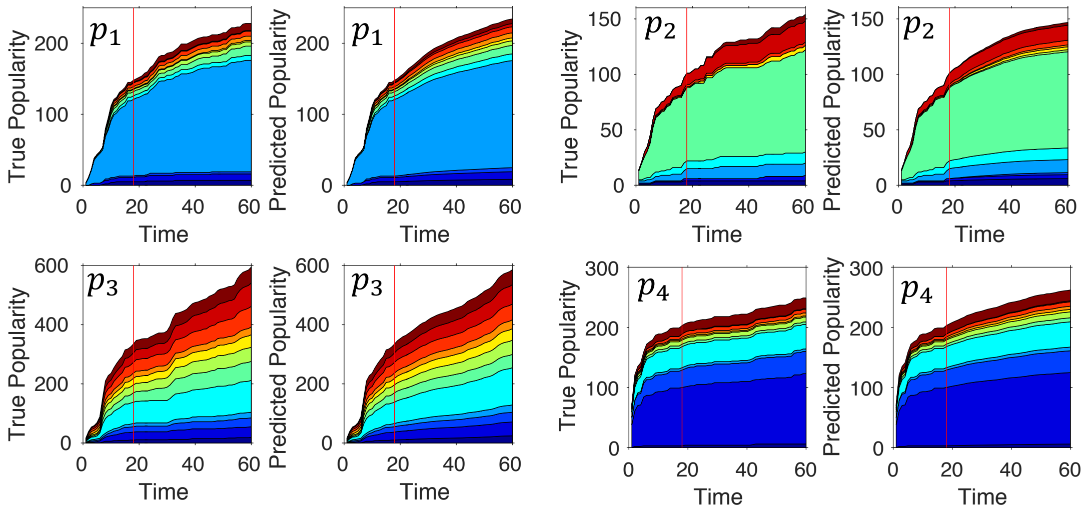


Figure 9.9: GPOP predictions for 4 example projects p_1-p_4 in Behance at time $t_1 = 18$ (3rd day, marked by the vertical red lines).

predictions hierarchically, i.e., bottom-up and optimal combination [175], leading to 4 baselines for hierarchical time series prediction (see Table 9.2). Here each time series corresponds to a user group. As shown in Table 9.3, GPOP outperforms all these 4 baselines significantly.

Population-level prediction only: MBRF [4] predicts the population-level popularity of Youtube videos by learning a multivariate linear model. Clearly, GPOP is superior to MBRF, both in terms of accuracy (Table 9.3) and the details of popularity at group level.

Future length and k: Table 9.4 shows the prediction errors of GPOP for Behance as the future periods $[t_1, q]$ and k vary ($t_1 = 18$). Clearly, the farther the future is, the harder it is to predict, leading to higher errors. As k increases, accuracy initially increases, but when k is too high, useless information is incorporated and increases the error.

Qualitative Performance

Fig. 9.9 shows the predictions of the next 7 days given the observations from the first 3 days for 4 example projects in Behance. As can be seen, GPOP makes good predictions for a variety of cases: typical projects p_1 and p_2 that are popular mostly in one user group; a project p_3 that are popular across all groups, possibly due to an unforeseen drift of users' interests w.r.t. the

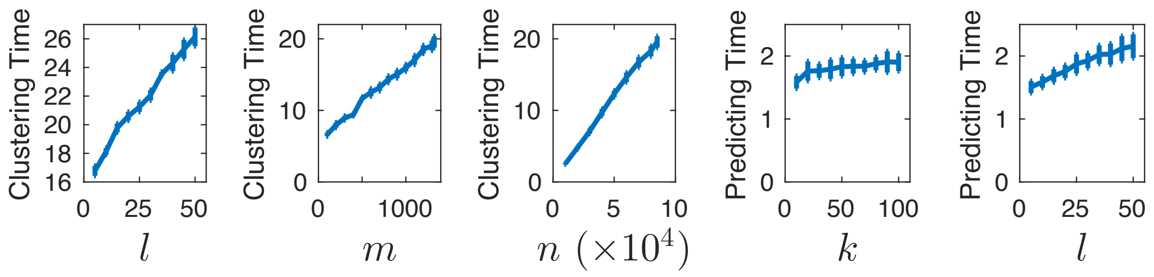


Figure 9.10: Running time for Behance (seconds) as l, m, n, k vary.

training data; and a project p_4 that ceased being popular after t_1.

9.5.5 Running time

Fig. 9.10 further shows that the average running time of GPOP (clustering and predicting) is linear in l, m, n, and k, making our solution scalable. On average, GPOP took 1.58 seconds for Behance, and 1.53 seconds for Twitter to predict one content. CMTF took more than 4 hours to finish one decomposition for Twitter, and did not finish after 1 day for Behance. TriMine finished predicting all content within 5 minutes but with much worse accuracy compared to GPOP.

9.6 Related Work

Popularity prediction: Since predicting the popularity of online content before publication is prone to large errors [176, 177], most earlier works focus on predictions after publication. Among these works, many papers simply use linear (or log-linear) regression to predict the aggregate (population-level) popularity of different types of contents [4, 165, 178–182], which often produces large errors [165, 183]. Thus, some papers adopt a classification approach to obtain higher accuracy at the loss of details: predict the range of popularity instead of the exact count [166, 184]. [168] uses the average of top-k similar tweets to predict a new tweet in Twitter, while [167] performs hierarchical prediction with ARMA model, obtaining good short-term but poor long-term predictions. [185, 186] combine data from different domains (websites) for prediction. Instead of treating user equally, several works also model behaviors of individual users (user-level popularity). For example, [161, 163] model users' behaviors to classify if a content would become popular; [162] proposes a probabilistic model based on Bayesian inference to predict the popularity of Twitter messages; [164] uses survival theory to predict the progression of an information cascade. These methods perform well for classification tasks but create large error for popularity count. We instead hierarchically predict popularity at group level, which is more fine-grained than the aggregate network level while less noisy than the individual user level.

Group-level information cascades: [85] and [187] solve the influence maximization and immunization problems for predefined groups respectively. [188] extracts community-level diffusion of retweets on the Weibo network but does not focus on predicting the future. We instead design the groups specifically for the task of predicting their future while also gaining insights into the group-level spread of information.

Time series modelling: Auto-regression and SIRS models have been added to tensor decomposition to model [189] and predict [3] time series (TriMine). Please see [175] for a survey of hierarchical time series prediction where predictions at different levels are combined in different ways.

9.7 Conclusion

In this paper, we developed a novel framework that addresses the important problem of online content prediction from a group-level popularity perspective. Our framework consists of two steps that first group users into clusters and then predict content popularity via a novel constrained tensor decomposition technique. Both network topology and interaction activities among users are exploited to learn a set of user clusters. Such a clustering solution is imposed as the hierarchical constraint in the PARAFAC tensor decomposition and we showed that optimizing its constrained function via gradient descent achieves faster convergence and leads to better prediction accuracy. Extensive empirical results demonstrate the effectiveness of our framework against eight baseline methods not only in terms of effectiveness but also of prediction accuracy, thus providing a better understanding about the spread of online content over social networks.

9.8 Acknowledgment

The work was supported by the US NSF grant IIS-1219254. Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053 (the ARL Network Science CTA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

Chapter 10

Beyond Models: Forecasting Complex Network Processes Directly from Data

10.1 Introduction

Complex network phenomena – such as information cascades in online social networks – are hard to fully observe, model, and forecast. In forecasting, a recent trend has been to forgo the use of parsimonious models in favor of models with increasingly large degrees of freedom that are trained to learn the behavior of a process from historical data. Extrapolating this trend into the future, eventually we would renounce models all together. *But is it possible to forecast the evolution of a complex stochastic process directly from the data without a model?* In this work we show that model-free forecasting is possible. We present SED, an algorithm that forecasts process statistics based on relationships of statistical equivalence using two general axioms and historical data. To the best of our knowledge, SED is the first method that can perform axiomatic, model-free forecasts of complex stochastic processes. Our simulations using simple and complex evolving processes and tests performed on a large real-world dataset show promising results.

Complex networked processes – such as information cascades and the spread of influence and viruses over online social networks – are hard to fully observe, model, and forecast. Self-reinforcing and latent effects can drive random and deterministic amplifications that are hard to predict without aggregate, large-population models. For instance, the way people react to social and monetary incentives can affect how information cascades spread over online social networks [190]; individuals also have evolving social interests [191, 192].

In our work we focus on forecasting statistics of complex network processes, such as the distribution of sizes of an epidemic process over a network. Recently, the availability of large-sample historical data – that details the evolution of similar processes in the past – has started a trend of forgoing parsimonious models in favor of models with increasingly large degrees of freedom that are trained over this historical data; these include latent Markovian infection models [191], auto-regressive models [193, 194], distance-based models for time series [195, 196], and classification-based approaches such as logistic regression [169] and naïve Bayes classifiers [197]. Hidden Markov models have also been proposed to replace parsimonious population models in ecology research [198].

Extrapolating this trend into the future, eventually we could renounce models all together. But is this possible? That is, *can we forecast statistics of complex network processes directly from the data without the help of models?* Such forecasting algorithm would be the *ultimate data-driven method.*

Contributions

In this work, we propose SED (Statistical Equivalence Digraph algorithm), an algorithm to forecast statistics of complex networked processes using general axioms that do not entail a model. Our algorithm works by extracting statistical information contained in the historical data through two axioms that define relationships of statistical equivalence between stochastic processes. To the best of our knowledge, our algorithm is the first that can perform (true) model-free forecast of network processes. Using simulation results and also a large real-world dataset, we show that SED is able to accurately forecast a variety of metrics under complex scenarios. We also show that accurate unbiased forecasting is limited to time horizons less than twice of that of the training data for a class of seemly simple infection processes. This proof further motivates our quest for forecasting directly from the data.

An important property of SED is its ability to adjust predictions according to the amount of evidence.

Consider forecasting the distribution of sizes of hashtag cascades #A and #B on Twitter:

- We observe 20 hashtag seeds[1] of #A: ten seeds get cascades of size one and ten get cascades of size at least ten.
- We also observe two seeds of #B generating cascades of size one and of size greater than ten, respectively.

Should we make the same forecast for #A and #B? As #B has less samples than #A we are more uncertain about #B's behavior. The lack of evidence prompts SED to assign greater uncertainty when forecasting metrics of #B, automatically adjusting forecasts to the amount of evidence.

Outline

Section 10.2 provides some definitions used throughout the paper. Section 10.3 explains the SED algorithm using the prediction of cascade statistics as an example. Section 10.4 presents the theory behind our approach. Section 10.5 tests the accuracy of SED using simulated data and shows an application of SED to forecast complex cascades over a large online social network dataset. Finally, Section 10.6 discusses the related work and Section 10.7 presents our conclusions.

10.2 Definitions

We first give a few definitions used throughout this work. In this paper we also often simplify our exposition by describing what are really general stochastic processes as an infection process on a network, a.k.a., a *cascade process*.

Definition 10.2.1 *[Cascade Process] $\Psi^{(i)} \in \Omega$ is a cascade process with ID i, $i = 1, \dots, |\Omega|$. For each cascade process $\Psi^{(i)} \in \Omega$ we observe a set of sample paths. The random variable $X^{(i)}_{\Delta t}$ is a measure over the sample paths of $\Psi^{(i)}$ over relative time interval $[0, \Delta t]$, where time zero is the time relative to the beginning of the cascade process. For instance, $X^{(i)}_{\Delta t}$ may be a random variable that gives the size of a cascade of $\Psi^{(i)}$ over time window $[0, \Delta t]$.*

[1]Seeds are network nodes that are infected independently from other already infected nodes.

The definition of $X^{(i)}_{\Delta t}$ allows us to be conservative and "mistakenly" merge two or more independent cascades of process $\Psi^{(i)}$ into a single larger cascade. This property is particularly useful when there is ambiguity to which independent cascade seed is responsible to infect a set of nodes.

Definition 10.2.2 *[Forecasting] The process to forecast $\Psi^{(0)}$, has k_0 observations $x^{(0,1)}_{\Delta t_1}, \dots, x^{(0,k_0)}_{\Delta t_1} \sim X^{(0)}_{\Delta t_1}$ over time window $[0, \Delta t_1]$, $\Delta t_1 > 0$, where time zero represents the time that the process starts. A forecast is an estimate of an statistic of $X^{(0)}_{\Delta t_2}$ where $\Delta t_2 > \Delta t_1$. A forecast assumes that all processes in Ω have been observed for at least Δt_2 time.*

For instance, if $\Psi^{(0)}$ is a viral marketing campaign we may want to forecast, using the observations of the first day of the campaign, what will be the average number of infected nodes per infected seed after a week, a statistic that predicts the performance of the campaign in the first week per user exposure.

Definition 10.2.3 *[Historical Data] The historical data of process $\Psi^{(i)}$, $i = 0, \dots$, defined as $\mathcal{X}^{(i)}_{\Delta t} = \{x^{(i,1)}_{\Delta t}, \dots, x^{(i,k_i)}_{\Delta t} \sim X^{(i)}_{\Delta t}\}$, is the set of all available observations of process $\Psi^{(i)}$ over time interval length Δt.*

In what follows we illustrate the SED algorithm. The theory behind the algorithm is explained in Section 10.4.

10.3 Algorithm

In this section, we present the SED Algorithm. Here we focus on the "how-to" solution. Our description of SED focuses on cascade processes to simplify our exposition; the algorithm for general networked processes is presented in Section 10.4, along with the justifications and the theory behind SED.

The SED algorithm (Algorithm 13) takes as input the set of observations of each process $\Psi^{(i)}$, over time interval lengths $\Delta t_1 > 0$ and $\Delta t_2 > \Delta t_1$ for $i = 1, \dots, |\Omega|$ and over interval length Δt_1 for the process we want to forecast $i = 0$. The goal is to forecast a statistic of $\Psi^{(0)}$ over time interval length Δt_2. The algorithm also requires a parameter α that later we describe how to automatically optimize.

Algorithm 13 The SED Algorithm

```
Input: Historical data $\mathcal{X}^{(i)}_{\Delta t}$, $i = 1, \ldots, |\Omega|$, $\Delta t = \Delta t_1, \Delta t_2$ and $\mathcal{X}^{(0)}_{\Delta t_1}$;
    $\alpha$: Probability amplifier parameter;
    $n$: Bootstrap resample size;
    $m$: Number of bootstrap resamples;
    Stat: Statistic of interest of $X^{(0)}_{\Delta t_2}$.
Output: Set of predicted observations of Stat$(X^{(0)}_{\Delta t_2})$
 1: for $i := 0$ to $|\Omega|$ do
 2:     $p_i \Leftarrow \frac{\sum_{j=0,j\neq i}^{|\Omega|} P_{\text{Kuiper}}(\mathcal{X}^{(i)}_{\Delta t_1}, \mathcal{X}^{(j)}_{\Delta t_1})}{|\Omega|}$
 3:     $w(i \rhd 0) \Leftarrow \frac{P_{\text{Kuiper}}(\mathcal{X}^{(i)}_{\Delta t_1}, \mathcal{X}^{(0)}_{\Delta t_1})}{p_i}$
 4: end for
 5: $\vec{W} \Leftarrow [w(1 \rhd 0)^\alpha, ..., w(|\Omega| \rhd 0)^\alpha]$
 6: $\vec{W} \Leftarrow \frac{1}{\|\vec{W}\|_1}\vec{W}$
 7: $\hat{Y} \Leftarrow [\,]$
 8: for $j = 1$ to $m$ do
 9:     Sample $i$ with probability $\vec{W}_i$, $i \in \{1, \ldots, |\Omega|\}$.
10:     $\hat{\mathcal{X}}^{(0)}_{\Delta t_2} \Leftarrow$ BoostrapSampling(src $= \mathcal{X}^{(i)}_{\Delta t_2}$, size $= n$)
11:     $\hat{Y}[j] \Leftarrow$ Stat$(\hat{\mathcal{X}}^{(0)}_{\Delta t_2})$
12: end for
13: return $\hat{Y}$
```

For now it is enough to know that a large value of α indicates that the process to forecast behaves like an outlier in our dataset. Two other parameters are related to bootstrap sampling, n and m, where $n = m = 100$ should suffice for most applications but larger values always give more accurate results. The final input is the statistic we wish to forecast.

Step 2 of Algorithm 13 applies a two-sample Kuiper's test [199] to get the probabilities that the observations of two processes i and j, $i \neq j$, come from the same underlying distribution. Step 3 of Algorithm 13 computes the equivalence score $w(i \rhd 0)$ between process $\Psi^{(i)}$ and the process to forecast $\Psi^{(0)}$. We call this equivalence score the *Equivalence Odds Ratio* (EOR), described in details in Section. 10.4.

With the EOR computed we build a weight vector $\vec{W}$ at step 5 of Algorithm 13, and normalize it using its $L1$-norm at step 6. The probability amplifier α (step 5) is a real number that is chosen automatically by another algorithm that we describe later. The next steps (9-11) forecast the statistic of interest using a two-step bootstrapping process with m and n resamples. These statistics may include, but are not limited to, the Complementary Cumulative Distribution Function (CCDF), mean, standard deviation and second moment of cascade sizes. We start by randomly sampling a process index i according to the weight in $\vec{W}$ (step 9). At step 10 we perform bootstrap sampling from the long-term observation $\mathcal{X}^{(i)}_{\Delta t_2}$ to obtain an estimate of the bootstrap resample of $\mathcal{X}^{(0)}_{\Delta t_2}$, where $|\hat{\mathcal{X}}^{(0)}_{\Delta t_2}| = n$. Finally, the statistics of the estimated $\hat{\mathcal{X}}^{(0)}_{\Delta t_2}$, e.g., the mean of $\hat{\mathcal{X}}^{(0)}_{\Delta t_2}$, are computed and stored in the vector $\hat{Y}$ (step 11). By repeating steps 9 to 11 m times we obtain the bootstrapped statistics, which provide the forecast of the statistic of interest over $X^{(0)}_{\Delta t_2}$.

10.3.1 Choosing Amplifier α and Sample Size n

To complete the algorithm, we also need to choose the amplification factor α and the sample size n. Larger values of n make the forecast more accurate, thus n is only limited by computational constraints. In our forecasts, however, we would like to compare our prediction with the ground truth of $X^{(0)}_{\Delta t_2}$, thus, we set $n = |X^{(0)}_{\Delta t_2}|$ to make the comparison easy. Finally, we determine α heuristically by minimizing the squared error in SED's prediction of the value of Stat($X^{(0)}_{\Delta t_1}$), which we can easily get an estimate with Stat($\mathcal{H}^{(0)}_{\Delta t_1}$).

10.4 Theory

The framework of classical statistics, as developed by pioneers like R.A. Fisher [200], describes stochastic processes using models as building blocks. For instance, a process can be a Galton-Watson process, Markovian, auto-regressive, or a hierarchical combination of processes and non-parametric distributions. In 1933 Kolmogorov [201] laid the modern axiomatic foundations of probability theory that eliminated the necessity of models, although models are still useful in Kolmogorov's framework. To the best of our knowledge, our SED method is the first method able to perform forecasting using axioms rather than models.

Roadmap: In what follows (Section 10.4.1) we detail our axiomatic forecast framework and our SED method. Section 10.4.2 takes an in-depth look at the problem providing forecast accuracy bounds for a class of stochastic processes, proving that in some scenarios even for the best model cannot make accurate long-term forecasts of seemly simple processes.

10.4.1 SED: Axiomatic Forecasting

In what follows, we describe our SED method. SED performs forecasts using the following two axioms in lieu of a model. For the sake of conciseness and clarity, the definition of dynamic networked processes is given later in the text in Definition 10.4.1.

Axiom 10.4.1 *The state of any dynamic networked process is unique at any time $t \geq 0$.*

Axiom 10.4.1 guarantees that at any given time t there is no ambiguity about the true state of the process, even if this true state is not observable.

Axiom 10.4.2 *Let Ω be a set of processes with historical data and let $\Psi^{(0)}$ be the process that we wish to forecast. Each process $\Psi' \in \Omega$, where $\Omega' = \Omega \cup \{\Psi^{(0)}\}$, with the exception of at most one process, is equivalent to one and only one other process in Ω' besides itself according to a measure $\lambda : \mathcal{S} \to \mathbb{R}$, where $\mathcal{S}$ is the appropriate σ-algebra.*

Note that Axioms 10.4.1 and 10.4.2 do not entail a model. To illustrate the use of Axiom 10.4.2 in our method consider the following example. Ω' is the set of epidemic processes over the same graph $G = (V, E)$. We wish to forecast another epidemic $\Psi^{(0)}$ using multiple independent sample paths of the processes in Ω'. Let $\lambda(\sigma)$ as defined in Axiom 10.4.2 count the number of infected nodes in a sample path $\sigma \in \mathcal{S}$. The example also works if λ is another non-trivial metric, such as the cumulative number of infected nodes, the Wiener index, the reproduction number at time $t > 0$. Axiom 10.4.2 guarantees that the set of all observed infection processes Ω' is such that for any process $\Psi' \in \Omega'$ there is one and only one other process $\Psi'' \in \Omega'$ for which the distribution of the number of infected nodes is exactly the same.

Unless stated otherwise in what follows we simplify our exposition assuming that Ω' has an even number of elements. Our method also can be readily extended to support $\lambda : \mathcal{S} \to \mathbb{R}^n$, $n > 1$, to incorporate complex features of the process. As we see later, the scenario with $n > 1$ requires the use of kernel two-sample tests [202] for the forecast. In what follows we consider $\lambda : \mathcal{S} \to \mathbb{R}$ unless stated otherwise.

Intuition: Axiom 10.4.2 goes at the heart of what it means to forecast the evolution of a process using historical data. We can only predict the future if, somehow, the present mimics what has happened in the past. When aided by models, there is a notion of behavior distance (e.g. likelihood function), which helps us find which processes in the past look similar to the process we are trying to forecast and use them to predict the future. Because models often have trouble assigning measures of similarity between two significantly different processes, many modern methods only consider "close enough distances", creating a manifold that is then used in the forecast. One of the challenges in manifold learning is determining a close enough distance where the model empirically works without being fooled by the data. Axiom 10.4.2 makes this distance precisely zero, which brings some advantages. First, the forecast does not need to learn parameters, often a computationally intensive task. Second, in a complex scenario it is probably better to state ignorance about process behavior than being very wrong. Network processes can be hard to forecast. In Section 10.4.2 we show that the statistical information used in the forecast of a seemly simple process cannot be extrapolated by models beyond the time horizon of the dataset (Δt_2).

Choice of λ: The measure λ can be thought of as a "fingerprint" of process behavior. With a bad choice of λ, too many "unrelated" processes have similar (but not equivalent) statistics, which in turn makes it hard to empirically distinguish them. In these cases SED assigns similar weights to most processes, and forecasting is performed conservatively (large confidence intervals). If the process to forecast has too few observations, SED also acts conservatively to reflect the uncertainty in the limited number of observations.

Violating Axiom 10.4.2: If no process in the historical data is equivalent to the process we wish to forecast (a reality in some scenarios), then one of two things can happen. If the process to forecast and most processes in the historical data have similar processes but not an equal process, then Axiom 10.4.2

provides a robust metric of process similarity. It is robust because it does not assume how processes behave and cannot be fooled by data. This shows well in our experiments with a real dataset. If most processes in the historical data are equal and the process to forecast is an outlier, SED will forecast in an uninformative way with large confidence intervals. This outlier status is easy to spot and stating the lack of confidence in the prediction is highly desirable.

Formal Definitions: In what follows, we use Random Marked Point Processes (RMPP) to formalize the networked process. Note that RMPPs allow us to avoid describing how the process evolves and, thus, no model is described. Let $G = (V, E)$ be a directed graph with node set V and edge set $E \subseteq V \times V$. Let $\mathcal{L}_V$ and $\mathcal{L}_E$ denote an arbitrary set of node and edge labels that will be used to define the state of the process at any time. As an example, in an epidemic process $\mathcal{L}_V = \{\text{infected}, \text{susceptible}\}$ and $\mathcal{L}_E = \emptyset$. The following generalization takes into account the complexity of real-world cascading processes.

Definition 10.4.1 *[Cascade Process] Let $Z_k^{(i)} \subseteq V \times \mathcal{L}_V \times E \times \mathcal{L}_E$ denote the state of the i-th process after $k > 0$ events. Let $W_k^{(i)} > 0$ be the time between the k-th and the $k+1$-st events. A dynamic networked process $\Psi^{(i)}$ is a simple Random Marked Point Process (RMPP) $\Psi^{(i)} = \{(Z_k^{(i)}, W_k^{(i)})\}_{k \in \mathbb{Z}^\star}$ operating over $V \times \mathcal{L}_V \times E \times \mathcal{L}_E$, where $\mathbb{Z}^\star$ is the set of non-negative integers. By convention,*

$$0 = T_0^{(i)} < T_1^{(i)} < \cdots$$

are the successive times at which the process evolves with $W_k^{(i)} := T_{k+1}^{(i)} - T_k^{(i)}$, $k \geq 0$.

Note that the dynamic networked process *need not follow the edges* of G. RMPPs are versatile stochastic models that allow arbitrary spatial and temporal correlations in process evolution. Also note that in real life we only have a finite number of realizations of the processes and, thus, we may not be able to distinguish some of them. In what follows, we define the measure λ precisely and introduce the notion of λ-stochastic equivalence.

Definition 10.4.2 (λ-stochastic Equivalence) *We define two stochastic processes $\Psi^{(a)}$ and $\Psi^{(b)}$ as equivalent according to a Lebesgue measure $\lambda : \mathcal{S} \to \mathbb{R}$ at time window $[0, \Delta t]$, $\Delta t \in (0, \infty)$, if $X_{\Delta t}^{(a)} \overset{d}{=} X_{\Delta t}^{(b)}$, where $X_{\Delta t}^{(k)} \equiv \lambda(\sigma_k^{(\Delta t)})$ is a random variable, $\sigma_k = \{(Z_i^{(k)}, W_i^{(k)}) : 0 \leq T_i^{(k)} \leq \Delta t\}$ is a* *sample path of the stochastic process $\Psi^{(k)}$, $k \in \{a, b\}$, $\mathcal{S}$ is the appropriate σ-algebra, and the operator $\overset{d}{=}$ defines the random variables are equal in distribution.*

In what follows, we use Axioms 10.4.1 and 10.4.2 and the definition of λ-stochastic equivalence to build an equivalence digraph representing the equivalence relationships between all processes in Ω'.

Stochastic Equivalence Digraph

Axioms 10.4.1 and 10.4.2 state that the cascade process $\Psi \in \Omega'$ – e.g., the process of spreading a Twitter hashtagover the Twitter network – has one and only one λ-stochastic equivalent process $\Psi' \in \Omega'$. To create our forecast, all we need to do is to identify this process. This idea yields a simple statement wherein lies the answer to our practical forecasting problem:

Theorem 10.4.1 *Any two-sample hypothesis test can be used to determine whether two processes $\Psi^{(i)}, \Psi^{(j)} \in \Omega'$ are λ-stochastic equivalent at time interval $\Delta t > 0$.*

Proof: Axiom 10.4.1 states that sample paths are simple and thus admit a stochastic equivalence measure λ. From Axiom 10.4.2, we know that all processes in Ω' have a λ-stochastic equivalent process (except at most one process if $|\Omega'|$ is odd). The set of sample paths of $\Psi^{(i)}$ and $\Psi^{(j)}$ entail a set of independent observations of $X^{(i)}_{\Delta t}$ and $X^{(i)}_{\Delta t}$). Thus, λ-stochastic equivalence yields $X^{(i)}_{\Delta t} \overset{d}{=} X^{(i)}_{\Delta t}$. A two-sample hypothesis test assesses whether independent observations of the random variables $X^{(i)}_{\Delta t}$ and $X^{(i)}_{\Delta t}$ are drawings of the same underlying distribution. Hence, a two-sample hypothesis test is a tool to determine whether $\Psi^{(i)}$ and $\Psi^{(j)}$ are λ-stochastic equivalent at time $\Delta t > 0$. ■

Interestingly, a test with low statistical power forces us to make conservative forecasts with large confidence intervals. The forecast uncertainty reflects the lack of statistical information. In such case, we can identify the problem as the test says that all processes in Ω' have a similar p-value. An interesting consequence of Axiom 10.4.2 and Theorem 10.4.1 is our ability to create a complete digraph whose nodes are the processes in Ω' and the weights represent the probability that they are λ-stochastic equivalent. We call the weights of the edges in this digraph the Equivalence Odds Ratio (EOR), defined as follows.

Definition 10.4.3 *[Equivalence Odds Ratio (EOR)] The EOR, denoted $w_t(i \rhd j)$, $i \neq j$, under n observations*
$\mathcal{X}^{(i)}_{\Delta t} = \{x^{(i,1)}_{\Delta t}, \ldots, x^{(i,n)}_{\Delta t}\}$ of $X^{(i)}_{\Delta t}$ and m observations
$\mathcal{X}^{(j)}_{\Delta t} = \{x^{(j,1)}_{\Delta t}, \ldots, x^{(j,m)}_{\Delta t}\}$ of $X^{(j)}_{\Delta t}$ is

$$w_t(i \rhd j) = |\Omega'| \frac{P_{Kuiper}(\mathcal{X}^{(j)}_{\Delta t}, \mathcal{X}^{(i)}_{\Delta t})}{\sum_{\Psi^{(h)} \in \Omega' \backslash \{\Psi^{(j)}\}} P_{Kuiper}(\mathcal{X}^{(i)}_{\Delta t}, \mathcal{X}^{(h)}_{\Delta t})} \tag{10.1}$$

where $P_{Kuiper}(\cdot, \cdot)$ is the p-value of a two-sample Kuiper's test [199, 203], $\mathcal{X}^{(h)}$ are the observations of process $\Psi^{(h)}$, and with Ω' is as defined in Axiom 10.4.2 and $X^{(i)}_{\Delta t}$, $X^{(j)}_{\Delta t}$ as in Definition 10.4.2.

The Equivalence Odds Ratio (EOR) $w_t(i \rhd j)$ is the ratio between $p(\mathcal{X}^{(i)}_{\Delta t}, \mathcal{X}^{(j)}_{\Delta t})$ and the average p-value when i is compared to all other cascades in $\Omega' \backslash \{\Psi^{(j)}\}$. The EOR is not to be confused with a likelihood-ratio test, which compares the likelihood of two models given the data. In our framework there are no models.

Definition 10.4.4 (Stochastic Equivalence Digraph) *Let $\mathcal{G}_{X_{\Delta t}}$ be a simple (no loops) complete digraph that connects all processes in Ω' with directed edge weights. For any two cascade processes $\Psi^{(i)}, \Psi^{(j)} \in \Omega'$, $i \neq j$, the weight of edge $i \to j$ is the EOR $w_t(i \rhd j)$.*

An EOR $w_t(i \rhd j) > 1$ suggests that process $\Psi^{(i)}$ is $w_t(i \rhd j)$ times more likely to be λ-stochastic equivalent to $\Psi^{(j)}$ than a random process selected from $\Omega' \backslash \{\Psi^{(j)}\}$. Note that the equivalence odds ratio is not symmetric, i.e., there can be processes $\Psi^{(i)}$ and $\Psi^{(j)}$ such that $w_t(i \rhd j) \neq w_t(j \rhd i)$. Note that Definition 10.4.4 does not impose the one-to-one equivalence required by Axiom 10.4.2. This is because the one-to-one equivalence is unnecessary and computationally expensive if Ω' is large, thus we assume we can safely approximate it through probabilistic matching. The Kuiper test in P_{Kuiper} can be also replaced by any two-sample hypothesis test such as Kolmogorov-Smirnov's test or extend our method to include P_{kernel} [202] that allows the use of a multi-dimensional λ through kernel two-sample tests. In one dimension, we choose Kuiper's test over Kolmogorov-Smirnov's test because the former gives equal importance to all domain values while the latter tends to be most sensitive around the median value [199, 203].

Next, we show how to use the *stochastic equivalence digraph* $\mathcal{G}_{X_{\Delta t}}$ to make predictions of cascade characteristics.

Forecasting Procedure in Time

Using the stochastic equivalence digraph $\mathcal{G}_{X_{\Delta t_1}}$ we can forecast $\Psi^{(i)}$ using an of estimate $P[X^{(i)}_{\Delta t_2} > x]$, $0 < \Delta t_1 < \Delta t_2$, obtained by the mixture

$$P[X^{(i)}_{\Delta t_2} > x] = \\ \frac{1}{C} \sum_{\Psi^{(j)} \in \Omega' \backslash \{\Psi^{(i)}\}} w_{\Delta t_1}(j \rhd i) P[X^{(j)}_{\Delta t_2} > x],$$

where $C = \sum_{j \in \Omega'} w_{\Delta t_1}(j \rhd i)$ is a normalization constant. The above equation gives the first iteration of Sinkhorn's algorithm that finds probabilistic pairwise matches in a weighted graph [204]. For computational reasons we limit our matching to a single iteration. Sinkhorn's algorithm converges to a doubly stochastic matrix that defines the probability of pairwise matchings in a weighted graph, widely used in soft matching problems [205]. If the SED adjacency matrix is irreducible then the probabilistic matching is unique [204]. To increase estimation accuracy when Ω' is large, we can add an exponential amplification factor:

$$P[X^{(i)}_{\Delta t_2} > x] = \\ \frac{1}{C_\alpha} \sum_{\Psi^{(j)} \in \Omega' \backslash \{\Psi^{(i)}\}} w_{\Delta t_1}(j \rhd i)^\alpha P[X^{(j)}_{\Delta t_2} > x],$$

where $C_\alpha = \sum_{j \in \Omega'} w_{\Delta t_1}(j \rhd i)^\alpha$ and α is chosen as to minimize a regret function over the estimated distribution $P[X^{(i)}_{\Delta t'_2} > x]$ for $\Delta t'_2 \approx \Delta t_1$. Note that even for historical cascades $\Psi^{(j)} \in \Omega' \backslash \{\Psi^{(i)}\}$, we do not have the true function $P[X^{(j)}_{\Delta t_2} > x]$ that is used inside the sum. Therefore, we estimate $P[X^{(b)}_{\Delta t_2} > x]$ by bootstrapping the observations of $X^{(j)}_{\Delta t_2}$ in the historical cascades. In Section 10.5.2 we show that the estimates have good accuracy in practice.

10.4.2 A Big Data Forecast Paradox

In what follows we present a seemly simple forecasting problem and prove that under conditions common to large social networks, no model or procedure is capable of extrapolating accurate unbiased cascade statistics beyond the time horizons of the historical data. This inability of models to extrapolate cascade statistics beyond what is already in the historical (training) data indicate that model-free forecasts can do as well as forecasts with perfect models.

Moreover, we also show the following paradox: As a the size of a power law network scales and increases both the historical data and the maximum cascade sizes, forecasts beyond the historical data horizon get more inaccurate while forecasts within the horizon get more accurate. This paradox poses a great challenge for big data analytics, where short-term forecasts can get increasingly better as the system and the historical data grow while long-term forecasts get worse.

This paradox happens because the present is a biased view of the future. More precisely, in the historical data we are more likely to see at least one infection from a cascade that will be larger in the future than a cascade that will be smaller. This bias is related to the inspection paradox [206] and depends on the observation window. The bias is so hard to correct for power law distributions that it denies us the ability to forecast beyond historical data horizons, where we have not yet recorded the bias. Moreover, our results are not confined to power laws; we consider all distributions (Type I, II, and III). The results apply to statistics such as the unbiased average of cascade sizes and the cascade size distribution.

Model: We collect data during time interval $[0, T]$ and seek to forecast statistics for interval $[0, cT]$, $c > 1$. Consider n independent cascades. Cascade infections arrive according to a constant-rate Poisson process with possibly distinct rates in the interval $[0, cT]$. The size of a cascade at time cT has distribution $\Lambda(cT) \sim \theta$, where θ is a distribution with support $\{1, \ldots, W\}$. Our initial goal is to estimate an unbiased average of the cascade sizes over the interval $[0, cT]$, $c > 1$. At first inspection the forecasting problem looks like a simple task and the impossibility results stated above seem surprising. We choose the Poisson process in our illustration precisely because its simplicity allows us to clearly understand why the forecasting problem is hard.

Our use of the Poisson process is also of interest because it connects the axiomatic framework with the classic model-based one. Poisson processes are arguably among the simplest and most widely used stochastic processes. Let N_t be the number of events in the time interval $[0, t]$. A Poisson process $\{\Lambda(\Delta t)\}$ with constant rate β is defined as an arrival process that satisfies the following three axioms:

A1. $\lim_{\epsilon \to 0} P[\Lambda(\Delta t) - \Lambda(\Delta t + \epsilon) = 1] = \beta\epsilon + o(\epsilon)$, and
$\lim_{\Delta t \to 0} P[N_t - N_{t+\Delta t} > 1] = o(\Delta t)$; that is, no two events can happen at the same time;

A2. for all t, $s > 0$, $N_{t+s} - N_t$ is independent of the history up to t, $\{N_u, u \leq t\}$;

A3. for all t, $s > 0$, $N_{t+s} - N_t$ is independent of t.

An equivalent constructive way to define the same process is: $P[N_t \geq k] = P[\sum_{i=1}^{k} X_i < t]$, where $\{X_1, \ldots, X_k\}$ are independent and identically distributed (i.i.d.) exponential random variables with parameter β.

Forecast Problem Definition. Let $\mathcal{O}_T$ denote the set of indices of the cascades in the historical data observed during interval $[0, T]$. The forecasting problem is defined as follows: We observe the cascades during time window $[0, T]$ and wish to forecast the average number of events we will observe in the window $[0, cT]$, $m_{cT} = \sum_{i=1}^{n} N_{cT}^{(i)}/n$.

Forecast Accuracy Bounds

In the following theorem, we prove that forecasting the average number of events in our mixture Poisson process is a hard problem.

Theorem 10.4.2 *Let $\{N_{cT}^{(i)}\}_{i=1}^{n}$, $N_{cT}^{(i)} \sim (\theta_1, \ldots, \theta_W)$, be a set of observed events of a mixture of n Poisson processes during time interval $[0, cT]$. Let $|\mathcal{O}_T|$ denote the number of processes with at least one arrival in the interval $[0, T]$. Assume we are given the best unbiased forecast function $m_{cT|T}$ of the average number of events m_{cT}, where $m_{cT|T}$ takes as input the observations over time interval $[0, T]$. Then, the following conditions regarding the mean squared error $MSE(m_{cT|T}) = E[(m_{cT|T} - m_{cT})^2]$ hold:*

1. *If θ_W decreases faster than exponentially in W, i.e.,*
   $-\log\theta_W = \omega(W)$*, then* $MSE(m_{cT|T}) = \Omega(1/|\mathcal{O}_T|)$.

2. *If θ_W decreases exponentially in W, i.e.,* $\log\theta_W = W\log b + o(W)$ *for some* $0 < b < 1$*, then*

   (a) $\log[MSE(m_{cT|T})] = \Omega(W - \log|\mathcal{O}_T|)$*, provided* $c > 1 + 1/b$*,*

   (b) $MSE(m_{cT|T}) = \Omega(W/|\mathcal{O}_T|)$*, provided* $c = 1 + 1/b$*,*

   (c) $MSE(m_{cT|T}) = \Omega(1/|\mathcal{O}_T|)$*, provided* $c < 1 + 1/b$.

3. *If θ_W decreases more slowly than exponential, i.e.,*
   $-\log\theta_W = o(W)$*, then*

   (a) $\log[MSE(m_{cT|T})] = \Omega(W - \log|\mathcal{O}_T|)$*,*
   provided $c > 2$*,*

   (b) $MSE(m_{cT|T}) = \omega(1/|\mathcal{O}_T|)$*, provided* $c = 2$ *and*
   $\sum_{j=1}^{W} j^2\theta_j = \omega(1)$*,*

   (c) $MSE(m_{cT|T}) = \Omega(1/|\mathcal{O}_T|)$*, provided either* $c < 2$ *or* $c = 2$ *and* $\sum_{j=1}^{W} j^2\theta_j = O(1)$.

Proof of Theorem 10.4.2. Our proof of Theorem 10.4.2 shows that the forecasting mixtures of Poisson processes can be mapped into the set size distribution estimation problem; we then use our results on the hardness of estimating set size distributions from sampling [207] to prove the theorem. We start by showing how to map the forecasting mixtures of Poisson processes into the set size distribution estimation problem. Let $P[N_T^{(i)} = k|N_{cT}^{(i)}]$ be the probability that k events happen in the time window $[0, T]$ given the number of events at time cT is $N_{cT}^{(i)}$, $c > 1$. The distribution of $P[N_T^{(i)} = k|N_{cT}^{(i)}]$ is given in the following lemma.

Lemma 10.4.1 *If N_{cT} is the number of events of a Poisson process at time cT and N'_T is the number of events that happened between* $[0, T]$*,* $c > 1$*, then* $P[N'_T = k|N_{cT}] = \binom{N_{cT}}{k} c^{-k}(1 - 1/c)^{N_{cT}-k}$.

Proof: A Poisson process is defined as an arrival process that satisfies the three axioms listed above. Axiom A3 states that $\forall t$, $s > 0$, $N_{t+s} - N_t$ is independent of t. Thus, the counts of the number of events at a time interval $[0, T]$ is independent of the number of events at time interval $(T, cT]$ (axiom A2). Our Poisson process is time-homogeneous and thus the probability that an event lands in $[0, T]$ is proportional to $T/(cT) = 1/c$ (axiom A1). As the number of events between $[0, T]$ and $(T, cT]$ are independent, we have $P[N'_T = k | N_{cT}] = \binom{N_{cT}}{k} c^{-k}(1 - 1/c)^{N_{cT}-k}$. ■

In what follows, we show that the timestamps of the past events in the interval $[0, T]$ are not relevant in the forecast.

Lemma 10.4.2 *The event counts in the historical data provide all the statistical information collected in the interval* $[0, T]$ *w.r.t. the counts* $\{N^{(i)}_{cT}\}_{i=1}^{n}$ *in the interval* $[0, cT]$.

Proof: Axiom A2 states that the exact timestamps of the events in the interval $[0, T]$ gives no statistical information about future events in $(T, cT]$. ■

We are now ready for theorem that connects the forecast with an estimation problem that is known to be hard.

Theorem 10.4.3 *The problem of forecasting any function* $g(\{N^{(i)}_{cT}\}_{i=1}^{n})$ *of the mixture Poisson sample paths described in Section 10.4.2 is equivalent to the set size estimation problem [207], where elements are randomly sampled from a collection of non-overlapping sets and we seek to recover the original set size distribution from the samples.*

Proof: We prove the theorem by mapping the forecasting problem into the set size estimation problem. Let $\{N^{(i)}_{cT}\}_{i=1}^{n}$ be the sizes of n sets, whose elements are sampled independently with probability $1/c > 0$, leading to sampled set sizes that are binomially distributed as in Lemma 10.4.1. Finally, Lemma 10.4.2 shows that these non-zero sampled set sizes are the only information available to forecast the process. ■

Theorem 10.4.3 shows that the present is a biased view of the future, as the set size problem suffers from the *inspection paradox* [207]. In what follows we prove our main theorem.

Proof: [of Theorem 10.4.2] Using Theorem 10.4.3 we can construct a perfect map between the forecasting problem and the set size distribution problem. Theorem 10.4.2 follows from using Theorem 4.3 of our previous work (Murai et al. [207]) to this mapping, where we analytically compute the inverse of the Fisher information matrix of the data, giving rise to the error bounds in the theorem (through the Cramér-Rao bound). ■

Theorem 10.4.2 shows clear limitations of forecasting with models. In realistic scenarios, where event sizes are large and have heavier-than-exponential distributions, no algorithm can obtain accurate unbiased forecasts of the average number of events beyond time interval $[T, 2T]$ if the dataset is limited to time horizon $[0, T]$. Fortunately, SED is not impacted by this problem as it limits its forecasts to the time horizons in the historical data. These results inspire the following apparent paradox.

A Big Network Data Paradox

When More Data Means Worse Forecasts. Theorem 10.4.2 creates an apparent paradox. The forecasting can become more inaccurate as the dataset $\mathcal{O}_T$ grows. The forecasting is not just more inaccurate, the forecast simply *breaks down*; as $\mathcal{O}_T$ grows, and the data time horizon remains the same $[0, T]$, there is nearly zero statistical (Fisher) information to forecast beyond horizon $[T, 2T]$.

To showcase the impact of this paradox, consider a growing network where the distribution of the size of events (cascade sizes) is Pareto with parameter $\beta > 1$. Our results hold generally but for illustration we use a scenario where the number of seeds increases proportionally with network size. In this scenario the relationship between the maximum number of events per seed W and the number of seeds $|\mathcal{O}_T| \gg 1$ is $E[W] \approx \sqrt[\beta]{|\mathcal{O}_T|}\, \Gamma(\beta - 1)$, a known result from extreme value theory [208]. Theorem 10.4.2 case 3(a) shows that the MSE error is lower bounded by $\log[\text{MSE}(m_{cT|T})] = \Omega(W - \log |\mathcal{O}_T|)$. As $E[W] \propto \sqrt[\beta]{|\mathcal{O}_T|}$ and $|\mathcal{O}_T|$ increases, so does W. We can think of the MSE lower bound growing roughly as $\exp(\sqrt[\beta]{|\mathcal{O}_T|})/|\mathcal{O}_T|$.

Thus, if the training data $\mathcal{O}_T$ has time horizon $[0, T]$ and event (cascade) sizes have a power law distribution, Theorem 10.4.2 cases 3(b-c) show that as the network grows the estimate in the interval $[T, 2T]$ gets more accurate. However, Theorem 10.4.2 case 3(a) shows that as the network grows the estimation error in the interval $(2T, cT]$, $c > 2$, grows exponentially with network size.

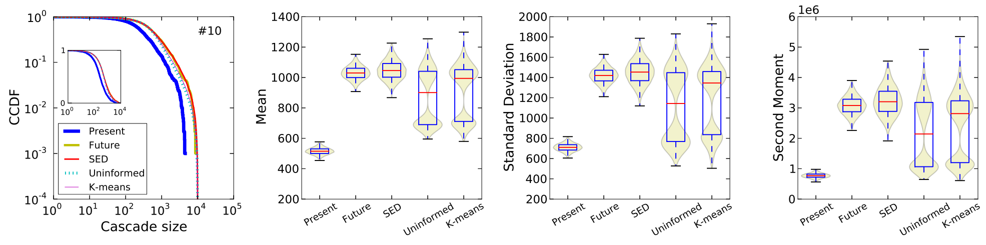
10.5 Results

In this section we present our results. We start with our simulation results in Section 10.5.1. Section 10.5.2 shows our results over a large Twitter dataset.

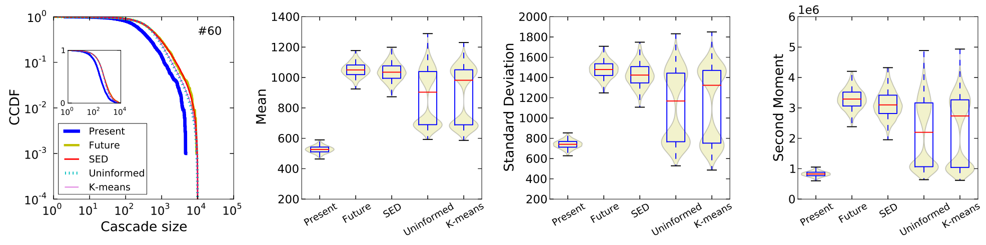
10.5.1 Forecasting Using Simulated Data

In this section, we test the forecasts of SED on synthetic datasets. We test the performance of SED over two distinct models: the Poisson process and the Galton-Watson process. Recall that $\Omega' = \Omega \cup \{\Psi^{(0)}\}$ is the union of the historical processes Ω and the process to forecast $\Psi^{(0)}$. SED forecasts metrics of $\Psi^{(0)}$ at time interval Δt_2 using data from time interval Δt_1, $\Delta t_1 < \Delta t_2$ In addition to SED, we also have two baseline naive predictors: *uniformed prediction*, and *K-means prediction*. Uninformed prediction returns the bootstrapped statistics at time interval Δt_2 of a randomly selected process in Ω. Whereas, K-means prediction first clusters the set of historical cascades using K-means algorithm and Euclidean distances between the CCDFs of cascade metrics (probability density functions, PDFs, give similar results). The number of cluster k is chosen using the elbow method, without making the clusters too small (k equals 10 for the two synthetic datasets, and 20 for the Twitter dataset). After that, given a new cascade $\Psi^{(0)}$, K-means prediction finds the cluster it belongs to, and returns the bootstrapped statistics at time interval Δt_2 of a randomly selected process in that cluster.

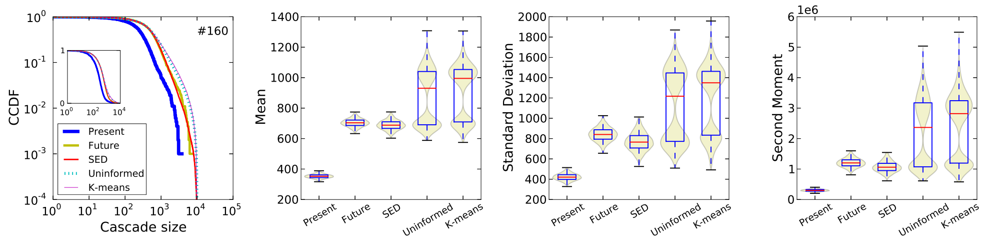
For evaluation, we use violin plots to compare the predicted statistics with the bootstrap statistics from the ground-truth empirical distribution of $\Psi^{(0)}$ at Δt_2. A violin plot is a box plot combined with a kernel density estimate of the probability density function, giving a more detailed view of the data's variance. The box extends from the first quartile Q_1 to the third quartile Q_3 of the data, with a line at the median. The whiskers extend from the box to $Q_1 - 1.5 * \text{IQR}$ and $Q_3 + 1.5 * \text{IQR}$, where $\text{IQR} = Q_3 - Q_1$ is the inter-quartile range. The violin plots show the spread.



(a) ID 10

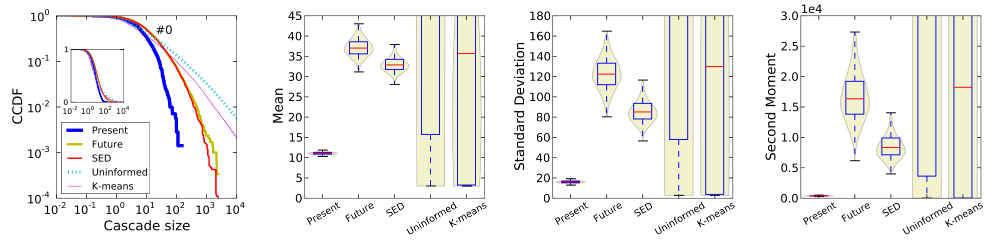


(b) ID 60

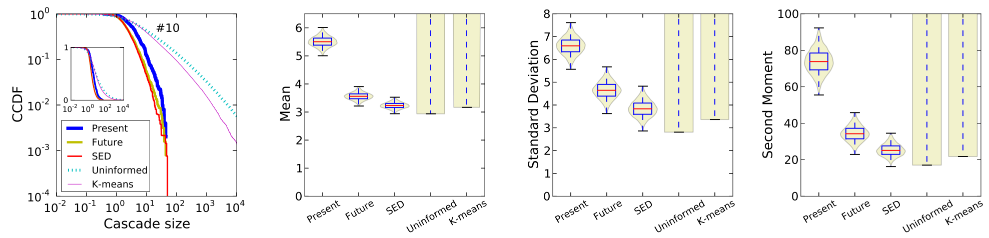


(c) ID 160

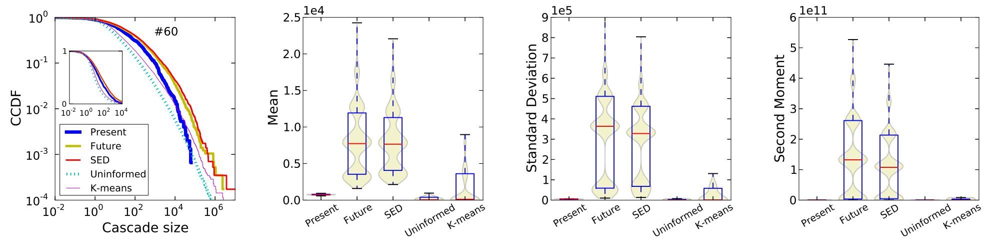
Figure 10.1: Prediction for mixture Poisson processes (Scenario 1) of the future CCDF, average, standard deviation, and second moment metrics. SED predictions match well the true future distributions and clearly outperform uninformed predictions.



(a) ID 0



(b) ID 10



(c) ID 60

Figure 10.2: Prediction for Galton-Watson processes (Scenario 2) of the future CCDF, average, standard deviation, and second moment metrics. SED predictions again match well the future distributions and are clearly superior to uninformed predictions.

Scenario 1: Mixture Poisson Processes

Our mixture Poisson process is a Poisson process with random rate β that has support $(0, W]$, $W > 0$.

Data Generation: The simulated historical data $\mathcal{H}$ is generated through simulations from time zero until $\Delta t_2 = 1$ month from two sets of mixture Poisson processes. The first set has $n_1 = 100$ processes with rate distribution $\gamma_1 \propto \beta^{-2}$ and the second set has $n_2 = 100$ processes with rate distribution $\gamma_2 \propto \beta^{-3}$. Poisson rates are measured in number of events per month. Each process has $1,000$ infection seeds. The process to forecast $\Psi^{(0)}$ belongs to one of two sets and was observed from time zero until $\Delta t_1 = \Delta t_2/4$, approximately one week. In what follows we test how well SED can forecast $\Psi^{(0)}$. In Section 10.4.2

we show that estimating the Poisson rates is hard due to observation bias.

Results: For conciseness, Figure 10.1 shows only the results for three processes, one per column: processes 10, 60 (type 1), and 160 (type 2). Four forecasts are evaluated: forecasting the CCDFs, mean, standard deviation, and second moment. For comparison baselines we use uninformed prediction – uniform matching weights – and matching using K-means with Euclidean distances, as the latter have been extensively and successfully used in mining and classifying time series [196]. For each process we present the true statistics at Δt_1 (Present) and Δt_2 (Future), and the predicted statistics at Δt_2, including SED, uninformed, and K-means predictions. The Figure 10.1 shows that SED is the method that better represents, consistently, the true statistics at time Δt_2 under all scenarios (Future).

Scenario 2: Galton-Watson Process

The Galton-Watson process is a branching process $\{X_n\}$ such that $X_{n+1} = \sum_{j=1}^{X_n} \psi_j^{(n)}$, where $X_0 = 1$ and $\{\psi_j^{(n)} \sim Poisson(\lambda) : n, j \in \mathbb{N}\}$ is a set of i.i.d. natural number-valued random variables [209]. Intuitively, $\psi_i^{(n)}$ is the number of male children of the j-th descendants, and X_n is the number of descendants in the n-th generation. The total number of people with the considered family name at time t would be $\sum_{i=0}^{t} X_t$. A simple epidemic cascading on a network can be modeled as a Galton-Watson process.

To make this model harder to forecast, we replace the Poisson distribution of the number of children by a log-normal distribution. The purpose of the log-normal distribution is to skew the obtained distribution of the number of children, mimicking real life situation where the degree of a node in a network follows a heavy tail distribution. We call the values generated by the log-normal distribution the descendant effect of a node. Here is the model we use $X_{n+1} = \sum_{j=1}^{\lfloor X_n \rfloor} \psi_j^{(n)}$, where $X_0 = 1$ and $\{\psi_j^{(n)} \sim \text{log-normal}(\mu, \sigma^2) : n, j \in \mathbb{N}\}$. The operation $\lfloor X_n \rfloor$ is to get an integer-valued number of descendants.

Cascade Process Evolution: For each information cascade, we use a birth rate γ to generate new seeds over time. Given a single seed node ($X_0 = 1$), the total descendant effect up to time t in a Galton-

		Parameters			Temporal delta		
Process Type	IDs	μ_0	σ_0	γ_0	Δ_μ	Δ_σ	Δ_γ
Small-inc	0..9				+0.2	+0.1	+30
Small-dec	10..19	1.5	1	300	-0.2	-0.1	-30
Small-const	20..29				0	0	0
Medium-inc	30..39				+0.2	+0.1	+30
Medium-dec	40..49	2	2	500	-0.2	-0.1	-30
Medium-const	50..59				0	0	0
Large-inc	60..69				+0.2	+0.1	+30
Large-dec	70..79	3	2.5	700	-0.2	-0.1	-30
Large-const	80..89				0	0	0

Table 10.1: Galton-Watson process simulation parameters.

Watson process, i.e., $\sum_{i=0}^{t} X_i$, will be used as the cascade size at time t for this seed. As a result, we obtain a sample of the cascade size per seed. To add more complexity to the model, we allow the parameters to change over time: $\mu_t = \mu_0 + t * \Delta_\mu$, $\sigma_t = \sigma_0 + t * \Delta_\sigma$, $\gamma_t = \gamma_0 + t * \Delta_\gamma$, where μ_t, σ_t, γ_t are the parameter values at time point t, and Δ_μ, Δ_σ, Δ_γ are the amount of change at each time step.

Table 10.1 shows our parameter settings. We create 9 different cascade types, each one containing 10 cascades spanning a period of seven time units. These cascade types can be grouped into three big groups – small, medium and large – based on their mean μ in the log-normal distribution. Larger cascades have a larger variation (larger σ) and higher birth rate γ. For each of these big groups, we generate three subgroups with different evolutionary trends: increasing in size (inc), decreasing in size (dec), and constant size (const). Finally, we set $\Delta t_1 = 2$ time units and $\Delta t_2 = 7$ time units.

Results: Figure 10.2 shows the results of three processes: 0 (Small-inc), 10 (Small-dec) and 60 (Large-inc). We get similar results in the other processes. Note that the statistics of Present and Future are significantly different. Figure 10.2 shows that the uninformed and k-means forecasts are far from the truth that the confidence intervals do not fit in our plot. Over the same scenario SED accurately forecasts the mean, standard deviation and second moment as shown in the last three rows of Figure 10.2.

10.5.2 Forecasting Twitter Cascades

We study the Twitter dataset collected by Yang et al. [58][2], which contains 467 million Twitter posts from 20 million users covering a 7-month period from June 1st to December 31st, 2009. The underlying Twitter follower-followee network is obtained from Kwak et al. [210]. The fact that this Twitter dataset is only a sample of all tweets does not invalidate the dataset to test SED. SED is designed to make predictions over general dissemination processes, including sampled cascading processes such as our Twitter dataset.

Hashtag cascades

We are interested in the diffusion of a hashtag over the Twitter online social network. When an individual uses a hashtag in her tweet at a given time, if neither she nor any of her followees have tweeted the same hashtag for at least a week prior to the post timestamp, we consider her as a seed. The timestamp of the first tweet is the timestamp of a node hashtag "infection". Once the seed of a cascade has been identified, we start tracking the cascade over time. Note that our definition of seed is stringent and we may merge almost unrelated cascades into a single cascade. From our RMPP definition, metrics over merged cascade processes are allowed and, thus, we can be conservative to define independent cascades in our dataset.

If a hashtag was not been used for more than a week, we assume the cascade has ended. The first occurrence of the same hashtag after that would mark the beginning of a new cascade. Due to this one-week time window, hashtags appearing for the first time in the dataset during the first and last weeks of the monitored time frame are ignored. The time difference between the first and the last posts of an information cascade is defined as the duration of the cascade.

Cascade metric

In our case study, we are interested in the distribution of the number of infected descendants of a seed node for each hashtag in Twitter. In particular, given an infected seed node, we quantify how

[2] `http://snap.stanford.edu/data/twitter7.html`

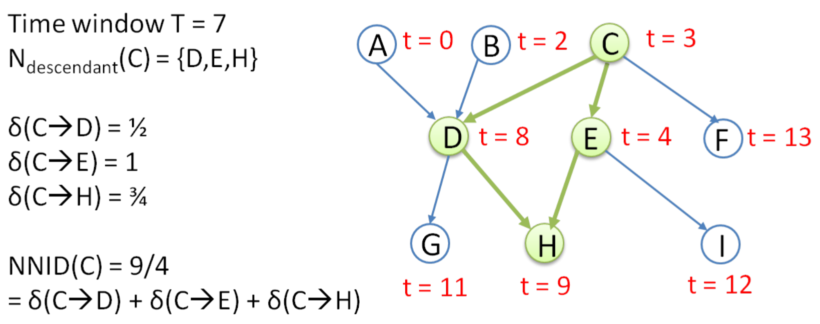


Figure 10.3: An example induced subgraphs of infected nodes and cascade metric. Node infection timestamps are shown in red.

many of its descendants are "infected" by a hashtag complying with temporal infection order (infected children in the branching process must be infected after their parents). These quantities are obtained from the induced subgraph of the infected nodes over the original follower-followee network. Note that a user may have several followees who posted the same hashtag before she did. As a result, simply counting the number of infected descendants of an infected user does not guarantee the independence between samples required by our theory. To reduce the effect of these independence violations, we assume that two infected parents of an infected node are evenly responsible for the infection. Under this assumption we propose a normalized metric as follows.

Definition 10.5.1 *Given a time window length T, the number of normalized infected descendants (NNID) of a node v are defined as:*

$$NNID(v) = \sum_{w \in N_{des}^{T}(v)} \delta(v \to w)$$

where $N_{in}^{T}(v)$ and $N_{des}^{T}(v)$ are the set of infected parents, and infected descendants of node v within the time window T respectively; $\delta(v \to v) = 1$; $\delta(v \to w)$ is the amount of infection spread from node v to w through all possible time-ordering paths, and is defined recursively as follows:

$$\delta(v \to w) = \frac{\sum_{u \in N_{in}^{T}(w) \cap N_{des}^{T}(v)} \delta(v \to u)}{|N_{in}^{T}(w)|}$$

We assume that if a node v is infected at time point t, then after time point $t + T$, its effect on other nodes will diminish to zero. Thus, the set of infected descendants of node v do not contains nodes infected after time $t + T$. To get a distribution of NNIDs for a cascade c at a given time point t, we first extract the induced subgraph $G_{ind} = (V_{ind}, E_{ind})$, where V_{ind} is the set of infected nodes of c up to time t. In addition, given two infected nodes u and v, if the edge $(u \to v) \in E$ and $t_{infect}(v) - t_{infect}(v) \leq T$, then $(u \to v) \in E_{ind}$, where $t_{infect}(v)$ is the timestamp at which v is infected in c. Next, for each seed node of the cascade, i.e., nodes with no infected parents within a time window T, we compute its NNID in G_{ind} using Definition. 10.5.1. In the end, we obtain a sample of NNID values, each for a seed in c, giving us an empirical distribution of NNIDs of c.

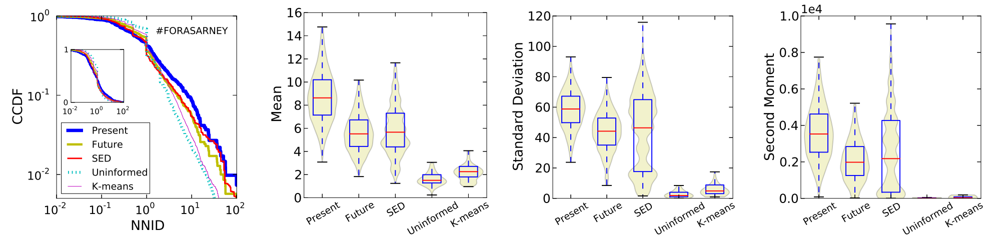
Example 10.5.1 *An induced subgraph of infected nodes is shown in Fig. 10.3. Given a time window $T = 7$, the descendant set of C is $\{D, E, H\}$; the infected parent sets of D and E are $\{B, C\}$ and $\{C\}$ respectively. Thus, $\delta(C \to D) = \frac{1}{2}$, and $\delta(C \to E) = 1$. The infection from C can spread to H following two different paths: $C \to D \to H$, and $C \to E \to H$. Thus, $\delta(C \to H) = \frac{\delta(C \to D) + \delta(C \to E)}{2} = \frac{3}{4}$. Finally, the number of normalized descendants of C is $NNID(C) = \delta(C \to D) + \delta(C \to E) + \delta(C \to H) = \frac{1}{2} + 1 + \frac{3}{4} = \frac{9}{4}$.*

Similarly, the set of seed nodes with no infected parents within a time window $T = 7$ is $\{$A, B, C, F, I$\}$, which gives a sample of NNIDs $\{1, \frac{3}{4}, \frac{9}{4}, 1, 1\}$ accordingly.

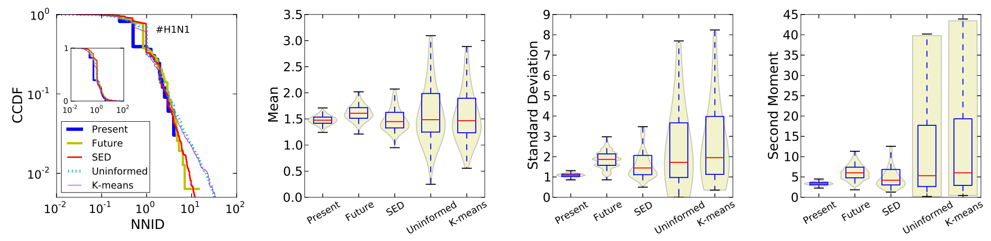
Twitter Results

For our tests, we identify the Twitter hashtag cascades that last at least 8 weeks. Using this dataset, we predict distribution of NNID (time window $T = 1$ week) at week 8 (Δt_2) using data from the week 2 (Δt_1). To guarantee that enough data is available for prediction, we only consider cascades that contain at least 300 infected users after 8 weeks and at least 30 infected users after one week. In the end, we are left with 1050 cascades split among the different hashtags. Similar to the simulations, besides the CCDF plots, we use violin plots to compare the predicted statistics (SED prediction and uninformed prediction) with the bootstrap statistics from the ground-truth sample at Δt_2. It is worth noting that the uninformed prediction simply returns the average statistics of all cascades in Ω at time Δt_2.

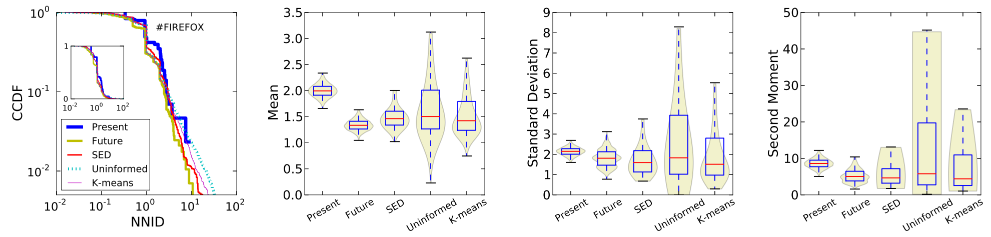
The forecasting results for five Twitter hashtags are shown in Figure 10.4. SED forecasts frequently outperform uninformed and k-means predictions. For example, all #FORASARNEY statistics differ



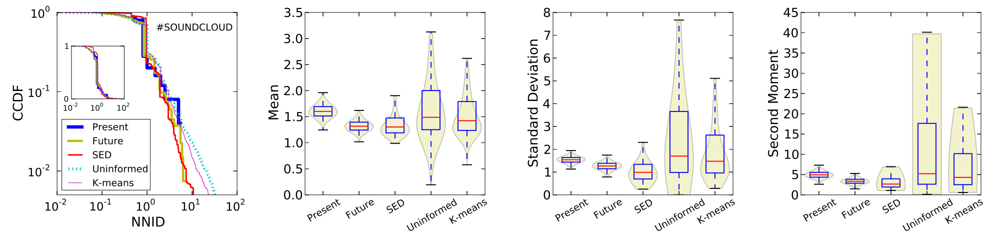
(a) FORASARNEY



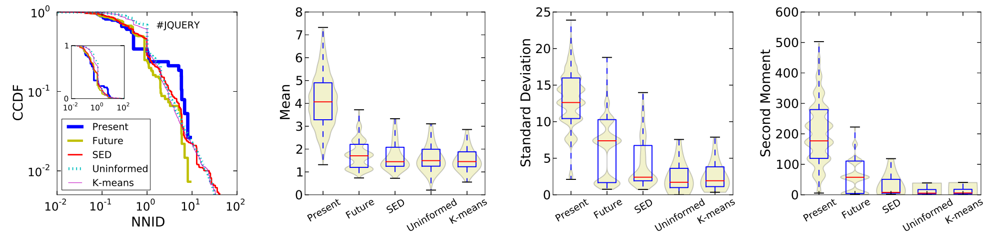
(b) H1N1



(c) FIREFOX



(d) SOUNDCLOUD



(e) JQUERY

Figure 10.4: Predictions of statistics of NNID for five example hashtags in Twitter.

considerably from those of uninformed and k-means predictions. Whereas SED accurately forecasts the average CCDFs and all three statistics, as observed by contrasting SED forecast violin plots against those of the ground truth (Future). For #H1N1, #FIERFOX and #SOUNDCLOUD, the uninformed forecasts are rather uncertain of the values of the mean, the standard deviation, and the second moment, leading to significantly lengthened violin plots. On the other hand, the SED forecasts are, once again, close to the ground truth. In the case of forecasting the mean of #JQUERY (last column, second row), the true mean (Future) is so close to the future average statistics (uninformed prediction), that SED shows no clear improvement over naive approaches. We include this #JQUERY example to show that in hard-to-forecast cases SED does no worse than uninformed and k-means. Note, however, that for standard deviation and the second moment of #JQUERY show SED superior to uninformed and k-means.

It must be noted that the hashtag Twitter dataset that we are using is noisy (cascades are not fully independent), making our forecasting task very difficult. Moreover, for many of the cascades in this dataset, the distribution at week 8 (Δt_2) is very close to the average prediction (uninformed prediction). Thus, more challenging datasets and further research are needed to further validate practical aspects of our approach.

10.6 Related Work

Analyzing and predicting network processes in general, and information cascades in particular, has attracted much attention in recent years. Multiple works focus on building theoretical models (e.g., [211–213]) of dissemination processes. These models capture information diffusion at nodes through network topology as well as user interests and the information content [211, 214, 215]. Wang et al. [216] uses partial differential equations to predict information diffusion over both temporal and spatial dimensions. One challenge in this line of research is the gap between empirical data and theory. Building parsimonious models is a complex task and the resulting models are often not vetted against real data.

Most relevant to this paper are the works that predict cascade popularity. Some of these works predict the volume of aggregate activity, such as the number of votes on Digg stories [165], or Twitter hashtag usage [193]. A few works examine how information cascades grow in size depending on its content [192,217] and user interests [218]. Other works make prediction using observations of a cascade during a given fixed time interval [193,219,220]. The cascade size prediction task is seen as a regression problem in a variety of works [165, 219–221]. Matsubara et al. [194] proposes a parsimonious model that captures the rise and fall patterns in information diffusion. In lieu of predicting the exact cascade sizes, many studies bin the cascades sizes and solve a binary classification problem of whether or not a cascade becomes viral [166, 197, 219] or if it doubles in size [169]. Instead of modeling the diffusion process, Najar et al. [222] directly predicts the final propagation state of the information given its initial state. Outside cascade sizes, Cheng et al. [169] also predicts structural features of the cascade sample paths. Our goal is different as we do not perform binary classification of the evolution of a sample path, or just predict cascade sizes. We propose a general axiomatic forecasting framework that is not tied to a specific set of cascade features or process and can be used off-the-shelf in any similar forecasting scenario.

Since we predict cascade statistics, our work also relates to research on fitting empirical data to parsimonious statistical models [223, 224]. While the empirical data can be readily fitted to many known parsimonious models such as power laws, log-normal, or exponential, there is no guarantee that the fitted model can be used to predict the tail of the distribution or how the distribution changes with the observation window. Indeed, our experimental results using Twitter data show that cascade statistics can significantly change over time. Thus, the need to go beyond parsimonious models to design a data-driven statistical approach using axiomatic forecasting.

10.7 Conclusion

In this work we propose SED, a new algorithm for forecasting statistics of complex networked processes. SED is the first of its kind, a (true) model-free approach that uses axioms rather than models to extract statistical information from the data, pointing to a promising new direction of axiomatic forecasting. More importantly, we provide the underlying theory behind SED's model-free axiomatic forecasting approach. We test SED on two synthetic datasets and a large Twitter dataset, showing that SED can forecast accurately a variety of statistics under complex scenarios.

10.8 Acknowledgment

We thank C. Faloutsos and D. Towsley for their helpful comments. This work was partially supported by NSF grants CNS-1065133, IIS-1219254, and ARL Cooperative Agreement W911NF-09-2-0053. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied of the NSF, ARL, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

Chapter 11

Conclusions and future work

11.1 Conclusions

In this thesis, I studied the network effect in real-world data and utilize the network structures for different mining and predicting tasks. Network structures exist in many types of data, and guide different networked behaviors in these data, including the evolution of the data over time and various types of network processes. The existence of the network structures require new metrics and methods for analyzing, mining, and modeling networked behaviors.

Part I explored collaboration networks to gain insights into its structure and its evolution over time. Specifically, I used simplicial complex as a new representation for networks, devised a number of new metrics for evaluating the effectiveness of collaborations, and found insightful observations by tracking these metrics over time. In addition, I studied the evolution of collaboration networks by investigating the egonets and defined rising stars as those whose characteristics change suddenly compared to both local and global trends. Again, I discovered many insights into how these rising stars interact with each other over time. There are a number of interesting avenues for further exploration: a more in-depth analysis of the results, and better support for our explanations; a similar study of other data sets; an efficient generalized algorithm for computing k-MNFs and persistence across time; and relationships with other features of simplicial complex, for e.g Betti numbers.

In Part II, I mined different types of network patterns by proposing three mining problems for a single large graph, a set of graph snapshots, and a database of graphs. The subgraph search space is exponential in the size of the network, making it infeasible to be stored, or computed in its entirety. Therefore, I employed sampling to navigate in this space and devised heuristics to prune the search space. Furthermore, many mining problems become NP-hard in network settings, rendering it infeasible to be solve in large scale. As a result, I proposed a number of approximated algorithms, both with and without quality guarantees, for these mining problems.

Finally, in Part III, I utilized the network structures for three prediction tasks in a traffic network and online social networks. Specifically, the traffic network are created based on the traffic patterns between different regions of a city. After that, this network is used in a Bayesian network to predict the instantaneous changes of traffic flows, in addition to the seasonal and trend patterns. For online social networks, I predicted information spreads, such as the spread of hashtags on Twitter, and the spread of likes on projects in Behance.net. I solved this prediction task using two approaches: with and without models. Both of these approaches have their own strengths and outperformed other baselines in the corresponding tasks.

All in all, I tackled all three steps in working with network-based data: analyzing, mining, and predicting. Since this is a broad area, there are still many open questions that need answers.

11.2 Promising future directions

11.2.1 Analyzing and mining networked behaviors

Besides the three types of patterns proposed in this thesis, there are also other possibilities, including probabilistic, evolutional, and sentiment-aware network patterns, among others. Such patterns can provide a stronger modeling and predicting power.

- **Probabilistic network patterns:** Networked behaviors are often noisy. For example, the activities of users in online social networks are subject to a lot of exogenous factors that are not available to us, such as the randomness in users' behaviors and real-life events. Therefore, instead of exact matching of network patterns, I can assign probability to different elements in a pattern, indicating how likely a node or an edge will participate in a network processes. Such a probabilistic definition would be more realistic and create more matches between patterns and network processes.

- **Evolutional network patterns:** Networks change over time. Thus, finding patterns for a single snapshot of a network at a certain time is not enough. These patterns should be allowed to change over time as well, i.e., additions and deletions of nodes and edges. Mining these evolutional patterns is key to modeling the evolution of a network over time.

- **Sentiment-aware network patterns:** Besides nodes and edges, networks may also have additional information on these nodes and edges. Including these information in the patterns is thus a natural next step. For example, information spreads in online social networks are largely affected by the sentiments coded in the messages being spread. Different sentiments corresponding to different topics intrigue different sets of users. As a consequence, the same network pattern might be active or inactive in different scenarios.

11.2.2 Modeling and predicting networked behaviors

The new types of patterns in Section 11.2.1 opens opportunities to new models that utilize them. I mention here a number of potential problems in this direction.

- **Data-driven modeling:** The patterns found in Section 11.2.1 can be used as basis subgraphs for data-driven models, in lieu of parametric models. In particular, each network process can be represented as a set of basis subgraphs, each of which can evolve over time, or include additional side information on nodes and edges. Such an approach is feasible thanks to the availability of a large amount of data today.

- **Predicting unexpected events:** All of the proposed methods for prediction in this thesis are based on the assumption that new events are similar to some historical events. Thus, I can build a model from historical data, and use it to predict the future. However, many events are unexpected in real life, leading to poor descriptive power if only past data is taken into account. Moreover, data for rare events are sparse, making it even harder to model. In such case, one possible solution is to make certain assumptions on the evolution of unexpected events. Nevertheless, this is an open direction with many challenges.

- **Manipulating networked behaviors:** Besides mining patterns and modeling networked behaviors, another important task is to manipulate them. This task has applications in many real-world scenarios, such as, controlling the spread of fake news online, maximizing the influence of a targeted ad campaign, and designing networks (adding and deleting nodes and edges) to achieve some optimization goals. The mined patterns and the models proposed in this thesis can be used as the basis for these manipulating tasks.

Bibliography

[1] P. Miettinen, T. Mielikäinen, A. Gionis, G. Das, and H. Mannila, *The discrete basis problem*, *TKDE* (2008).

[2] X. Yan and J. Han, *gspan: Graph-based substructure pattern mining*, in *ICDM*, 2002.

[3] Y. Matsubara, Y. Sakurai, *et. al.*, *Fast mining and forecasting of complex time-stamped events*, in *KDD*, 2012.

[4] H. Pinto, J. M. Almeida, and M. A. Gonçalves, *Using early view patterns to predict the popularity of youtube videos*, in *WSDM*, 2013.

[5] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, *SIAM review* **51** (2009), no. 3.

[6] G. W. Milligan and M. C. Cooper, *An examination of procedures for determining the number of clusters in a data set*, *Psychometrika* **50** (1985), no. 2 159–179.

[7] L. Qin, J. X. Yu, and L. Chang, *Diversifying top-k results*, *PVLDB* **5** (2012), no. 11 1124–1135.

[8] J. J. Ramasco, S. N. Dorogovtsev, and R. Pastor-Satorras, *Self-organization of collaboration networks*, *Phys. Rev. E* **70** (Sep, 2004).

[9] J. D. Adams, G. C. Black, J. R. Clemmons, and P. E. Stephan, *Scientific teams and institutional collaborations: Evidence from us universities, 1981–1999*, *Research Policy* **34** (2005), no. 3 259–285.

[10] S. Wuchty, B. F. Jones, and B. Uzzi, *The increasing dominance of teams in production of knowledge*, *Science* **316** (2007), no. 5827 1036–1039.

[11] A. Barabâsi, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek, *Evolution of the social network of scientific collaborations*, *Physica A: Statistical Mech. and its Applications* **311** (2002), no. 3 590–614.

[12] R. Atkin, *Mathematical structure in human affairs*. Heinemann Educational London, 1974.

[13] R. Ramanathan, A. Bar-Noy, P. Basu, M. Johnson, W. Ren, A. Swami, and Q. Zhao, *Beyond graphs: Capturing groups in networks*, in *Proc. IEEE NetSciComm*, pp. 870–875, IEEE, 2011.

[14] T. Moore, R. Drost, P. Basu, R. Ramanathan, and A. Swami, *Analyzing collaboration networks using simplicial complexes: A case study*, in *Proc. IEEE NetSciComm*, pp. 238–243, IEEE, 2012.

[15] C. Berge, *Hypergraphs.* North-Holland, 1989.

[16] E. Spanier, *Algebraic topology*, vol. 55. Springer, 1994.

[17] "DBLP computer science bibliography, http://www.informatik.uni-trier.de/$\sim$ley/db/."

[18] "Internet movie database (IMDB), http://www.imdb.com/."

[19] A. Clauset, C. Shalizi, and M. Newman, *Power-law distributions in empirical data*, *SIAM review* **51** (2009), no. 4 661–703.

[20] J. Leskovec, J. Kleinberg, and C. Faloutsos, *Graphs over time: densification laws, shrinking diameters and possible explanations*, in *Proc. of the 11th ACM SIGKDD*, pp. 177–187, ACM, 2005.

[21] G. R. Lopes, M. M. Moro, L. K. Wives, and J. P. M. De Oliveira, *Collaboration recommendation on academic social networks*, in *Advances in Conceptual Modeling–Applications and Challenges*, pp. 190–199. Springer, 2010.

[22] M. E. J. Newman, *The structure of scientific collaboration networks*, *Proc. National Academy of Sciences* **98** (2001), no. 2 404–409.

[23] P. Zhang, K. Chen, Y. He, T. Zhou, B. Su, Y. Jin, Y. Chang, H. an d Zhou, L. Sun, B. Wang, *et. al.*, *Model and empirical study on some collaboration networks*, *Physica A: Statistical Mechanics and its applications* **360** (2006), no. 2 599–616.

[24] L. Amaral, A. Scala, M. Barthélémy, and H. Stanley, *Classes of small-world networks*, *Proceedings of the National Academy of Sciences* **97** (2000), no. 21 11149–11152.

[25] M. Newman, *Power laws, pareto distributions and zipf's law*, *Contemporary physics* **46** (2005), no. 5 323–351.

[26] M. Goldstein, S. Morris, and G. Yen, *Problems with fitting to the power-law distribution*, *The European Physical Journal B-Condensed Matter and Complex Systems* **41** (2004), no. 2 255–258.

[27] P. Gould and A. Gatrell, *A structural analysis of a game: the Liverpool v Manchester United Cup Final of 1977*, *Social Networks* **2** (1980), no. 3 253–273.

[28] H.-H. Chen, L. Gou, X. Zhang, and C. L. Giles, *Collabseer: A search engine for collaboration discovery*, in *Proc. of the 11th International ACM/IEEE Joint Conference on Digital Libraries*, pp. 231–240, 2011.

[29] J. Tang, S. Wu, J. Sun, and H. Su, *Cross-domain collaboration recommendation*, in *Proc. of the 18th ACM SIGKDD*, KDD '12, pp. 1285–1293, 2012.

[30] M. Franceschet, *Collaboration in computer science: A network science approach*, *Journal of the American Society for Information Science and Technology* **62** (2011), no. 10 1992–2012.

[31] L. Bennett, H. Gadlin, and S. Levine-Finley, *Collaboration and team science: A field guide*, *Bethesda, MD: National Institutes of Health* (2010).

[32] S. Wasserman and K. Faust, *Social network analysis: Methods and applications*, vol. 8. Cambridge university press, 1994.

[33] "The Network Science CTA, http://www.ns-cta.org/."

[34] J. Hirsch, *An index to quantify an individual's scientific research output*, *Proceedings of the National Academy of Sciences of the United states of America* **102** (2005), no. 46 16569.

[35] D. Stokols, K. L. Hall, B. K. Taylor, and R. P. Moser, *The science of team science: overview of the field and introduction to the supplement*, *Am J Prevent Med* **35** (2008), no. 25 77–89.

[36] T. Lappas, K. Liu, and E. Terzi, *Finding a team of experts in social networks*, in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 467–476, ACM, 2009.

[37] R. T. Sparrowe, R. C. Liden, S. J. Wayne, and M. L. Kraimer, *Social networks and the performance of individuals and groups.*, *Academy of management journal* **44** (2001), no. 2 316–325.

[38] P. Balkundi and D. A. Harrison, *Ties, leaders, and time in teams: Strong inference about network structure's effects on team viability and performance.*, *Academy of Management Journal* **49** (2006), no. 1 49–68.

[39] A. Woolley, C. Chabris, A. Pentland, N. Hashmi, and T. Malone, *Evidence for a collective intelligence factor in the performance of human groups*, *Science* **330** (2010), no. 6004 686–688.

[40] L. Akoglu, M. McGlohon, and C. Faloutsos, *Anomaly detection in large graphs*, in *In CMU-CS-09-173 Technical Report*, Citeseer, 2009.

[41] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos, *Neighborhood formation and anomaly detection in bipartite graphs*, in *Data Mining, Fifth IEEE International Conference on*, IEEE, 2005.

[42] W. Eberle and L. Holder, *Discovering structural anomalies in graph-based data*, in *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, IEEE, 2007.

[43] C. C. Noble and D. J. Cook, *Graph-based anomaly detection*, in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2003.

[44] J. Wu, Z. Guan, Q. Zhang, A. K. Singh, and X. Yan, *Static and dynamic structural correlations in graphs*, *Knowledge and Data Engineering, IEEE Transactions on* **25** (2013), no. 9.

[45] A. N. Langville and C. D. Meyer, *Google's PageRank and beyond: the science of search engine rankings*. Princeton University Press, 2011.

[46] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, *Lof: identifying density-based local outliers*, in *ACM Sigmod Record*, vol. 29, ACM, 2000.

[47] K. Henderson, B. Gallagher, L. Li, L. Akoglu, T. Eliassi-Rad, H. Tong, and C. Faloutsos, *It's who you know: graph mining using recursive structural features*, in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2011.

[48] L. Akoglu and C. Faloutsos, *Event detection in time series of mobile communication graphs*, in *Army Science Conference*, 2010.

[49] R. Rossi, B. Gallagher, J. Neville, and K. Henderson, *Role-dynamics: fast mining of large dynamic networks*, in *Proceedings of the 21st international conference companion on World Wide Web*, ACM, 2012.

[50] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu, *Graphscope: parameter-free mining of large time-evolving graphs*, in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2007.

[51] X. Li, Z. Li, J. Han, and J.-G. Lee, *Temporal outlier detection in vehicle traffic data*, in *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, IEEE, 2009.

[52] C. C. Aggarwal, Y. Zhao, and P. S. Yu, *Outlier detection in graph streams*, in *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, IEEE, 2011.

[53] W. Yu, C. C. Aggarwal, S. Ma, and H. Wang, *On anomalous hotspot discovery in graph streams*, in *ICDM*, 2013.

[54] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, *An analysis of approximations for maximizing submodular set functions*, *Mathematical Programming* **14** (1978), no. 1 265–294.

[55] U. Feige, *A threshold of ln n for approximating set cover*, *Journal of the ACM (JACM)* **45** (1998), no. 4 634–652.

[56] A.-L. Barabási and R. Albert, *Emergence of scaling in random networks*, *science* **286** (1999), no. 5439 509–512.

[57] H. Kwak, C. Lee, H. Park, and S. Moon, *What is twitter, a social network or a news media?*, in *WWW*, 2010.

[58] J. Yang and J. Leskovec, *Patterns of temporal variation in online media*, in *WSDM*, 2011.

[59] "Behance.net social network." `https://www.behance.net/`, 2014.

[60] S. Maurus and C. Plant, *Ternary matrix factorization*, in *ICDM*, 2014.

[61] X. Yan and J. Han, *Closegraph: mining closed frequent graph patterns*, in *KDD*, 2003.

[62] M. Al Hasan and M. J. Zaki, *Output space sampling for graph patterns*, *Proceedings of the VLDB Endowment* **2** (2009), no. 1 730–741.

[63] D. Natarajan and S. Ranu, *A scalable and generic framework to mine top-k representative subgraph patterns*, in *ICDM*, 2016.

[64] T. K. Saha and M. Al Hasan, *Fs3: A sampling based method for top-k frequent subgraph mining*, *Statistical Analysis and Data Mining: The ASA Data Science Journal* **8** (2015), no. 4 245–261.

[65] A. R. Benson, D. F. Gleich, and J. Leskovec, *Higher-order organization of complex networks*, *Science* **353** (2016), no. 6295 163–166.

[66] M. Koyutürk and A. Grama, *Proximus: a framework for analyzing very high dimensional discrete-attributed datasets*, in *KDD*, 2003.

[67] H. Lu, J. Vaidya, and V. Atluri, *Optimal boolean matrix decomposition: Application to role engineering*, in *ICDE*, 2008.

[68] P. Miettinen, *Sparse boolean matrix factorizations*, in *ICDM*, 2010.

[69] P. Miettinen and J. Vreeken, *Model order selection for boolean matrix factorization*, in *KDD*, 2011.

[70] P. Miettinen, *Dynamic boolean matrix factorizations*, in *ICDM*, 2012.

[71] P. Miettinen, *Generalized matrix factorizations as a unifying framework for pattern set mining: Complexity beyond blocks*, in *ECML-PKDD*, 2015.

[72] P. Miettinen and J. Vreeken, *mdl4bmf: Minimum description length for boolean matrix factorization*, *TKDD* (2014).

[73] S. Karaev *et. al.*, *Getting to know the unknown unknowns: Destructive-noise resistant boolean matrix factorization*, in *SIAM*, 2015.

[74] C. Lucchese, S. Orlando, and R. Perego, *Mining top-k patterns from binary datasets in presence of noise*, in *SIAM*, 2010.

[75] C. Lucchese, S. Orlando, and R. Perego, *A unifying framework for mining approximate top-k binary patterns*, *TKDE* (2014).

[76] M. Araujo, P. Ribeiro, and C. Faloutsos, *Faststep: Scalable boolean matrix decomposition*, in *PAKDD*, 2016.

[77] S. Ravanbakhsh, B. Poczos, and R. Greiner, *Boolean matrix factorization and noisy completion via message passing*, *ICML* (2016).

[78] E. Acar *et. al.*, *All-at-once optimization for coupled matrix and tensor factorizations*, *arXiv preprint arXiv:1105.3422* (2011).

[79] P. Miettinen, *On finding joint subspace boolean matrix factorizations.*, in *SDM*, pp. 954–965, SIAM, 2012.

[80] S. Gurukar *et. al.*, *Commit: A scalable approach to mining communication motifs from dynamic networks*, in *SIGMOD*, 2015.

[81] J. Ratkiewicz *et. al.*, *Truthy: mapping the spread of astroturf in microblog streams*, in *WWW Companion*, 2011.

[82] E. Bakshy *et. al.*, *Everyone's an influencer: quantifying influence on twitter*, in *WSDM*, 2011.

[83] J. Leskovec *et. al.*, *Patterns of cascading behavior in large blog graphs*, in *SIAM*, 2007.

[84] K. Saito, R. Nakano, and M. Kimura, *Prediction of information diffusion probabilities for independent cascade model*, in *Knowledge-Based Intelligent Information and Engineering Systems*, 2008.

[85] M. Eftekhar, Y. Ganjali, and N. Koudas, *Information cascade at group scale*, in *KDD*, 2013.

[86] N. Barbieri, F. Bonchi, and G. Manco, *Cascade-based community detection*, in *WSDM*, 2013.

[87] R. Fagin, *Combining fuzzy information from multiple systems*, in *PODS*, 1996.

[88] M. Drosou and E. Pitoura, *Disc diversity: result diversification based on dissimilarity and coverage*, *PVLDB* **6** (2012), no. 1 13–24.

[89] G. Capannini, F. M. Nardini, R. Perego, and F. Silvestri, *Efficient diversification of web search results*, *PVLDB* **4** (2011).

[90] A. Angel and N. Koudas, *Efficient diversity-aware search*, in *SIGMOD Conference*, pp. 781–792, 2011.

[91] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong, *Diversifying search results*, in *WSDM*, 2009.

[92] C. Yu, L. Lakshmanan, and S. A. Yahia, *It takes variety to make a world: diversification in recommender systems*, in *EDBT*, 2009.

[93] E. Vee, U. Srivastava, J. Shanmugasundaram, P. Bhat, and S. Amer-Yahia, *Efficient computation of diverse query results*, in *ICDE*, 2008.

[94] R. Li and J. X. Yu, *Scalable diversified ranking on graphs*, in *ICDM*, 2011.

[95] H. Tong, J. He, Z. Wen, R. Konuru, and C.-Y. Lin, *Diversified ranking on large graphs: an optimization viewpoint*, in *SIGKDD*, 2011.

[96] S. Ranu and A. K. Singh, *Mining statistically significant molecular substructures for efficient molecular classification*, *J. Chem. Inf. Model.* **49** (2009) 2537–2550.

[97] H. Cheng, D. Lo, Y. Zhou, X. Wang, and X. Yan, *Identifying bug signatures using discriminative graph mining*, in *Symposium on software testing and analysis*, 2009.

[98] K. Macropol and A. Singh, *Content-based modeling and prediction of information dissemination*, in *ASONAM*, 2011.

[99] X. Zhu, A. Goldberg, J. Van Gael, and D. Andrzejewski, *Improving diversity in ranking using absorbing random walks*, *HLT-NAACL* (2007) 97–104.

[100] M. Kanehisa and S. Goto, *Kegg: Kyoto encyclopedia of genes and genomes*, *Nucleic Acids Research* **28** (2000), no. 1 27–30.

[101] "SNAP, http://snap.stanford.edu/."

[102] J. Bourgain, *On lipschitz embedding of finite metric spaces in hilbert space*, *Israel Journal of Mathematics* **52** (1985) 46–52.

[103] H. He and A. K. Singh, *Closure-tree: An index structure for graph queries*, in *ICDE*, 2006.

[104] Z. Zeng, A. K. H. Tung, J. Wang, J. Feng, and L. Zhou, *Comparing stars: On approximating graph edit distance*, *PVLDB* **2** (2009), no. 1.

[105] DUD, "http://dud.docking.org/r2/."

[106] C. G. Ballard, *Advances in the treatment of alzheimer's disease: benefits of dual cholinesterase inhibition.*, *Eur Neurol* **47** (2002), no. 1 64–70.

[107] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, *An analysis of approximations for maximizing submodular set functions-i*, *Mathematical Programming* **14** (1978), no. 1 265–294.

[108] U. Feige, *A threshold of ln n for approximating set cover*, *J. ACM* **45** (July, 1998) 634–652.

[109] P. Zezula, G. Amato, V. Dohnal, and M. Batko, *Similarity Search - The Metric Space Approach.* Springer-Verlag, 2006.

[110] Q. Mei, J. Guo, and D. Radev, *Divrank: the interplay of prestige and diversity in information networks*, in *SIGKDD*, 2010.

[111] M. Hua, J. Pei, A. W. C. Fu, X. Lin, and H. fung Leung, *Efficiently answering top-k typicality queries on large databases*, in *VLDB*, pp. 890–901, 2007.

[112] M. Hua, J. Pei, A. Fu, X. Lin, and H.-F. Leung, *Top-k typicality queries and efficient query answering methods on large databases*, *The VLDB Journal* **18** (2009), no. 3 809–835.

[113] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang, *Selecting stars: The k most representative skyline operator*, in *ICDE*, pp. 86–95, 2007.

[114] Y. Zhu, L. Qin, J. X. Yu, and H. Cheng, *Finding top-k similar graphs in graph databases*, in *EDBT*, pp. 456–467, 2012.

[115] X. Yan, P. S. Yu, and J. Han, *Substructure similarity search in graph databases*, in *SIGMOD*, 2005.

[116] L. Zou, L. Chen, and M. T. Özsu, *Distance-join: Pattern match query in a large graph database*, *PVLDB* **2** (2009), no. 1 886–897.

[117] P. Bogdanov, M. Mongiovi, and A. K. Singh, *Mining heavy subgraphs in time-evolving networks*, in *ICDM*, 2011.

[118] D. Lee, O.-R. Jeong, and S.-g. Lee, *Opinion mining of customer feedback data on the web*, in *ICUIMC*, 2008.

[119] J. Dutkowski and T. Ideker, *Protein networks as logic functions in development and cancer*, *PLoS Comput Biol* **7** (09, 2011).

[120] S. Ranu, B. T. Calhoun, A. K. Singh, and S. J. Swamidass, *Probabilistic substructure mining from small-molecule screens*, *Molecular Informatics* **30** (2011) 809–815.

[121] M. Berlingerio, F. Bonchi, B. Bringmann, and A. Gionis, *Mining graph evolution rules*, in *ECML PKDD*, 2009.

[122] C. Robardet, *Constraint-based pattern mining in dynamic graphs*, in *ICDM*, pp. 950–955, 2009.

[123] X. Yan, H. Cheng, J. Han, and P. S. Yu, *Mining Significant Graph Patterns by Scalable Leap Search*, in *SIGMOD*, 2008.

[124] S. Ranu and A. K. Singh, *Graphsig: A scalable approach to mining significant subgraphs in large graph databases*, in *ICDE*, 2009.

[125] N. Jin, C. Young, and W. W. 0010, *Gaia: graph classification using evolutionary computation*, in *SIGMOD*, 2010.

[126] M. A. Hasan and M. J. Zaki, *Output space sampling for graph patterns*, *PVLDB* **2** (2009), no. 1.

[127] K. K. Kumar, B. Rajagopalan, and M. A. Cane, *On the weakening relationship between the indian monsoon and enso*, *Science* **284** (1999), no. 5423 2156–2159.

[128] L. Hyafil and R. L. Rivest, *Constructing optimal binary decision trees is np-complete*, *Information Processing Letters* **5** (1976), no. 1 15 – 17.

[129] B. A. Berg, *Introduction to markov chain monte carlo simulations and their statistical analysis*, *NATIONAL UNIVERSITY OF SINGAPORE* **7** (2005).

[130] S. A. Chowdhury, R. K. Nibbe, M. R. Chance, and M. Koyutürk, *Subnetwork state functions define dysregulated subnetworks in cancer*, *Journal of Computational Biology* **18** (2011), no. 3 263–281.

[131] Y.-W. Chang and C.-J. Lin, *Feature ranking using linear svm*, *Journal of Machine Learning Research* **3** (2008) 53–64.

[132] R. A. Fisher, *The design of experiments / by Sir Ronald A. Fisher.* Oliver & Boyd, Edinburgh :, 7th ed. ed., 1960.

[133] C. Li and H. Li, *Network-constrained regularization and variable selection for analysis of genomic data*, *Bioinformatics* **24** (2008), no. 9 1175–1182.

[134] J. Noirel, G. Sanguinetti, and P. C. Wright, *Identifying differentially expressed subnetworks with mmg*, *Bioinformatics* **24** (2008), no. 23 2792–2793, [http://bioinformatics.oxfordjournals.org/content/24/23/2792.full.pdf+html].

[135] Y. Ye, Y. Zheng, Y. Chen, J. Feng, and X. Xie, *Mining individual life pattern based on location history*, 2009.

[136] J. Zheng and L. M. Ni, *An unsupervised framework for sensing individual and cluster behavior patterns from human mobile data*, in *UbiComp*, 2012.

[137] J. Shang *et. al.*, *Inferring gas consumption and pollution emission of vehicles throughout a city*, in *KDD*, 2014.

[138] Y. Wang, Y. Zheng, and Y. Xue, *Travel time estimation of a path using sparse trajectories*, in *KDD*, 2014.

[139] N. J. Yuan, Y. Zheng, and X. Xie, *Segmentation of urban areas using road networks*, tech. rep., MSR-TR-2012-65, 2012.

[140] G. Karypis and V. Kumar, *Multilevel k-way partitioning scheme for irregular graphs*, *Journal of Parallel and Distributed computing* **48** (1998), no. 1 96–129.

[141] H. Rue and L. Held, *Gaussian Markov random fields: theory and applications*. CRC Press, 2005.

[142] M. Blangiardo and M. Cameletti, *Spatial and Spatio-temporal Bayesian Models with R-INLA*. John Wiley & Sons, 2015.

[143] Y. Li, Y. Zheng, H. Zhang, and L. Chen, *Traffic prediction in a bike-sharing system*, in *SIGSPATIAL*, 2015.

[144] R. J. Hyndman *et. al.*, *Automatic time series for forecasting: the forecast package for R*, tech. rep., Monash University, Department of Econometrics and Business Statistics, 2007.

[145] Y. Kamarianakis and P. Prastacos, *Spatial time series modeling: A review of the proposed methodologies*, *The Regional Economics Applications Laboratory* (2003).

[146] Z. Fan *et. al.*, *CityMomentum: an online approach for crowd behavior prediction at a citywide level*, in *UbiComp*, 2015.

[147] X. Song *et. al.*, *Modeling and probabilistic reasoning of population evacuation during large-scale disaster*, in *KDD*, 2013.

[148] X. Song *et. al.*, *Prediction of human emergency behavior and their mobility following large-scale disaster*, in *KDD*, 2014.

[149] A. Abadi *et. al.*, *Traffic flow prediction for road transportation networks with limited traffic data*, *IEEE Transactions on ITS* **16** (2015), no. 2.

[150] Y. Kamarianakis and P. Prastacos, *Space–time modeling of traffic flow*, *Computers & Geosciences* **31** (2005), no. 2.

[151] Y. Kamarianakis, W. Shen, and L. Wynter, *Real-time road traffic forecasting using regime-switching space-time models and adaptive lasso*, *Applied Stochastic Models in Business and Industry* **28** (2012), no. 4.

[152] R. Silva, S. M. Kang, and E. M. Airoldi, *Predicting traffic volumes and estimating the effects of shocks in massive transportation systems*, *PNAS* **112** (2015), no. 18.

[153] S. Sun, C. Zhang, and G. Yu, *A bayesian network approach to traffic flow forecasting*, *IEEE Transactions on ITS* **7** (2006), no. 1.

[154] Y. Xu *et. al.*, *Accurate and interpretable bayesian mars for traffic flow prediction*, *IEEE Transactions on ITS* **15** (2014), no. 6.

[155] P.-T. Chen, F. Chen, and Z. Qian, *Road traffic congestion monitoring in social media with hinge-loss markov random fields*, in *ICDM*, 2014.

[156] E. J. Horvitz *et. al.*, *Prediction, expectation, and surprise: Methods, designs, and study of a deployed traffic forecasting service*, *arXiv preprint arXiv:1207.1352* (2012).

[157] Y. Zheng *et. al.*, *Urban computing: concepts, methodologies, and applications*, *TIST* **5** (2014), no. 3.

[158] W. Liu *et. al.*, *Discovering spatio-temporal causal interactions in traffic data streams*, in *KDD*, 2011.

[159] Y. Zheng, Y. Liu, J. Yuan, and X. Xie, *Urban computing with taxicabs*, in *UbiComp*, 2011.

[160] J. Yuan *et. al.*, *Discovering regions of different functions in a city using human mobility and POIs*, in *KDD*, 2012.

[161] P. Yin, P. Luo, M. Wang, and W.-C. Lee, *A straw shows which way the wind blows: ranking potentially popular items from early votes*, in *WSDM*, 2012.

[162] T. Zaman, E. B. Fox, E. T. Bradlow, *et. al.*, *A bayesian approach for predicting the popularity of tweets*, *The Annals of Applied Statistics* **8** (2014), no. 3 1583–1611.

[163] K. Lerman and T. Hogg, *Using a model of social dynamics to predict popularity of news*, in *WWW*, 2010.

[164] L. Yu *et. al.*, *From micro to macro: Uncovering and predicting information cascading process with behavioral dynamics*, in *ICDM*, 2015.

[165] G. Szabo and B. A. Huberman, *Predicting the popularity of online content*, *Communications of the ACM* **53** (2010).

[166] L. Hong, O. Dan, and B. D. Davison, *Predicting popular messages in twitter*, in *WWW Companion*, 2011.

[167] G. Gürsun, M. Crovella, and I. Matta, *Describing and forecasting video access patterns*, in *INFOCOM*, 2011.

[168] S. Kong, F. Ye, and L. Feng, *Predicting future retweet counts in a microblog*, *Journal of Computational Information Systems* **10** (2014), no. 4 1393–1404.

[169] J. Cheng, L. Adamic, P. A. Dow, J. M. Kleinberg, and J. Leskovec, *Can cascades be predicted?*, in *WWW*, 2014.

[170] "Hierarchical group-level popularity prediction for online contents." `http://cs.ucsb.edu/~mhoang/gpop_longversion.pdf`.

[171] B. W. Bader, T. G. Kolda, *et. al.*, "Matlab tensor toolbox version 2.6." `http://www.sandia.gov/~tgkolda/TensorToolbox/`, 2015.

[172] D. M. Dunlavy, T. G. Kolda, and E. Acar, *Poblano v1.0: A matlab toolbox for gradient-based optimization*, Tech. Rep. SAND2010-1422, Sandia National Laboratories, 2010.

[173] G. Karypis and V. Kumar, *Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices*, .

[174] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup, *Scalable tensor factorizations for incomplete data*, *Chemometrics and Intelligent Laboratory Systems* **106** (2011), no. 1 41–56.

[175] R. J. Hyndman, R. A. Ahmed, G. Athanasopoulos, and H. L. Shang, *Optimal combination forecasts for hierarchical time series*, *Computational Statistics & Data Analysis* **55** (2011), no. 9 2579–2589.

[176] M. Tsagkias *et. al.*, *Predicting the volume of comments on online news stories*, in *CIKM*, 2009.

[177] R. Bandari, S. Asur, and B. A. Huberman, *The pulse of news in social media: Forecasting popularity*, *arXiv preprint arXiv:1202.0332* (2012).

[178] A. Kaltenbrunner, V. Gomez, and V. Lopez, *Description and prediction of slashdot activity*, in *Web Conference, 2007. LA-WEB 2007. Latin American*, pp. 57–66, IEEE, 2007.

[179] A. Tatar, P. Antoniadis, M. D. De Amorim, and S. Fdida, *Ranking news articles based on popularity prediction*, in *ASONAM*, 2012.

[180] A. Tatar *et. al.*, *From popularity prediction to ranking online news*, *Social Network Analysis and Mining* **4** (2014), no. 1 1–12.

[181] A. Tatar *et. al.*, *Predicting the popularity of online articles based on user comments*, in *International Conference on Web Intelligence, Mining and Semantics*, 2011.

[182] S.-D. Kim, S.-H. Kim, and H.-G. Cho, *Predicting the virtual temperature of web-blog articles as a measurement tool for online popularity*, in *International Conference on Computer and Information Technology (CIT)*, 2011.

[183] A. Tatar *et. al.*, *A survey on predicting the popularity of web content*, *Journal of Internet Services and Applications* **5** (2014), no. 1 1–20.

[184] S. Jamali and H. Rangwala, *Digging digg: Comment mining, popularity prediction, and social network analysis*, in *International Conference on Web Information Systems and Mining (WISM)*, 2009.

[185] S. D. Roy, T. Mei, W. Zeng, and S. Li, *Towards cross-domain learning for social video popularity prediction*, *IEEE Transactions on multimedia* **15** (2013), no. 6 1255–1267.

[186] A. Oghina *et. al.*, *Predicting IMDB movie ratings using social media*, in *European Conference on Information Retrieval*, pp. 503–507, Springer, 2012.

[187] Y. Zhang *et. al.*, *Controlling propagation at group scale on networks*, in *ICDM*, 2015.

[188] Z. Hu, J. Yao, B. Cui, and E. Xing, *Community level diffusion extraction*, in *SIGMOD*, 2015.

[189] Y. Matsubara, Y. Sakurai, W. G. Van Panhuis, and C. Faloutsos, *Funnel: automatic mining of spatially coevolving epidemics*, in *KDD*, 2014.

[190] T.-K. Huang, B. Ribeiro, H. V. Madhyastha, and M. Faloutsos, *The socio-monetary incentives of online social network malware campaigns*, in *Proc. COSN*, pp. 259–270, ACM, 2014.

[191] S. A. Myers, C. Zhu, and J. Leskovec, *Information diffusion and external influence in networks*, in *Proc. SIGKDD*, 2012.

[192] J. Leskovec, L. Backstrom, and J. Kleinberg, *Meme-tracking and the dynamics of the news cycle*, in *Proc. SIGKDD*, 2009.

[193] Z. Ma, A. Sun, and G. Cong, *On predicting the popularity of newly emerging hashtags in twitter*, *JASIST* **64** (2013).

[194] Y. Matsubara, Y. Sakurai, B. A. Prakash, L. Li, and C. Faloutsos, *Rise and fall patterns of information diffusion: model and implications*, in *Proc. SIGKDD*, pp. 6–14, 2012.

[195] E. Keogh, S. Lonardi, and C. Ratanamahatana, *Towards parameter-free data mining*, *Proc. SIGKDD* (2004) 206–215.

[196] Z. Xing, J. Pei, and E. Keogh, *A brief survey on sequence classification*, *ACM SIGKDD Explorations Newsletter* **12** (Nov., 2010) 40.

[197] M. Jenders, G. Kasneci, and F. Naumann, *Analyzing and predicting viral tweets*, in *Proc. WWW*, 2013.

[198] C. T. Perretti, S. B. Munch, and G. Sugihara, *Model-free forecasting outperforms the correct mechanistic model for simulated and experimental data.*, *PNAS* **110** (Mar., 2013) 5253–7.

[199] N. H. Kuiper, *Tests concerning random points on a circle*. Proceedings of the Koninklijke Nederlandse Akademie, 1960.

[200] R. A. Fisher, *The design of experiments*. Oliver & Boyd, 1935.

[201] A. N. Kolmogorov, *Foundations of the Theory of Probability.* Chelsea Publishing Co., 1950.

[202] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, *A kernel two-sample test*, *JMLR* **13** (Mar., 2012) 723–773.

[203] M. A. Stephens, *Use of the Kolmogorov-Smirnov, Cramer–Von-Mises and related statistics without extensive tables*, *J. of the Royal Statistical Society Series B* **32** (1970).

[204] R. Sinkhorn and P. Knopp, *Concerning nonnegative matrices and doubly stochastic matrices.*, *Pacific Journal of Mathematics* **21** (1967), no. 2 343–348.

[205] R. Zass and A. Shashua, *Probabilistic graph and hypergraph matching*, in *CVPR*, pp. 1–8, IEEE, 2008.

[206] J. R. Wilson, *The inspection paradox in renewal-reward processes*, *Operations Research Letters* **2** (Apr., 1983) 27–30.

[207] F. Murai, B. Ribeiro, D. Towsley, and P. Wang, *On Set Size Distribution Estimation and the Characterization of Large Networks via Sampling*, *IEEE Journal on Selected Areas in Communications* **31** (2013), no. 6 1017–1025.

[208] S. I. Resnick, *Extreme values, regular variation, and point processes*. Springer Science & Business Media, 2007.

[209] F. Murai, B. Ribeiro, D. Towsley, and K. Gile, *Characterizing Branching Processes from Sampled Data*, in *Proc. WWW Companion*, pp. 805–811, 2013. arXiv:1302.5847.

[210] H. Kwak, C. Lee, H. Park, and S. Moon, *What is Twitter, a social network or a news media?*, in *Proc. WWW*, 2010.

[211] J. Yang and J. Leskovec, *Modeling information diffusion in implicit networks*, in *ICDM*, 2010.

[212] M. Goetz, J. Leskovec, M. McGlohon, and C. Faloutsos, *Modeling blog dynamics.*, in *ICWSM*, 2009.

[213] A. Goyal, F. Bonchi, and L. V. Lakshmanan, *Learning influence probabilities in social networks*, in *Proc. WSDM*, 2010.

[214] D. Ramage, S. T. Dumais, and D. J. Liebling, *Characterizing microblogs with topic models.*, *ICWSM* **10** (2010).

[215] W. Galuba, K. Aberer, and D. Chakraborty, *Outtweeting the twitterers-predicting information cascades in microblogs*, in *Proc. WOSN*, 2010.

[216] F. Wang, H. Wang, and K. Xu, *Diffusive Logistic Model Towards Predicting Information Diffusion in Online Social Networks*, *WINE* **cs.SI** (Aug., 2011) [arXiv:1108.0442].

[217] D. M. Romero, C. Tan, and J. Ugander, *On the interplay between social and topical structure.*, in *ICWSM*, 2013.

[218] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan, *Group formation in large social networks: membership, growth, and evolution*, in *Proc. SIGKDD*, 2006.

[219] A. Kupavskii, L. Ostroumova, A. Umnov, S. Usachev, P. Serdyukov, G. Gusev, and A. Kustarev, *Prediction of retweet cascade size over time*, in *CIKM*, 2012.

[220] O. Tsur and A. Rappoport, *What's in a hashtag?: content based prediction of the spread of ideas in microblogging communities*, in *Proc. WSDM*, 2012.

[221] E. Bakshy, B. Karrer, and L. A. Adamic, *Social influence and the diffusion of user-created content*, in *EC*, 2009.

[222] A. Najar, L. Denoyer, and P. Gallinari, *Predicting information diffusion on social networks with partial knowledge*, in *Proc. WWW Companion*, pp. 1197–1204, ACM, 2012.

[223] A. Clauset, C. R. Shalizi, and M. E. Newman, *Power-law distributions in empirical data*, *SIAM review* **51** (2009).

[224] L. A. Adamic, *Zipf, power-laws, and pareto-a ranking tutorial*, *Xerox Palo Alto Research Center* (2000).