SUBJECTIVELY INTERESTING PATTERNS IN NETWORKS

Ph.D. THESIS

by

SARANG KAPOOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE – 247667 (INDIA)
JUNE, 2021

SUBJECTIVELY INTERESTING PATTERNS IN NETWORKS

A THESIS

Submitted in partial fulfilment of the requirements for the award of the degree

of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE AND ENGINEERING

by

SARANG KAPOOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE – 247667 (INDIA)
JUNE, 2021

INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

STUDENT'S DECLARATION

I hereby certify that the work presented in the thesis entitled **"SUBJECTIVELY INTERESTING PATTERNS IN NETWORKS"** is my own work carried out during a period from August, 2015 to February, 2021 under the supervision of Dr. Dhish Kumar Saxena, Associate Professor, Department of Mechanical and Industrial Engineering, Indian Institute of Technology Roorkee, Roorkee and Dr. Matthijs van Leeuwen, Associate Professor, LIACS, Leiden University, The Netherlands.

The matter presented in the thesis has not been submitted for the award of any other degree of this or any other Institute.

Dated: June 14, 2021 **(SARANG KAPOOR)**

SUPERVISOR'S DECLARATION

This is to certify that the above mentioned work is carried out under my supervision.

Dated: June 14, 2021 **(Dhish Kumar Saxena)** **(Matthijs van Leeuwen)**

Abstract

Graph mining is one of the well-established fields of data mining. Still, it remains an active research area, considering that in many real-world scenarios (e.g., social media, airline operations), a large amount of data comes in the form of graphs. Graphs are helpful when the structure of data is equally important as the content, and necessarily, graphs are used to model relationships between entities. Graph mining methods focus on discovering subsets of vertices in a graph, as patterns representing useful structural insights about the data.

The usefulness of a pattern for an analyst is mainly quantified using an interestingness measure. In general, most of the interestingness measures are data-driven, i.e., the interestingness is determined only based on the available data and the pre-specified patterns' structures to be discovered, with minimal analyst's interaction. However, in practice, the concept of interestingness is mostly subjective, as the usefulness of a pattern is dependent on the analyst and his/her prior knowledge. Here, a pattern is considered to be subjectively interesting if it departs from the analyst's expectations. Notably, the existing notion of subjective interestingness is largely limited for simple graphs (graphs having a single edge between a vertex-pair). This marks a major research gap, since, there do exist many real-world scenarios which could be realistically modelled through multigraphs, characterized by multiple edges between a vertex-pair. For example, in the co-authorship network, two authors may have multiple co-authored publications, a scenario that could be modelled through multiple edges (each representing a unique co-authored publication) between the two vertices (representing each author).

One of the fundamental contributions of this thesis relates to bridging the research gap, through the proposition of a subjective interestingness measure for multigraph patterns. Subsequently, an algorithm to iteratively discover subjectively interesting multigraph patterns is developed. The proposed algorithm's advantages over existing methods are demonstrated by extensive experiments on synthetic and real-world datasets, such as co-actor and co-authorship networks.

Arguably, the relationships in the data also evolve with time. For instance, in a collaboration network of actors, the collaboration between two actors may or may not

be observed each year. This fact makes the corresponding graph representation dynamic. The problem of summarization, i.e., succinctly describing a dynamic graph, is widely studied to enable a user to store and visualize large volumes of data efficiently. The focus of existing methods has been on finding a minimal set of temporal graph patterns. However, in many applications, the evolution of a graph structure is equally important for an analyst as a graph pattern's temporal presence. For example, an analyst may be interested in learning informative changes, such as how the traffic load changes in real-time between different airports.

Recognizing the above, this thesis also fundamentally contributes by way of laying out and defining the fundamental concepts of subjective interestingness for dynamic graphs. Here, a dynamic graph is treated as a sequence of static graph snapshots, and each snapshot could be treated both as a simple graph or a multigraph. Further, the problem of online summarization of dynamic graphs is introduced, where the summary (of informative changes) is incrementally conveyed to the analyst, as the graph evolves while accounting for his/her knowledge at the specific time instances. Here, an algorithm for online summarization of dynamic graphs has been proposed, enabling revelation of interpretable changes to the connectivity structure of the graph with time. The efficacy of the proposed approach and the algorithm is demonstrated by experimental evaluations over a large variety of real-world datasets, such as dynamic interaction network, co-authorship network, the network of links on the web, and a dedicated treatment of a real-world airline case study.

Acknowledgements

Completing a Ph.D. thesis is not an individual experience; instead, it takes place in a collective context. Several people stimulated, encouraged, motivated, and guided me through the process towards the end goal. Whom all I would like to express my sincere gratitude.

Words might not be enough in expressing my indebtedness towards my thesis supervisors Prof. Dhish K. Saxena and Prof. Matthijs van Leeuwen. I am grateful to Prof. Saxena for giving me an opportunity to pursue my goals and believing in me both before and during the journey in these glorious years of my life. He always made sure to put me in situations where I not only learned academically but also improved myself personally. He had been caring towards me but never expressed in front of me, only to be sure that I never lose direction. His methods have been quite tricky, but then Prof. Van Leeuwen was always there to comfort me and always motivated me. Although I was miles away from him, he always had time for me. Numerous discussions and several weekly Skype meetings with him made me overcome every obstacle I faced during this course. He always went beyond expectations, even when his newborn baby needed all the attention; he dedicated even his personal time to advise me on my thesis. I am blessed to have both of them in my life. Anything more I say about them is customarily trivial.

I would also like to thank my friends and colleagues Avadh Kishor, Tarun Kulshrestha, Prateek Singh, Ajay Chauhan, Shivam Rai, Divyam Aggarwal, Kunal Bansal, and Sukrit Mittal. They have been very supportive during the ups & downs and were always there to provide me the personal help whenever required. A special mention to Avadh Kishor and Tarun Kulshrestha for always encouraging me, understanding me, helping me overcome challenging & difficult times, and stimulating discussions. I also acknowledge the support of Hugo Proença; he is a true friend whose criticism always helped me overcome my limitations; he also understood me and the difficulties in my journey quite aptly. I am also grateful to the support of my other friends including Paras Mishra, Rahul Sharma, Parminder Grewal, and Arpit Malhotra.

The best outcome of the past few years is finding a best friend for life, my soul-mate Deepanksha Arora. She was the only person who appreciated me, unconditionally stood

by my side, and invested all her emotions in me. She has been non-judgmental of me and instrumental in instilling confidence. She had faith in me and my intellect even when I felt like digging the hole and crawling into one because I often didn't have faith in myself. The journey has not been a comfortable ride, both academically and personally, as we were in a long-distance relationship and occupied with pursuing career goals in these past several years. I genuinely thank Deepanksha for sticking by my side, even when I was irritable and depressed. I feel that we both learned a lot about life, strengthening our commitment & determination to each other.

Finally, I especially thank my parents and my siblings. They have sacrificed their lives for me and provided unconditional love and care. I love them so much, and I would not have made it this far without them. My father gave me all the strength to stand still and taught me to make the most of the worst of times. I am grateful to him for all his teachings, he taught me the real meaning of a very popular quote.

> *"He is a wise man who does not grieve for the things which he has not, but rejoices for those which he has."*
>
> -Epictetus

Last but not the least, I gratefully acknowledge the sponsors MeitY (India), NWO (The Netherlands), and GE Aviation (Bangalore, India) for funding the Indo-Dutch joint research project titled: *"A Systems Approach towards Data Mining and Prediction in Airlines Operations"* (SAPPAO). This thesis is sponsored as a part of this project.

Sarang Kapoor

Dedicated

to

My Buzurgs,
Parents, Teachers,
Siblings and My Fiancée

Contents

List of Figures

List of Tables

List of Abbreviations

MDL	Minimum Description Length
MaxEnt	Maximum Entropy
SSG	Subjective Subgraph Patterns
SIMP	Subjectively Interesting Multigraph Patterns
DSSG	Dynamic Subjective Subgraph Patterns
DSIMP	Dynamic and Subjectively Interesting Multigraph Patterns
CR	Compression Ratio
SBT	Scheduled Block Time
NAS	National Aviation System
BTS	Bureau of Transportation Statistics
AD	Aggregate Deviation
SI	Self Information
IC	Information Content
IG	Information Gain
FI	Feature Importance
CNM	**C**lauset, A., **N**ewman, M. E., and **M**oore, C.
TC	TimeCrunch
SDGM	Scalable Dynamic Graph Mining method

List of Symbols

G	Graph data
G^S	Simple graph
G^M	Multigraph
V	Set of vertices in a graph
$u,\ v$	vertex in a graph
E	Set of edges in a graph
P, Q, $\mathbf{Pr}$	Probability Distribution
p	probability of success
ρ	Density of a graph—undirected: $\rho = \frac{2*\|E\|}{\|V\|\cdot(\|V\|-1)}$, directed: $\rho = \frac{\|E\|}{\|V\|\cdot(\|V\|-1)}$
η	Average vertex degree of a graph—undirected: $\eta = \frac{2*\|E\|}{\|V\|}$, directed: $\rho = \frac{\|E\|}{\|V\|}$
d	Diameter of a graph
G_T	Dynamic graph
G_T^S	Evolving simple graph
G_T^M	Evolving multigraph
$\mathbf{A}$	Adjacency matrix of a graph
$\mathbf{D_T}$	Dynamic data
$\mathcal{B}$	Set of prior beliefs
$\mathbf{C}$	Set of discovered knowledge in form of a set of constraints
$\mathcal{SI}$	Self-information
$\mathcal{AD}$	Aggregate Deviation
$\mathcal{IC}$	Information Content
$\mathcal{DL}$	Description Length
SI, $\mathbf{I}$	Subjective Interestingness
$\mathcal{IG}$	Information gain
α	Atomic change or action
q	Parameter (probability) used to encode set of vertices
$\mathcal{E}$	Expectation matrix

$\mathcal{G}$	Gulf matrix
P_X	Maximum entropy distribution w.r.t. set of constraints X
$\mathcal{S}$	Number of states/snapshots in a dynamic data
H, θ	Subgraph pattern
KL	Kullback–Leibler divergence
k_W	Number of edges in a vertex-induced subgraph with W vertices
n_W	Maximum number of edges in a vertex-induced subgraph with W vertices
L	Codelength
$\mathrm{L}_{\mathbb{N}}$	Universal Codelength to encode an integer

Chapter 1

Introduction

"Data is the new oil. It's valuable, but if unrefined it cannot really be used. It has to be changed into gas, plastic, chemicals, etc. to create a valuable entity that drives profitable activity; so must data be broken down, analyzed for it to have value." This phrase dating back to 2006 is credited to the UK mathematician Clive Humby. Over the last decade, this phrase has grown all the more significant, as nearly everything impacting our day to day life is linked to a data source: internet, call data records, healthcare records, customer transactions, stock market data, news, literature, scientific publications, weather data, etc. Hence, the need to analyze data sets to generate value is now being considered critically important. Notably, the early attempts to analyze data sets and summarize their main characteristics link to Exploratory Data Analysis (EDA) [Tukey, 1977]. It focuses on gaining insights into the systematic relations between variables characterizing the data, often using visual and statistical methods. However, with an ever-increasing volume, velocity and variety of data, the field of Knowledge Discovery from Data (KDD) [Piatetsky-Shapiro, 1990] has gained prominence. KDD, commonly referred to as data mining, is a means to discover interesting patterns and knowledge from data, which otherwise may not be easily visualized. Although data mining methods may adopt EDA techniques, their scope is not limited to them.

Subjective Interestingness Measures

In data mining, *interestingness* measures play an instrumental role in quantifying the potential usefulness of patterns and even ranking them. *Interestingness measures* are used to underline aspects such as *conciseness*, *novelty*, *utility* and *actionability* [Geng and Hamilton, 2006]. Notably, a pattern that considerably improves an analyst's understanding is either novel or may contradict his/her expectations about the data. In the same spirit, De Bie [2011a] suggested that a data mining process is driven by both an analyst and the data itself. An analyst can intervene in the data-mining process at different levels, such as

defining a method for pattern discovery, ranking or filtering. In that, an interestingness measure is considered *subjective* if it incorporates an analyst's knowledge either *a priori* or by intermittent interaction through the data-mining process. The property of an interestingness measure to depart from or contradict an analyst's knowledge is referred to as *surprisingness*. The advantages of using *subjective interestingness measures* in a data mining method relate to the following:

- uncertainty and misconceptions of an analyst about the data can be reduced.
- unique patterns can be found which otherwise may be overlooked.
- patterns discovered may pinpoint the shortcomings of any previous learning in the data.
- the patterns found may point to facets of the data that may need further study.

Although subjective interestingness measures have several advantages over any other interestingness measure, it is challenging to quantify an analyst's prior beliefs. Most of the data mining methods use an approach where a measure is proposed to approximate the actual interestingness of a pattern [van Leeuwen et al., 2016]. Another challenge in data mining methods is that searching for a pattern in data is a computationally expensive task. Hence, most data mining methods use an interestingness measure which: 1) approximate the actual interestingness, and 2) can be computed efficiently. The former is mostly treated informally to develop a cost-effective method and may lack the element of surprisingness.

Graph Mining

Many real-world scenarios where relationships between entities play a vital role are best studied as graphs (or networks). Some examples include social networks, interaction networks, transport networks, protein interaction networks and computer networks. The fact that network analysis can be used to untangle many real-world problems is widely accepted and explored. In network analysis, interesting patterns may render information in the form of a subset of vertices acting as a coherent group that is intently connected and may share common properties, interests, and theme compared to any randomly selected subset.

In order to find interesting patterns which are closely related or densely connected, a graph mining method often uses a form of density measures, such as average degree [Charikar, 2000; Khuller and Saha, 2009], k-cores [Batagelj and Zaversnik, 2003], cliques [Palla et al., 2005], quasi-cliques [Tsourakakis et al., 2013; Uno, 2010], or k-plex [McClosky and Hicks, 2012]. These measures have fixed beliefs on the appearance of a closely connected subset of vertices or subgraphs, and are designed to optimize the computational cost of finding such subgraphs. As discussed earlier, these measures also neglect a critical aspect of interestingness: usefulness to an analyst. For example, it is more surprising and useful to know in a network that vertices with low vertex degree forms a dense subgraph pattern than vertices with high vertex degree. Thus, interestingness is subjective and quantified by considering an analyst's prior belief about the network. Informally, subjectively interesting subgraph patterns are defined as patterns that are surprisingly dense relative to an analyst's expectations.

One of the major contributions towards formalizing subjective interestingness, is a general framework introduced by De Bie [2011b]. The author suggests that the formulation of subjective interestingness is guided by the information contained in a pattern and the complexity required to encode this information. Based on this, van Leeuwen et al. [2016] proposed a method to discover subjective subgraph patterns, while Bendimerad et al. [2020] addressed the problem of finding subjectively interesting attributed subgraphs.

Arguably, the relationships in the graph may also evolve with time. For instance, in a collaboration network of actors, the collaboration between two actors may or may not be observed each year. This fact makes the corresponding graph representation dynamic. In dynamic graph mining, the problem of summarization, i.e., succinctly describing a *dynamic graph*, is widely studied to enable a user to store and visualize large volumes of data efficiently. The focus of existing methods has been on finding a minimal set of temporal graph patterns [Shah et al., 2015; Sun et al., 2007; Tsalouchidou et al., 2020]. In that, a common practice is to treat a dynamic graph as a sequence of static graph snapshots, upon fragmentation of time into consecutive internals of particular duration. The summary is created by stitching together structures found in different snapshots. The structures discovered by TimeCrunch [Shah et al., 2015] include cliques, stars, cores,

bipartite cores, and their near representatives. These structures are presumed to exist in a graph snapshot which may or may not be informative for an analyst.

Aim and Objectives

The brief introduction above has pointed to the practical utility of the notion of subjective interestingness. It has also established with reference to the state of the art that this concept has only been realized in the context of static simple graphs and attributed graphs. This, in turn, points to the following research gap:

- *the concept of subjective interestingness has not been developed for multigraphs*, implying graphs where multiple edges may characterize any vertex pair. This is alarming, given that many real-world scenarios could be realistically modelled only through *multigraphs*. For instance, an airline network (where multiple flights/edges are natural between a pair of airports/vertices) or co-authorship network (where two authors/vertices may co-author multiple publications/edges).
- *the concept of subjective interestingness has not been developed for dynamic graphs*. In such a scenario, the incremental evolution of subjectively interesting graph structures with time (online summarization), which could be of great interest for analysts, can not be studied. For instance, an interested analyst may be deprived of learning as to how the air-traffic between different airports changes incrementally in real-time, in a manner that departs from his/her prior knowledge. This points to the need to develop the concept of subjective interestingness for *evolving simple graphs and multigraphs* (dynamic graphs treated as a sequence of static–simple graphs and multigraphs, respectively).

In pursuit of bridging the above research gap, this thesis *aims to develop the concept of subjective interestingness for different graph types, including, static multigraphs, evolving simple graphs, and evolving multigraphs*. This aim has been realized through the following objectives:

- subjective interestingness has been *defined* for static multigraphs.
- an algorithm to iteratively discover subjectively interesting (static) multigraph patterns has been proposed.

- the definitions of subjective interestingness for static simple graphs and multigraphs have been *adapted* to cater to their dynamic counterparts.
- a novel algorithm for online summarization offering subjectively interesting incremental changes has been proposed, for both evolving simple graphs and multigraphs.
- the efficacy of the proposed definitions and algorithms has been demonstrated through experimental evaluations on several synthetic and real-world datasets, including a dedicated treatment of a real-world airline case study.

Key Challenges

Some of the key challenges faced in this thesis, and the remedies adopted are highlighted below.

- A subjective interestingness measure should be computationally cost-effective. The formulation of subjective interestingness measure in [De Bie, 2011b] computes self-information of a pattern as the negative logarithm of the probability of a pattern. This factor can be computed in a (somewhat) cost-effective manner if a pattern is known, but this factor turns out to be cost-ineffective when used in a search process to discover a pattern. In case of static simple graphs, van Leeuwen et al. [2016] approximated the self-information of a pattern using the upper bounds of the upper tail probability distributions which can be computed in much less cost. However, for addressing static multigraphs, heuristic based novel and computationally cost-effective subjective interestingness measures have been proposed in this thesis.
- In the process of discovering subjectively interesting graph patterns, a crucial factor is the number of patterns that are sufficient for an analyst to learn. A large set of patterns may overwhelm an analyst. This issue is not of major concern in static graphs and can be considered as a user-defined parameter. However, in the case of evolving graphs, as the number of subjectively interesting patterns may vary across different time snapshots, fixing a user-defined parameter a priori may not be effective, as it may lead to under-fitting or over-fitting of the model or the knowledge discovered. To overcome this challenge, this thesis draws inspiration from the Minimum Description Length (MDL) principle [Grünwald, 2007] and proposes

a novel subjective interestingness measure that implicitly helps avoid either under-fitting or over-fitting.

- Further, in the case of evolving graphs, another challenge relates to how the changes in the graph patterns should be discovered, evaluated and communicated to an analyst? To address the discovery challenge, six different types of atomic changes or actions have been defined. These atomic changes are evaluated using the proposed MDL inspired subjective interestingness measure. In each iteration, the most promising atomic change is communicated to the analyst.

Contributions and Organization of the Thesis

In this thesis, we develop graph mining methods to discover subgraph patterns in several types of graphs, including static multigraphs, evolving simple graphs and evolving multigraphs. The main contributions of this thesis are summarized below vis-à-vis the organization of this thesis.

Chapter 2 provides the literature survey of the related work. In this, we highlight the research gap in the literature and show how this thesis's contributions are distinguished from previous (related) work.

Chapter 3 presents the first contribution of the thesis. This chapter discusses the conceptual framework proposed for subjective interestingness measures for multigraph patterns. We lay the foundations by modeling the different type of prior beliefs. We also propose a greedy algorithm, termed SIMP, for iteratively discovering subjectively interesting multigraph patterns. Finally, we demonstrate the proposed algorithm's efficacy through extensive experiments on synthetic and real-world examples, such as co-authorship and co-actor networks.

Chapter 4 presents the second contribution of the thesis. In this chapter, we introduce a problem of subjective online summarization of evolving simple graphs. We approached this problem through a proposed novel generic framework for subjective interestingness for sequential data, considering that an evolving graph is treated as a sequence of static graph snapshots. This framework is then instantiated for evolving *simple* graphs. Finally, we present DSSG, a heuristic algorithm to practically ap-

proach the proposed problem. We demonstrate the efficacy of our approach through experimental evaluations on real-world data.

Chapter 5 presents the third contribution of the thesis. This chapter discusses a different form of graph called evolving multigraphs, where a dynamic graph is treated as a sequence of static multigraph snapshots. Subsequently, we instantiate the novel framework proposed in Chapter 4, for evolving multigraphs by laying out the definitions of necessary concepts. We demonstrate the efficacy of the proposed approach through experiments on real-world examples which can be realistically modeled as evolving multigraphs.

Chapter 6 presents the fourth contribution of the thesis. In this chapter, the proposed methods and algorithms have been applied to real-world airline case studies to establish practical applicability. First, we present a case study where we showcase a practical application of subjectively interesting multigraph patterns in studying airline networks to learn the set of connected airports with surprisingly high traffic volumes. Next, we propose the means to translate the discovered multigraph patterns into machine-interpretable features. Later, we endeavor to address two major problems of airliners, i.e., SBT selection and delay prediction. We highlight the benefits that may be associated with a subjectively interesting pattern in these domains. Finally, using DSSG, we present a dedicated treatment of a real-world airline case study. In this case study, we exhibit a scenario where an online and incremental analysis of structural changes in dynamic graphs can render valuable insights.

Chapter 7 concludes the thesis and discusses the potential directions of future work.

Chapter **2**

Literature Review

This chapter presents a literature survey of the two domains most closely related to this thesis research problem. These domains are subjective interestingness measures and graph mining methods. In Section 2.1, we discuss the studies in the literature focused on the definition and formulation of subjective interestingness measures. In Section 2.2, we discuss the methods of graph mining present in the literature. We highlight the two research gaps identified in the literature, i.e., the concept of subjective interestingness for multigraphs (in Section 2.2.1) and for dynamic graphs (in Section 2.2.2). Finally, we conclude in Section 2.3.

2.1 Subjective Interestingness

Knowledge discovery from data (KDD), commonly known as data mining, is the process of extracting insights from the data in the form of patterns. These extracted patterns are evaluated for the information contained based on interestingness measures [Han et al., 2011]. Data mining aims to find patterns that can be added to the analyst's knowledge (or belief) about the data. Thus, interestingness measures are instrumental for ranking patterns based on their inherent knowledge or potential significance to the analyst. These measures also help the analyst balance the time-space trade-off in costs associated with a data mining method. Geng and Hamilton [2006] proposed that an interestingness measure may have the following properties, including *conciseness, coverage, reliability, peculiarity, diversity, novelty, surprisingness, utility*, and *actionability*.

A pattern is *concise* if an analyst can easily understand it; while *coverage* describes the comprehensiveness of a pattern or a set of patterns, i.e., it shall cover a large part of the data [Padmanabhan and Tuzhilin, 2000]. If a pattern has high conviction about the data, it is *reliable* to an analyst. Reliability is commonly evaluated in association rule mining methods in the form of measures such as support and confidence [Tan et al., 2002]. *Peculiarity* and *diversity* implies that a pattern shall be considerably disparate from the

other found patterns [Hilderman and Hamilton, 2013; Zhong et al., 2003]. *Novelty* and *surprisingness* propound the characteristic feature of a pattern, if an analyst does not know it earlier [Sahar, 1999] and it controverts an analyst's prior beliefs [Liu et al., 1999; Silberschatz and Tuzhilin, 1996], respectively. Finally, with *utility* and *actionability*, a pattern can be categorized if it is usable and can offer decision making support to an analyst for a required problem [Chan et al., 2003; Ling et al., 2002; Wang et al., 2002; Yao and Hamilton, 2006].

Among the above discussed nine properties, *novelty* and *surprisingness* are based on an analyst's current expectations and are thus not limited to the raw data itself. Hence, a measure is considered to be *subjective* if it incorporates the analyst's knowledge either explicitly or by interaction with the analyst through the mining process [Geng and Hamilton, 2006]. The critical differences between the objective measures and the subjective measures can be understood from Table 2.1.

Table 2.1. Comparison of objective measures versus subjective measures.

Objective Measures	**Subjective Measures**
• Domain **independent** with minimal interaction with the analyst.	• Domain **dependent** with the analyst's prior knowledge built in.
• Only data driven and **do not access** the prior knowledge of the analyst.	• **Access** the analyst's prior knowledge along with the raw data.
• Criteria on which these measures are based: ⋆ conciseness, ⋆ coverage, ⋆ reliability, ⋆ peculiarity, and ⋆ diversity.	• Main criteria on which these measures are based: ⋆ novelty & surprisingness, and ⋆ usefulness & actionability. These measures may also address the criteria for objective measures.

The central idea of *subjective interestingness* was first proposed by Silberschatz and Tuzhilin [1996], who argued that a useful pattern for one analyst might not be useful for another. In other words, a pattern's interestingness is not only limited to the data and the structure of a pattern, but also on the analyst who investigates it. The authors classified

an interestingness measure based on two criteria: *surprisingness* and *actionability*. Since actionability is challenging to capture, the authors believed that "*most surprising patterns are actionable and most actionable patterns are surprising*". Also, a pattern is considered surprising if it contradicts an analyst's expectations, modelled as a belief system, where beliefs are defined as logical statements.

Although the notion of subjective interestingness was proposed a few decades ago, the challenge of modelling prior beliefs has been persistent. This challenge has been attempted to overcome by defining prior beliefs: in the form of rules $X \rightarrow Y$ (X and Y being conjunctions of literals) [Padmanabhan and Tuzhilin, 2000; Silberschatz and Tuzhilin, 1996], or using a Bayesian network model [Jaroszewicz and Simovici, 2004]. These methods have their limitations; they are either specific for patterns in the form of rules or dependent on the type of data. Also, using a Bayesian network, it is not straightforward to implement broad classes of prior beliefs.

It is only a decade ago when, to overcome the above limitations, De Bie [2011b] proposed a framework for FORmalizing Subjective Interestingness in Exploratory Data mining (FORSIED)—based on the Maximum Entropy (MaxEnt) principle [Jaynes, 1982]. The FORSIED framework can be used to model the broad classes of prior belief as probability distributions, referred to as background distribution, for general types of data. The author also proposed that given a background distribution, the subjective interestingness measure can be defined using information-theoretic principles. The defined measure of subjective interestingness comprises the following quantities: self-information of a pattern, and the complexity required to encode a pattern. Self-information is the negative log-probability of a pattern [Cover and Thomas, 1991] which quantifies the aspects of *novelty* and *surprisingness* of a pattern. While the complexity of a pattern is described in terms of description length—i.e., the bits required to encode a pattern and addresses the aspect of *conciseness*. To cater to the criteria of *coverage*, *peculiarity*, and *diversity*, the patterns can be iteratively discovered by updating background distribution at each step [De Bie, 2011a]. The idea behind updating the background distribution is to align with the analyst's current knowledge (prior knowledge and the knowledge discovered). Thus, if a pattern is already learned then the probability will be substantially reduced, and hence,

a *diverse* set of patterns with high *coverage* can be discovered.

Based on the FORSIED framework, the method for subjective interestingness for subgraph patterns was proposed by van Leeuwen et al. [2016]. Lijffijt et al. [2016] presented a generic framework for interesting structured relational patterns. Some of the other methods proposed in the literature based on the FORSIED framework include subjectively interesting alternative clustering [Kontonasios and De Bie, 2015]; subjectively interesting connecting trees [Adriaens et al., 2017]; subjectively interesting motifs in time series [Deng et al., 2019]; and mining subjectively interesting attributed subgraphs [Bendimerad et al., 2020]. Our main contributions based on the FORSIED framework include subjectively interesting multigraph patterns [Kapoor et al., 2020] and online subjective summarization of dynamic graphs [Kapoor et al., 2021]. In this thesis, we will discuss these contributions in detail.

2.2 Graph Mining Methods

Graph mining is a prominent field of research in data mining. In many key domains, where relationships between entities are important, graph data is found—to name a few: bioinformatics [Bouvel et al., 2005; Paul and Anand, 2018], chemistry [Estrada et al., 2003], social networks [Dakiche et al., 2019], transportation & traffic movement [Marshall et al., 2018], computer network [Jain et al., 2011], text summarization [Chatterjee et al., 2018], and geographical data [Duflot et al., 2018]. The graphs are also studied in different forms, such as, simple graphs [van Leeuwen et al., 2016], weighted graphs [Andersen and Chellapilla, 2009], multigraphs graphs [Ingalalli et al., 2018], multilayer graphs [Papalexakis et al., 2013], attributed graphs [Bendimerad et al., 2020], directed graphs [Fang et al., 2018], graph database [Angles and Gutierrez, 2008], and dynamic graphs [Shah et al., 2015]. Example applications that can be solved using graph mining methods include gene selection using protein interaction graph [Dutta et al., 2019], identifying distinctive patterns in Drug-drug interaction network [Sahu and Anand, 2018], routing in a wireless sensor network [Gupta and Jana, 2015], finding relevant communities in a social network [Sharma et al., 2009], and so on.

This section discusses the widely known graph mining methods in the literature, di-

vided into static and dynamic graph mining methods.

2.2.1 Static Graph Methods

A graph is considered to be static if it is fixed and does not observe any changes with time. Although this thesis's contribution is most closely related to dense subgraph mining methods, we discuss other static graph mining methods in the literature for completeness.

Dense Subgraph Mining in Static Graphs

Dense subgraph mining is a well-researched problem, where density is generally considered as a measure of importance. A dense subgraph may indicate a group or set of vertices having common themes, common interests or high degree of interactions. The terms cliques, quasi-cliques [Abello et al., 2002; Matsuda et al., 1999], k-cores [Seidman, 1983], k-plex [Seidman and Foster, 1978], kD-cliques [Luce, 1950] and k-club [Mokken, 1979] in static graphs have been systematically defined as interestingness measures and are explored to represent dense subgraphs. In the past, significant effort has been on finding dense patterns based on average degree [Charikar, 2000; Khuller and Saha, 2009], k-cores [Batagelj and Zaversnik, 2003], cliques [Palla et al., 2005], quasi-cliques [Tsourakakis et al., 2013; Uno, 2010], or k-plex [McClosky and Hicks, 2012]. Recent work on identifying quasi-cliques includes Tsourakakis et al. [2013]; Veremyev et al. [2016], while Wu and Hao [2015] summarize all methods for solving the maximum clique problem. For weighted graphs, the notion of average degree has been extended in [Andersen and Chellapilla, 2009]. It has been noted that the modularity measure [Clauset et al., 2004; Newman, 2006], originally proposed for unweighted simple graphs, can be trivially extended to weighted simple graphs. Gibson et al. [2005] studied the graphs which are massive to identify large dense bipartite subgraphs.

Multilayer graphs are also widely studied for finding subgraph patterns or clusters in the data; dense pattern discovery [Dong et al., 2012; Papalexakis et al., 2013]; and community detection [Qi et al., 2012; Ruan et al., 2013; Silva et al., 2012; Xu et al., 2012; Zhou et al., 2009], by use of matrix factorization, cluster expansion, pattern mining, etc. For the discovery of dense subgraphs in multilayer graphs, Galimberti et al. [2017] proposed a multilayer core decomposition method. A multilayer can be treated as multiple

graphs necessarily having the same set of vertices. Thus, the problem of finding densest common subgraph has been studied in [Charikar et al., 2018; Jethava and Beerenwinkel, 2015]. Multigraphs, on the other hand, are limitedly studied in the literature for finding frequent patterns [Ingalalli et al., 2018], vertex colouring [Győri and Palmer, 2009] and edge-colouring [Goldberg, 1984]; with little or no emphasis on dense subgraph mining from a multigraph data. In the literature, the two terms, multigraph and multilayer graphs are often used in place of each other. However, it can be argued that a multilayer graph has several layers, each acting as an independent simple, weighted or attributed graph. In contrast, a multigraph does not necessarily contain independent layers. However, in some case, a multigraph can be informally represented as a weighted graph, such that weights are non-negative integers.

Notably, the interestingness of a pattern is often defined as the departure from the expectations. In the case when expectations are objectively defined (say, through modularity [Clauset et al., 2004; Newman, 2006] or edge surplus [Tsourakakis et al., 2013]), it is termed objectively interesting; and if expectations are derived subjectively (say, from the prior beliefs of an analyst), it is termed subjectively interesting. van Leeuwen et al. [2016] defined subjectively interesting patterns for simple graphs and introduced a heuristic algorithm for mining those. Here, though the expectations were computed using the prior beliefs, the background distribution was assumed to be the product of independent Bernoulli distributions, given which the generalization of this work to the other type of graph settings is a non-trivial and challenging task.

Thus, identifying the first research gap, one of the thesis's contributions is to propose a subjective interestingness based method to discover static multigraph patterns.

Clustering/Partitioning in Static Graphs

Another popular sub-category of static graph mining is clustering or partitioning of the graph. Most of the methods focus on discovering splits, cuts, or partitions in a graph to identify different regions or communities of interest using spectral partitioning [Alpert et al., 1999], min-max cut [Ding et al., 2001], minimum cut trees [Flake et al., 2004], betweenness measures [Newman and Girvan, 2004], quasi-cliques [Abello et al., 2002]

or modularity maximization [Newman, 2006]. Alternatively, for clustering, more than one conflicting measures can also be used to identify more generalized partitions in the data [Saha and Bandyopadhyay, 2013]. In clustering methods, often nodes are clustered to identify dense regions based on edge behavior. The edges in these methods may have weights or numerical labels associated. Thus, these methods can also be used for weighted and multilayer graphs, apart from simple graphs. In multilayer graphs, several methods are proposed in the literature for community detection using node clustering approach [Qi et al., 2012; Ruan et al., 2013; Silva et al., 2012; Xu et al., 2012; Zhou et al., 2009].

Clustering methods cover the graph as a whole, while pattern (subgraph) mining in graph data restricts the knowledge discovery to some areas of interest, giving more flexibility. One of the other limitations of clustering methods is that the number of partitions needs to be defined by the user *a priori*. Also, in many real-world scenarios, it is imperative to find overlapping communities. This highlights another shortcoming of these methods; to overcome a greedy clique expansion method is proposed in [Paul and Anand, 2018].

Static Graph Summarization

The idea of static graph summarization is to compress a graph [Koutra et al., 2014; Navlakha et al., 2008] or aggregate nodes/edges in a graph [Goebl et al., 2016; LeFevre and Terzi, 2010; Toivonen et al., 2011]. It is found to improve query efficiency [LeFevre and Terzi, 2010], speed up clustering algorithms [Toivonen et al., 2011], effectively compress a graph dataset [Navlakha et al., 2008], and provide better visualization [Koutra et al., 2014] of a graph dataset. Koutra et al. [2014] describe a graph by identifying structures using a predefined vocabulary of graph structures such as stars, full & near cliques, full & near bipartite cores, and chains, which minimizes the total encoded length of the graph along with the model (based on the minimum description length principle). Another popular objective of static graph summarization is to find influential dynamics in a network through patterns [Goebl et al., 2016]. These patterns provide a high-level description of a graph and are considered relevant and informative in real datasets such as social networks, where information propagation is an essential characteristic of the data.

Cook and Holder [1994] *subjectively* summarize a graph by providing a hierarchical description of structural regularities guided by the background knowledge in terms of rules, including compactness, connectivity, coverage and other types of domain-dependent rules. Similar to our proposed approach, the authors combine the concept of minimum description length with background knowledge. However, we model background knowledge using constraints and the maximum entropy principle.

Other Static Graph Mining Methods

The other categories of static graph mining methods apart from the above includes frequent subgraph mining methods [Bringmann and Nijssen, 2008; Elseidy et al., 2014; Meng and Tu, 2017] and network motifs discovery methods [Grochow and Kellis, 2007; Wernicke, 2006; Xia et al., 2019]. Frequent patterns are defined as structures in a graph, which are frequent with respect to a support measure. In comparison, network motifs are defined as small substructures in a graph significantly present in the input graph compared to their presence in any random graph.

2.2.2 Dynamic Graph Methods

Dynamic graphs are informally defined as graphs that evolve with time. In a dynamic graph, different settings can be assumed, where edges or vertices or both appear/disappear with time (topological changes), attributes of edges or vertices or both changes with time (label changes), and both attributes and labels can be observed to change together with time. This thesis considers a setting where only edges appear or disappear with time, and vertices remain fixed. Also, the edges and vertices are considered to be unlabelled.

Dynamic Subgraph Mining

This category covers methods that identify temporal graph patterns in a dynamic network. Due to the time factor in dynamic graphs, static graph methods cannot be used, unless redefined [Fournier-Viger et al., 2020]. In dynamic graph mining literature, one of the well-renowned problems is frequent subgraph mining [Borgwardt et al., 2006; Wackersreuther et al., 2010] similar to static graphs. Apart from frequent patterns, periodic patterns are also of interest in dynamic graphs Apostolico et al. [2011]; Halder et al. [2017].

Rozenshtein et al. [2017] studied interaction networks to find dense and temporally compact patterns. The authors introduced the k-Densest episode identification problem on temporal graphs [Rozenshtein et al., 2018], where an episode is defined as a pair of a time interval and a subgraph. Galimberti et al. [2018] proposed the idea of maximal span-cores and span-cores decomposition of temporal networks.

There are other dynamic graph mining methods present in the literature which can be considered in this section; for simplicity, these methods are discussed in the following section.

Dynamic Graph Summarization

This category is different from dynamic graph mining: graph summarization methods identify structures and evolution that provide a *succinct description* of a network, while graph mining methods identify *all possible patterns* in the network. As one of the contributions of this thesis fits this category, Table 2.2 shows an overview of both existing methods and ours; we will elaborate on this comparison in the last paragraph of this section.

GraphScope [Sun et al., 2007] was one of the first methods that focused on summarizing temporal graphs. It partitions the graph into bipartite cores and cliques. Simultaneously, by detecting the change in the graph segment's encoding cost upon presentation of a new graph with the state's evolution, segments are identified.

Com^2 [Araujo et al., 2014] identifies temporal edge-labelled communities in a graph and uses the minimum description length (MDL) principle with Canonical Polyadic (CP) or PARAFAC decomposition. TimeCrunch [Shah et al., 2015] also uses the MDL principle to summarize a temporal graph. The authors identify graph structures, using the vocabulary of graph structures given by Koutra et al. [2014], along with their corresponding temporal presence in terms of one-shot, periodic, flickering ranged. Adhikari et al. [2017] summarize a dynamic network by aggregating nodes into supernodes and time pairs into 'super time'. This method creates a flattened graph (static) after aggregation. Each of these methods concerns an instance of MDL-based dynamic graph compression (either lossy or lossless), but none of them directly summarize how a dynamic graph change and

evolves.

Various methods in the literature have directly or indirectly addressed the problem of summarizing the evolution of a dynamic graph. You et al. [2009] captures repeated addition and removal of subgraphs between two consecutive graph snapshots in a dynamic graph. Scharwächter et al. [2016] proposed to find frequent structural changes, such as triadic closure and homophilic rewiring, in the form of evolution rules. Ahmed and Karypis [2015] summarize graph evolution by capturing co-evolving relational motifs, which occur when all or a majority of the occurrences of a relational pattern—or motif—evolve similarly over time. Robardet [2009] proposed to capture the evolution of isolated pseudo-cliques over time employing a sequence of five temporal events, including formation, dissolution, growth, diminution and stability.

Similarly, Ahmed and Karypis [2012] proposed to epitomize an evolving graph by identifying Evolving Induced Relational States (EIRS). The authors defined EIRS as a sequence of Induced Relational States (IRS), which are a set of vertices that remain connected by similar edges having the same direction and label for several consecutive snapshots (based on a threshold). In EIRS, each IRS's time interval cannot overlap with other IRS and has several or at least a certain number of common vertices. Lin et al. [2011] focus on discovering evolving communities by analyzing the dynamic interactions between vertices by representing the multi-dimensional and multi-relational characteristics as a relational hypergraph called a 'metagraph'. Another recent method based on TimeCrunch [Shah et al., 2015] that aims to capture graph structures' evolution is given in the preliminary work by Saran and Vreeken [2019]. They capture evolving graph patterns by capturing dynamic events such as growth, split, merge, and change in structure type (e.g., from clique to star) of a pattern. Based on their characteristics, these methods can be referred to as methods for discovering evolving graph patterns.

All methods mentioned in this category thus far are defined for a 'fixed' dynamic graph, i.e., over a fixed time interval, and not for a 'streaming' dynamic graph that is generated on-the-fly and should also be analyzed on-the-fly, where the summary should change upon the presentation of a new snapshot of a graph. In other words, these methods do not support *online summarization*. Recent methods for online dynamic graph summarization,

Table 2.2. Comparison of dynamic graph summarization methods. The four rightmost columns show whether a method supports: **OS**–Online Summarization; **IS**–Incremental Summarization; **EP**–Evolving Patterns; **AS**–Automatic selection of summary size.

Paper	Algorithm	Summary type	Selection criterion	Pattern type/structure	OS	IS	EP	AS
Sun et al. [2007]	GraphScope	Temporal graph segments	MDL	Bipartites, cliques	✓	✗	✗	✓
Araujo et al. [2014]	COM^2	Edge labelled communities	(Lossy) MDL with tensor decomposition	Stars, bipartites, tiny groups	✗	✗	✗	✓
Shah et al. [2015]	TimeCrunch	Patterns with temporal presence	MDL	Stars, bipartites, cliques, chains	✗	✗	✗	✗*
Adhikari et al. [2017]	NetCondense	Condensed flattened network	Node and time pair based aggregation measure	Aggregated node and time pairs	✗	✗	✗	✗
You et al. [2009]	—	Graph rewriting rules	Structural changes	Compressed subgraphs	✗	✗	✗	✗
Scharwächter et al. [2016]	EVOMINE	Evolution rules	Embedding based and event base support	Triadic closure, homophilic rewiring	✗	✗	✓	✗
Ahmed and Karypis [2012]	EIRS	Evolution paths of stable relational states	Maximal evolving induced relational state	Induced relational states	✗	✗	✓	✗
Ahmed and Karypis [2015]	CRMminer	Frequent co-evolutions	User defined minimum support	Co-evolving relational motifs	✗	✗	✓	✗
Lin et al. [2011]	MetaFac	Emergent communities	Metagraph factorization	Communities	✓	✗	✓	✗
Saran and Vreeken [2019]	Mango	Evolving patterns with temporal presence	MDL	Stars, bipartites, cliques, chains	✗	✗	✓	✗*
Tang et al. [2016]	TCM	Condensed graph stream	Aggregation of nodes/edges	Graph sketch	✓	✗	✗	—
Khan and Aggarwal [2016]	gMatrix	Condensed graph stream	Hash-mapping of nodes	3D sketch	✓	✗	✗	—
Qu et al. [2016]	OSNet	Set of interesting subgraphs of cascades	proScope & proRadius based interestingness	Spreading tree	✓	✗	✓	✗
Tsalouchidou et al. [2020]	SDGM	Condensed network	k-partitions of nodes	Dense micro-clusters	✓	✗	✗	✗
This Thesis	DSSG	Informative evolving patterns	Maximum entropy and MDL based	Changes in dense subgraphs of any shape	✓	✓	✓	✓

* The size of the summary is dependent on the size of candidate structures generated in each static snapshot of a dynamic graph, which is not necessarily automatic.

discussed next, include Khan and Aggarwal [2016]; Qu et al. [2016]; Tang et al. [2016]; Tsalouchidou et al. [2020].

Tang et al. [2016] and Khan and Aggarwal [2016] generate a graphical sketch of a dynamic graph, aggregating vertices and edge weights, which is updated after each snapshot of a graph sequence. These graphical sketches are useful to improve the efficiency of graph-based queries. Qu et al. [2016] summarize a diffusion network, i.e., a dynamic graph where information propagates with time, by discovering spreading trees (n-ary) as cascades, which grows with a state change. Recently, Tsalouchidou et al. [2020] proposed the Scalable Dynamic Graph summarization Method (SDGM) to generate an online summary by extending the static graph summarization approach of LeFevre and Terzi [2010]. Although these methods provide online summarization, they do not summarize informative state-to-state relative changes in a dynamic graph. That is, they do not provide *incremental summaries*, where each relative change in the structure of the graph is summarized and communicated to the analyst step by step.

To bridge this gap in the literature, we consider the problem of discovering informative changes in a streaming dynamic graph in an incremental manner. As we are interested in finding all informative changes, we require our method to determine the number of returned patterns automatically. To this end, we propose to identify subgraphs that maximally deviate from the analyst's current knowledge. For this, we build on the notion of subjective interestingness proposed by De Bie [2011b]. To the best of our knowledge, we are the first to consider the problem of subjective, incremental, online graph summarization. This is corroborated by the qualitative comparison in Table 2.2, which shows the relevant characteristics for all dynamic graph summarization methods discussed in this section.

Since we propose to summarize a dynamic graph by means of dense patterns, we will adapt TimeCrunch [Shah et al., 2015] and SDGM [Tsalouchidou et al., 2020] to establish two baseline methods for empirical comparison in Chapter 4 (Section 4.5).

2.3 Summary

In this chapter, we discussed the relevant literature and described the research gaps that we address in this thesis. We highlighted the differences and advantages of subjective interestingness measures over objective interestingness measures in data mining. In the graph mining literature, it is observed that the use of subjective interestingness measures is limited and sporadic. Although the problem of dense subgraph mining is widely studied, no robust definition has been given in the past for dense multigraph patterns. We also observed that for summarizing a dynamic graph taking into account an analyst's prior knowledge, informative changes in an evolving graph could be incrementally communicated through evolving subjectively dense subgraphs of any shape. Until now, this problem has not been investigated in the literature. It is of note that a common practice for studying dynamic graphs is to segment the graph into a sequence of static graph snapshots. These snapshots can be realistically represented as both simple and multigraphs. Hence, we observed that another research gap is the absence of subjective interestingness measures and methods to subjectively summarize a dynamic graph. This thesis bridges these two research gaps by defining the subjective interestingness measures for static multigraphs and dynamic graphs, and hence, proposing the corresponding graph mining methods.

Chapter **3**

Subjective Interestingness of Multigraph Patterns

Over the past decade, network analysis has attracted substantial interest because of its potential to solve many real-world problems. This chapter[3.1] lays the conceptual foundation for an application in aviation, through focusing on the discovery of patterns in *multigraphs* (graphs in which multiple edges can be present between vertices). Our main contributions are two-fold. Firstly, we propose a novel *subjective interestingness measure for patterns in both undirected and directed multigraphs*. Though this proposition is inspired by a previous related research for simple graphs (having only single edges), the properties of multigraphs make this transition challenging. Secondly, we propose a greedy algorithm for subjectively interesting pattern mining, and demonstrate its efficacy through several experiments on synthetic and real-world examples.

[3.1] This chapter has been published as [Kapoor et al., 2020]: **Kapoor, S.**, Saxena, D.K. & van Leeuwen, M. Discovering subjectively interesting multigraph patterns. In *Mach Learn*, Springer, **109**, 1669–1696 (2020).

3.1 Introduction

Over the past decade, researchers have realized that network analysis can be used to address many real-world problems. Examples include problems related to computer network infrastructure, co-authorship (scientific or other), co-actors (e.g., in movies), transport (road, airline, ...), and even tax evasion [Aggarwal et al., 2010]. This has led to research on several types of networks, typically modelled as *simple graphs* (graphs having at most one edge between any pair of vertices) and *weighted graphs* (simple graphs but with weights on edges). A type of network that, to the best of our knowledge, has not yet been widely considered in the data mining literature[3.2] is one that needs to be modelled as a *multigraph* (graph in which multiple edges can be present between any pair of vertices). Motivated by an application in aviation, this chapter lays the conceptual foundations for the discovery of subjectively interesting multigraphs patterns (SIMPs). SIMPs are defined as those subgraphs that are unexpected and/or contradict an analyst's prior beliefs or background knowledge [van Leeuwen et al., 2016]. The rationale for the representation of an airline network as a multigraph and targeting of SIMPs vis-à-vis alternative approaches are discussed below.

In an airline network, symbolically depicted in Figure 3.1, there can be several flights (edges in a graph) between a pair of airports (vertices in a graph), which explains as to why this network could be modelled as a multigraph[3.3]. Arguably, an airline network could also be studied as a *multilayer* graph, where multiple sets of edges are defined on the same set of vertices. In that setting, each set of edges acts as a unique layer, and different layers are characterized by different data properties. For instance, between a pair of airports, multiple flights from different airlines might operate, and each airline's flights may constitute a layer, differing from other layers. Notably, multigraphs may constitute building blocks for multilayer graphs (so far investigated only through simple graphs [Papalexakis et al., 2013; Qi et al., 2012]). To avoid the added complexity of

[3.2] Note that the term multigraph was used before Dong et al. [2012]; Papalexakis et al. [2013], but those works employ an alternative definition; see next section for details.

[3.3] At this formative stage, our endeavour is to analyse 'static' multigraphs (for fixed time intervals), though the longer-term goal is to analyse dynamic multigraphs.

multilayer graphs, in this stage the multigraph representation of a network will form the basis for analysis in this chapter.

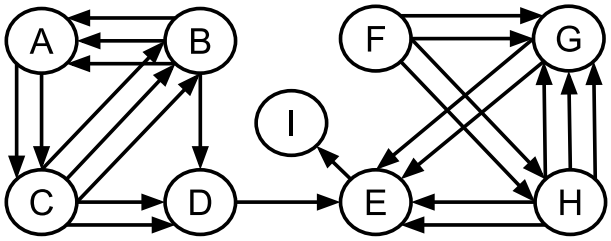


Fig. 3.1. An airline transportation network modelled as directed multigraph

Flight delays have punitive implications for airlines. Intuitively, and based on historical evidence, it is often believed that flight congestion between a pair of airports make them vulnerable to delays. Yet, delays are a reality, hence, it is critically important to mine the network data and facilitate scientifically informed assessment and decision making. Efforts in this direction have been made but they are limited in scope and practical relevance. For instance, finding *objectively* dense patterns (where density is defined through k-cores, cliques, k-plex, maximum average degree, etc.) is a commonly studied problem [Batagelj and Zaversnik, 2003; Charikar, 2000; Khuller and Saha, 2009; McClosky and Hicks, 2012; Palla et al., 2005; Tsourakakis et al., 2013]. However, simple graphs do not suitably model an airline network in the first place. This thesis attempts to overcome this limitation by focusing on multigraphs. Furthermore, it builds on the premise that capturing events (say, in terms of delays) which depart from an analyst's prior beliefs and may be referred as unexpectedly dense, *relative to what the analyst already knows* [van Leeuwen et al., 2016], may be more revealing (say, in terms of source of delay), interesting, and practically useful. This justifies our focus on SIMPs dedicatedly in multigraph settings, besides the fact that this conceptual foundation could be useful in several other applications, including co-authorship analysis.

The structure of this chapter is as follows. Our proposed approach is presented in Section 3.4, where we formalize the conceptual contributions on SIMPs, and present a greedy algorithm for the discovery of SIMPs in Section 3.5. Section 3.6 demonstrates

the efficacy of the proposed algorithm; discusses the properties of the discovered SIMPs; compares our approach to existing methods on synthetic and real-world data. The chapter concludes with key observations and future directions in Section 3.7. Notably, we present a case study in aviation in Chapter 6, highlighting how our approach could help improve an analyst's understanding of the problem.

3.2 Preliminaries

A *multigraph* is denoted by $G^M = (V, E^M)$, where V is a set of n vertices (usually indexed using symbol u or v) and E^M is a multiset of edges, where each edge $e \in E^M$ is an element of $V \times V$. In contrast to the common *simple graph* setting, there can be multiple edges between any pair of vertices. The adjacency matrix for the graph is denoted by $\mathbf{A} \in \mathbb{N}_0^{n \times n}$, with $a_{u,v} \in \mathbb{N}_0$ equal to the number of edges between u and v. For example, $a_{u,v} = 0$ means that there are no edges between u and v. This *undirected* definition can be straightforwardly extended to a *directed multigraph* by letting $a_{u,v}$ represent the number of edges from u to v. For the sake of simplicity, in this thesis we focus the exposition on multigraphs without self-edges, for which it holds that $(u, v) \in E \Rightarrow u \neq v$, but if desired this restriction could simply be dropped.

We build on the premise that an analyst knows (or has direct access to) the list of vertices V in the graph, and is interested in improving self's knowledge and understanding of the edges. Thus, the data to be mined is the edge multiset E^M, and the domain of this data is $\mathbb{N}_0^{n \times n}$ (further constrained by exclusion of self-loops, implying that the diagonal values of $\mathbf{A}$ have to be 0).

The framework by De Bie [2011b] suggests that prior knowledge (modelled as constraints) can be represented as a probability distribution P over the data domain. As the constraints typically leave many of such distributions possible, the *maximum entropy* (MaxEnt) *principle* is leveraged to argue that the distribution having the largest entropy should be used. The framework then quantifies the subjective interestingness of a pattern as the ratio of *information content* to *description length*, where information content is the negative logarithm of the probability of the pattern given the background distribution, and description length is the code length required to communicate the pattern to the user.

In the following, we will build on this framework for multigraph patterns, albeit with a different definition of subjective interestingness.

3.3 Prior Beliefs and Background Distributions

We here consider and model the following three different types of prior beliefs that an analyst may have:

1. **Total number of edges (Belief-c)**. The analyst here is assumed to have a prior belief concerning (only) the total number of edges in the network, e.g., on the total number of flights in case of airline data. This follows:

$$\sum_{\mathbf{A}\in\mathbb{N}_0^{n\times n}} P(\mathbf{A}) \sum_{u,v\in V} a_{u,v} = |E^M|. \tag{3.1}$$

   The MaxEnt distribution with constraint Eq. 3.1 results in a product of independent uniform geometric distributions, one for each random variable $a_{u,v} \in \mathbb{N}_0$ (cf. De Bie [2011b]), where

$$P(\mathbf{A}) = \prod_{u,v\in V} \exp(2\lambda \cdot a_{u,v}) \cdot (1 - \exp(2\lambda)). \tag{3.2}$$

   Here, each distribution represented as $P_{u,v}(a_{u,v})$ has a probability of success equal to $[1 - \exp(2\lambda)]$, where λ is a Lagrangian multiplier corresponding to the constraint in Eq. 3.1.

2. **Number of edges per vertex (Belief-i)**. In this case, the analyst is assumed to have prior beliefs on the row and/or column marginals of the adjacency matrix, denoted by d_u^r and d_v^c respectively. In the airline case, this corresponds to knowing the total number of flights leaving from (d_u^r) or arriving (d_v^c) at each airport. This belief is represented by

$$\sum_{\mathbf{A}\in\mathbb{N}_0^{n\times n}} P(\mathbf{A}) \sum_{v\in V} a_{u,v} = d_u^r, \ (\forall u), \tag{3.3}$$

$$\sum_{\mathbf{A}\in\mathbb{N}_0^{n\times n}} P(\mathbf{A}) \sum_{u\in V} a_{u,v} = d_v^c. \ (\forall v). \tag{3.4}$$

We observe that the MaxEnt distribution with constraints in Eq. 3.3-3.4 results in a product of independent geometric distributions given by

$$P(\mathbf{A}) = \prod_{u,v \in V} \exp((\lambda_u^r + \lambda_v^c) \cdot a_{u,v}) \cdot (1 - \exp((\lambda_u^r + \lambda_v^c)), \tag{3.5}$$

for each random variable $a_{u,v} \in \mathbb{N}_0$. This corresponds to the 'geometric' case in [De Bie, 2011b], where each distribution $P_{u,v}(a_{u,v})$ has a probability of success equal to $[1 - \exp(\lambda_u^r + \lambda_v^c)]$. Here, λ_u^r and λ_v^c are Lagrangian multipliers following the constraints in Eq. 3.3-3.4.

3. **Number of neighbors per vertex (Belief-m)**. In the third and final case, the analyst is assumed to have a prior belief about the number of unique neighbors of each vertex, referred to as m_u. In an airline case, this could be considered as the total number of unique routes on which an airline operates from any airport. This prior belief is represented as

$$\sum_{\mathbf{A} \in \mathbb{N}_0^{n \times n}} P(\mathbf{A}) \sum_{v \in V} 1_{a_{u,v}} = m_u^r, \ (\forall u), \tag{3.6}$$

$$\sum_{\mathbf{A} \in \mathbb{N}_0^{n \times n}} P(\mathbf{A}) \sum_{u \in V} 1_{a_{u,v}} = m_v^c, \ (\forall v), \tag{3.7}$$

where $1_{a_{u,v}}$ is the indicator function, which equals 1 if $a_{u,v}$ is a non-zero value and 0 otherwise. This case is a multigraph-specific belief, as in case of a simple graph d_u would be equal to m_u, intuitive of the fact that at most one edge can exist between any two vertices. Hence, we will use this belief to complement the previous two types of belief. In this chapter, we consider the case where this type of belief is combined with Belief-i. The MaxEnt distribution $P(\mathbf{A})$ for the data with constraints in Eq. 3.3-3.4 and Eq. 3.6-3.7 reduces to a product of independent probability distributions $P(\mathbf{A}) = \prod_{u,v \in V} P_{u,v}(a_{u,v})$ for each random variable $a_{u,v} \in \mathbb{N}_0$, where

$$P_{u,v}(a_{u,v}) = \frac{[1 - R_{u,v}]}{[1 - R_{u,v}\,(1 - S_{u,v})]} \cdot R_{u,v}^{a_{u,v}} \cdot S_{u,v}^{1_{a_{u,v} \neq 0}}, \tag{3.8}$$

$$\textit{where,} \quad R_{u,v} = \exp\left(\lambda_u^r + \lambda_v^c\right) \textit{ and } S_{u,v} = \exp\left(\mu_u^r + \mu_v^c\right).$$

Here λ_u^r, λ_v^c, μ_u^r and μ_v^c are Lagrangian multipliers corresponding to the constraints in Eq. 3.3, 3.4, 3.6 and 3.7, respectively. For completeness, a proof of the MaxEnt distribution $P(\mathbf{A})$ for this case is relegated to Appendix A.1.

The above-mentioned constraints are described for directed multigraphs represented by $\mathbf{A}$, however for undirected multigraphs $u < v$ should be added as an additional constraint. In this chapter, the above three types of prior beliefs or knowledge will be evaluated. However, other types of prior beliefs could also be considered, for example, details about different airline carrier's flights arriving or departing from an airport. Though it is beyond the scope of this chapter, such cases would also lead to a product of independent probability distributions, which can be used to compute the expected number of edges between any vertex pair.

3.4 Proposed Approach

Given the prior beliefs of the analyst, the background distribution of the data can be derived as the MaxEnt distribution [De Bie, 2011b]. We now establish a subjective interestingness measure for multigraph patterns given the background distribution and the data.

As multigraphs do not have a strict limit on the maximum number of edges that can occur between any pair of vertices, existing work on simple graphs by van Leeuwen et al. [2016] cannot be directly extended to multigraphs. We, therefore, introduce a new definition of interestingness based on the *expectation matrix* $\mathcal{E}$. In this matrix, of size $|V| \times |V|$, each entry $\mathcal{E}_{u,v}$ is defined as the number of expected edges—based on the prior beliefs—between vertices u and v.

The expectation of any geometric distribution of the form $(1-p)^x \cdot p$ for random variable $x \in \mathbb{N}_0$, where p is the probability of success, is given as $E(x) = \frac{1-p}{p}$. The probability distributions for Belief-c and Belief-i are represented in the natural form of a geometric distribution. Thus, we have expectation $\mathcal{E}_{u,v} = \frac{\exp(2\lambda)}{1-\exp(2\lambda)} = \rho$ and $\mathcal{E}_{u,v} = \frac{\exp(\lambda_u^r+\lambda_v^c)}{1-\exp(\lambda_u^r+\lambda_v^c)}$ for Belief-c and Belief-i, respectively. Here, ρ is the density[3.4] of a graph.

[3.4]For undirected graphs $\rho = \frac{2*|E|}{|V|\cdot(|V|-1)}$, for directed graphs $\rho = \frac{|E|}{|V|\cdot(|V|-1)}$

The probability distribution for Belief-m, however, cannot be represented in the natural form of a geometric distribution. Hence, the expected number of edges between vertices u and v is computed as

$$\mathcal{E}_{u,v} = \frac{\exp(\lambda_u^r + \lambda_v^c) \cdot \exp(\mu_u^r + \mu_v^c)}{[1 - \exp(\lambda_u^r + \lambda_v^c)]\,[1 - \exp(\lambda_u^r + \lambda_v^c)\,(1 - \exp(\mu_u^r + \mu_v^c))]}. \tag{3.9}$$

Next, we quantify the interestingness of a vertex-induced subgraph pattern by *the difference between the actual and the expected number of edges*. For this, we derive what we call the *gulf matrix* $\mathcal{G}$, which is computed as the difference between the adjacency matrix and expectation matrix, i.e., $\mathcal{G} = \mathbf{A} - \mathcal{E}$. A value $\mathcal{G}_{u,v}$ is positive if the expected number of edges between u and v is lower than the actual number of edges, and negative in the opposite case. Without loss of generality, we assume that only positive differences are of interest; one could reverse the signs to discover 'sparse subgraphs'.

For a given pattern, we sum the deviations over all vertex-pairs it contains, and define this sum as the aggregate deviation of the pattern, as follows.

Definition 3.1 (Aggregate Deviation) *Given multigraph $G^M = (V, E^M)$ and gulf matrix $\mathcal{G}$, the* aggregate deviation $\mathcal{AD}$ *of a subgraph $H = (W, E')$, where $W \subseteq V$ and $E' \subseteq E^M$, is given by $\mathcal{AD}(H, \mathcal{G}) = \sum_{u,v \in W} \mathcal{G}_{u,v}$.*

One might be inclined to mine subgraphs that maximize $\mathcal{AD}$, but in practice, this is likely to lead to large subgraphs. This is problematic because large subgraphs may not be interesting for and/or comprehensible to the analyst. Similar to existing subjective interestingness approaches [De Bie, 2011b; Lijffijt et al., 2016; van Leeuwen et al., 2016], we, therefore, penalize a pattern's deviation with its description length, i.e., its 'complexity'.

Definition 3.2 (Description Length) *Given multigraph $G^M = (V, E^M)$, subgraph $H = (W, E')$, and parameter q, the cost required to describe a subgraph to the analyst—in terms of its vertices—is given by* description length $\mathcal{DL}$*, defined as*

$$\begin{aligned}\mathcal{DL}(H) &= -\sum_{u \in W} \log(q) - \sum_{u \notin W, u \in V} \log(1 - q) \\ &= |W| \cdot \log\left(\frac{1-q}{q}\right) + |V| \cdot \log\left(\frac{1}{1-q}\right),\end{aligned}$$

where $-\log(q)$ *is the cost of a vertex included in* W *and* $-\log(1-q)$ *is the cost of a vertex excluded from* W*.*

Definition 3.2 uses Shannon-optimal codes to describe the pattern, using a vertex probability, i.e., parameter q, that is set by the analyst in advance. The smaller the analyst believes the size of an interesting pattern to be, the smaller the q and the smaller the exclusion cost of a vertex, and the other way around. Once q is fixed then the description length increases with the size of the pattern as for each added vertex in a pattern a cost equal to $\log((1-q)/q)$ is added to the description length. Thus, q can be interpreted as the expected probability that a vertex is included in a random pattern and is set by the analyst based on expected/desired pattern size. Description length can be used to penalize larger patterns, for which it is easier to have a large $\mathcal{AD}$.

Ideally, a pattern is considered to be interesting if it is highly informative (quantified in terms of aggregate deviation, $\mathcal{AD}$) and can be encoded with a short code (measured in terms of description length, $\mathcal{DL}$). Thus, we next define subjective interestingness of a pattern as the ratio of its aggregate deviation to its description length.

Definition 3.3 (Subjective Interestingness) *Given multigraph* $G^M = (V, E^M)$*, subgraph* H*, and gulf matrix* $\mathcal{G}$*, the* subjective interestingness $\mathbf{I}$ *of* H *is given by* $\mathbf{I}(H, \mathcal{G}) = \frac{\mathcal{AD}(H,\mathcal{G})}{\mathcal{DL}(H)}$.

Note that in the previous definitions of a subgraph pattern we considered *any* vertex-induced subgraph, but this includes subgraphs that consist of multiple components, i.e., subgraphs that are not connected. As an analyst will expect patterns to be connected, we add the constraint that each subgraph has to be connected[3.5]. This leads to the following problem for finding the subjectively most interesting multigraph pattern.

Problem 3.1 (Subjectively Interesting Multigraph Pattern (SIMP)) *Let* $G^M = (V, E^M)$ *be a multigraph and* $\mathcal{G}$ *a gulf matrix. Find a set of vertices* $W \subset V$ *and its corresponding vertex-induced subgraph* H *that maximizes* $\mathbf{I}(H, \mathcal{G})$ *such that* H *is a (weakly) connected component.*

[3.5]For directed multigraphs the constraint is relaxed to weakly connected component, i.e., the undirected equivalent of the directed graph is a connected graph

3.4.1 Updating the Background Distribution

When a new pattern is found, it is presented to the analyst, which then transforms the knowledge of the analyst, who learns from the information contained in the pattern. Hence, these newly learned information should be reflected in the background distribution. More specifically, in the updated background distribution $P'(\mathbf{A})$ the expectation of the number of edges in the pattern should be equal to the actual number of edges found. The rationale behind this is that, by updating the background distribution in this manner, the aggregate deviation of the pattern becomes (almost) zero and hence the pattern is no longer interesting.

Let $H = (W, E')$ be the communicated pattern, then the updated MaxEnt distribution is calculated using the following convex optimization problem, which is the *I-projection* of the preceding background distribution onto the set of distributions that are consistent with the communicated pattern [De Bie, 2011a]. Thus, the problem is formulated as

$$P'(\mathbf{A}) = \underset{Q \in \mathbf{Q}}{\operatorname{argmin}} \sum_{\mathbf{A}} Q(\mathbf{A}) \log\left(\frac{Q(\mathbf{A})}{P(\mathbf{A})}\right) \tag{3.10}$$

$$s.t. \sum_{\mathbf{A} \in \mathbb{N}_0^{n \times n}} Q(\mathbf{A}) \sum_{u,v \in W} a_{u,v} = |E'|, \tag{3.11}$$

$$\sum_{\mathbf{A} \in \mathbb{N}_0^{n \times n}} Q(\mathbf{A}) = 1, \tag{3.12}$$

where $\mathbf{Q}$ represents set of all possible distributions over $\mathbf{A}$ and the constraint in Equation 3.11 represents the acquired belief of the analyst on the data. That is, the vertex-induced subgraph H, with the set of vertices W, contains $|E'|$ edges. Using this updating procedure, we can perform an iterative exploratory data mining process: we can mine the subjectively most interesting multigraph pattern from the data, update the background distribution, and repeatedly perform these two steps to mine multiple SIMPs.

Theorem 3.1 *Let $P(\boldsymbol{A})$ be a product of independent probability distributions over data $\boldsymbol{A} \in \mathbb{N}_0^{V \times V}$, then the optimal solution to the problem defined by Equations 3.10-3.12 is also a product of an independent probability distributions $P'(\boldsymbol{A})$, such that:*

1. *if $P(\boldsymbol{A}) = \prod_{u,v \in V} (1 - p_{u,v})^{a_{u,v}} \cdot p_{u,v}$ then $P'(\boldsymbol{A}) = \prod_{u,v \in V} (1 - p'_{u,v})^{a_{u,v}} \cdot p'_{u,v}$*

2. *if* $P(\boldsymbol{A}) = \prod_{u,v \in V} \frac{1-R_{u,v}}{1-R_{u,v}(1-S_{u,v})} \cdot R_{u,v}^{a_{u,v}} \cdot S_{u,v}^{1a_{u,v}}$,
   then $P'(\boldsymbol{A}) = \prod_{u,v \in V} \frac{1-R'_{u,v}}{1-R'_{u,v}(1-S_{u,v})} \cdot (R'_{u,v})^{a_{u,v}} \cdot S_{u,v}^{1a_{u,v}}$

where

$$p'_{u,v} = \begin{cases} 1 - (1 - p_{u,v}) \exp(\lambda_H), & \text{if } (u,v) \in W \\ p_{u,v}, & \text{otherwise} \end{cases}$$

$$R'_{u,v} = \begin{cases} R_{u,v} \cdot \exp(\lambda_H), & \text{if } (u,v) \in W \\ R_{u,v}, & \text{otherwise} \end{cases}$$

Here λ_H *is a Lagrangian multiplier and a unique real number such that* $(1 - p_{u,v}) \exp(\lambda_H) \in (0\ 1) \subset \mathbb{R}$ *and* $R_{u,v} \in (0\ 1) \subset \mathbb{R}$.

It is observed that background distribution $P(\mathbf{A})$ can be updated using Theorem 3.1. For Belief-c and Belief-i claim 1 is followed, while for Belief-m we follow claim 2, where $R_{u,v} = \exp(\lambda_u^r + \lambda_v^c)$ and $S_{u,v} = \exp(\mu_u^r + \mu_v^c)$.

Both claims in Theorem 3.1 follow the same principle, hence for brevity, only the proof of claim 2 is given in Appendix A.2.

For the computation of aggregate deviation $\mathcal{AD}$, we require to compute the expected number of edges between two vertices given the background distribution. It is inefficient to update and store all the expectations every time the background distribution is updated. It is therefore recommended to only store the λ_H and compute the expectation whenever required. After a series of patterns $H = (W, E')$ are presented to the user $p'_{u,v}$ is given by $1 - (1 - p_{u,v}) \exp\left(\sum_{H:u,v \in W} \lambda_H\right)$, and $R'_{u,v}$ is given by $R_{u,v} \exp\left(\sum_{H:u,v \in W} \lambda_H\right)$.

3.5 Algorithm

To exhaustively solve Problem 3.1, we would have to consider all $2^{|V|}$ possible subsets of V, for each subset determine its vertex-induced subgraph, check if it is connected, and compute its interestingness. As there are hardly any possibilities for pruning this would lead to very large run-times and we resort to a greedy hill-climber, which was shown to give good solutions in little time in the simple graph setting [van Leeuwen et al., 2016].

As input Algorithm 3.1 takes a multigraph G^M, seed subgraph $H = (W, E')$, gulf matrix $\mathcal{G}$, and—for efficiency—corresponding interestingness $\mathbf{I}$ (i.e., $\mathbf{I}(H, \mathcal{G})$). For directed

Algorithm 3.1 Hill Climber

Input: Multigraph dataset $G^M = (V, E^M)$, seed subgraph $H = (W, E')$, gulf matrix $\mathcal{G}$, and interestingness of seed subgraph I

Output: Multigraph pattern H, a heuristic solution to Problem 3.1, together with its interestingness I

```
1: procedure HillClimber(G^M, H, 𝒢, I)
2:     H_a, I_a ← CheckGraphExtension(G^M, H, 𝒢, I)
3:     if I_a > I then
4:         H ← H_a, I ← I_a
5:         return HillClimber(G^M, H, 𝒢, I)
6:     else
7:         H_r, I_r ← CheckGraphReduction(G^M, H, 𝒢, I)
8:         if I_r > I then
9:             H ← H_r, I ← I_r
10:            return HillClimber(G^M, H, 𝒢, I)
11:        else
12:            return H, I
```

Algorithm 3.2 Evaluate Graph Extension

1: **procedure** CHECKGRAPHEXTENSION(G^M, H, $\mathcal{G}$, $\mathbf{I}$)
2: $H^* \leftarrow H$, $I^* \leftarrow \mathbf{I}$
3: **if** $type(G^M) = Undirected$ **then**
4: **for each** $u \in Neighbors(H, G^M) \setminus W$ **do**
5: $W' \leftarrow W \cup \{u\}$, $H' \leftarrow (W', E'_H)$, $I' \leftarrow I(H', \mathcal{G})$
6: **if** $I' > I^*$ **then** $H^* \leftarrow H'$, $I^* \leftarrow I'$
7: **else**
8: **for each** $u \in Predecessors(H, G^M) \setminus W_{out}$ **do**
9: $W'_{out} \leftarrow W_{out} \cup \{u\}$, $W' \leftarrow [W_{in}, W'_{out}]$, $H' \leftarrow (W', E'_H)$, $I' \leftarrow I(H', \mathcal{G})$
10: **if** $I' > I^*$ **then** $H^* \leftarrow H'$, $I^* \leftarrow I'$
11: **for each** $v \in Successors(H, G^M) \setminus W_{in}$ **do**
12: $W'_{in} \leftarrow W_{in} \cup \{v\}$, $W' \leftarrow [W'_{in}, W_{out}]$, $H' \leftarrow (W', E'_H)$, $I' \leftarrow I(H', \mathcal{G})$
13: **if** $I' > I^*$ **then** $H^* \leftarrow H'$, $I^* \leftarrow I'$
14: **return** H^*, I^*

multigraphs, each vertex is (virtually) split into two, one having in-degree equal to zero and the other having out-degree equal to zero, based on which corresponding concepts $Predecessors$ & $OutNode$ and $Successors$ & $InNode$, respectively, are defined. Hence, a directed (sub-)graph has two lists of vertices one of $OutNodes$, W_{out} and the other of $InNodes$, W_{in}, thus, $W = W_{in} \cup W_{out}$.

Description. Algorithm 3.1 initially tries to add neighboring vertices to the current subgraph (Line 2). If the addition of any neighbor vertex results in improved interestingness (L3), the addition is consolidated (L4) and the method recurses (L5). Otherwise, the algorithm eliminates, one by one, vertices from the current subgraph (L7-10) and checks whether this improves interestingness (L8). When no improvement can be made in any iteration, the procedure stops (L12).

Algorithm 3.2 and 3.3 are two subroutines that return the best addition or removal step possible respectively. Function $type(G^M)$ determines the type of graph; if the graph is undirected then vertices are added (Algorithm 3.2, L3–7) or removed (Algorithm 3.3, L3–7) one by one without distinguishing the type of neighbor as *predecessor* or *successor*,

Algorithm 3.3 Evaluate Graph Reduction

```
1: procedure CheckGraphReduction(G^M, H, 𝒢, I)
2:     H* ← H, I* ← I
3:     if type(G) = Undirected then
4:         for each u ∈ W do
5:             W' ← W \ {u}, H' ← (W', E'_H), I' ← I(H', 𝒢)
6:             if I' > I* then H* ← H', I* ← I'
7:     else
8:         for each u ∈ W_out do
9:             W'_out ← W_out \ {u}, W' ← [W_in, W'_out], H' ← (W', E'_H), I' ← I(H', 𝒢)
10:            if I' > I* then H* ← H', I* ← I'
11:        for each v ∈ W_in do
12:            W'_in ← W_in \ {v}, W' ← [W'_in, W_out], H' ← (W', E'_H), I' ← I(H', 𝒢)
13:            if I' > I* then H* ← H', I* ← I'
14:    return H*, I*
```

unlike in the case of directed graphs (Algorithm 3.2, 3.3; L8–13).

The proposed hill-climber, which is a greedy heuristic, may experience problems due to locally converging to a sub-optimal solution. This largely depends on the choice of seed (initial subgraph) provided to the algorithm. To overcome this pitfall, we propose to independently run the hill-climber for k different seeds and choose the best solution among the k returned patterns. The seeds can be chosen on the basis of different criteria; we consider the following three:

1. **Degree**: Select the top-k vertices having the highest degrees in the graph, where each individual vertex as a singleton graph is used as a seed once.

2. **Uniform**: Select k different vertices at random, where each individual vertex is used as a seed once.

3. **Interest**: Use the k most interesting vertices and use each of those individually as seed. The interestingness of a vertex is calculated as the subjective interestingness $\mathbf{I}$ of the vertex-induced subgraph of the vertex together with its immediate neighbors.

It is intuitively beneficial but cost-inefficient to evaluate all possible seeds (i.e., to use each vertex in a graph as independent seed). We demonstrate the effectiveness of the above-described seed selection strategies in Section 3.6.

Complexity. In a single iteration of the hill-climber interestingness computation is the most costly part of the computation and has complexity $\mathcal{O}(|W|^2)$, as aggregate deviation computation requires to sum elements in the gulf matrix. We can, however, maintain a list of potential vertices that can be added to the current subgraph, along with the potential gain in aggregate deviation associated with each candidate vertex. These potential gains are updated upon addition or removal of a vertex from the current subgraph, which has complexity $\mathcal{O}(|V|)$. As the complexity of the search procedure is identical, the resulting overall complexity is $\mathcal{O}(|V|)$.

3.6 Experiments

In this section, we evaluate our proposed approach and compare it to related methods. To distinguish the results obtained using different types of prior beliefs, we denote our proposed approach using the background distribution given by Belief-c as SIMP-c; by Belief-i as SIMP-i; and by Belief-m as SIMP-m. For the experiments we use both synthetic and real multigraphs.

Datasets. We generate synthetic datasets in two steps. First, a simple, undirected graph is generated using the preferential attachment method by Barabási and Albert [1999]. Second, a randomly generated sequence is used to add parallel edges to make it a multigraph. This sequence has a length equal to the number of edges in the simple graph, and is a combination of a Bernoulli (parameterized by the probability of success p_b) and geometric distribution (parameterized by p_g). The former determines whether parallel edges are added, while the latter determines how many parallel edges are added to the vertex-pair indicated by the index in the sequence (if any). For the Barabási-Albert model, parameter l is used to define the maximum number of vertices to which a newly inserted vertex should be connected. Parameter values and properties of the resulting four synthetic datasets are shown in Table 3.1, where superscripts S and M refer to the initial *s*imple graph and the final *m*ultigraph, respectively.

Table 3.1. Properties of the multigraph datasets: number of vertices ($|V|$), number of edges ($|E^M|$), number of edges in a simple graph projection ($|E^S|$), probabilities of success for generating multigraph sequences (p_b and p_g), and Barabási-Albert model parameter (l).

DS	p_b	p_g	l	$\|\mathbf{V}\|$	$\|\mathbf{E}^M\|$	$\|\mathbf{E}^S\|$
SYN1	0.2	0.40	10	200	2628	1900
SYN2	0.2	0.65	10	1000	12977	9900
SYN3	0.4	0.80	10	10000	149729	99900
SYN4	0.2	0.65	10	50000	653821	499900
DBLP1	-	-	-	5271	19888	16847
DBLP2	-	-	-	6956	23879	20837
DBLP3	-	-	-	18466	98493	78699
DBLP4	-	-	-	65074	230006	202642
IMDB	-	-	-	4644	13416	12702

From the DBLP[3.6] data, we generate a co-author graph, where authors are represented as vertices and co-authored publications as undirected edges. Due to its large size, we have created multiple datasets from the data using different queries: 1) all conference publications of October 2017 (DBLP1) and July 2017 (DBLP2); 2) all publications of the top-20[3.7] conferences of Data Mining, Machine Learning and Artificial Intelligence in 2016–2017 (DBLP3); and 3) all journal publications of May 2017 (DBLP4). To obtain the IMDB[3.8] dataset we build a co-actor graph, where actors are represented as vertices and common movies as undirected edges. For each dataset, we only consider the largest connected component.

Evaluation criteria. We characterize the results using several commonly used subgraph properties: the number of vertices $|V|$; the number of edges $|E|$; density ρ, given by $(2 \times |E|)/(|V| \times (|V| - 1))$; average degree η, given by $2 \times |E|/|V|$; and diameter d. Further, to demonstrate the benefits of considering multigraphs over simple graphs, we 'project' the multigraph patterns, indicated by superscript M, to their simple graph

[3.6]source: https://dblp.uni-trier.de/

[3.7]source: https://scholar.google.co.in/citations?view_op=top_venues&hl=en&vq=eng

[3.8]source: https://www.kaggle.com/carolzhangdc/imdb-5000-movie-dataset

counterparts, indicated by superscript S, by removing any 'parallel' edges between each vertex-pair. We then define a new measure, denoted γ, to quantify the number of parallel edges in a subgraph relative to the number of vertices in a subgraph: $(|E^M| - |E^S|)/|V|$.

3.6.1 Prior Beliefs and Interestingness Evaluation

The different types of prior belief that we defined reflect different types of knowledge an analyst may have. Here we demonstrate the different effects of the proposed types of prior beliefs. The expectation on the number of edges between two vertices (or the probability distribution) varies with the prior knowledge as quantified using the maximum entropy principle (shown earlier).

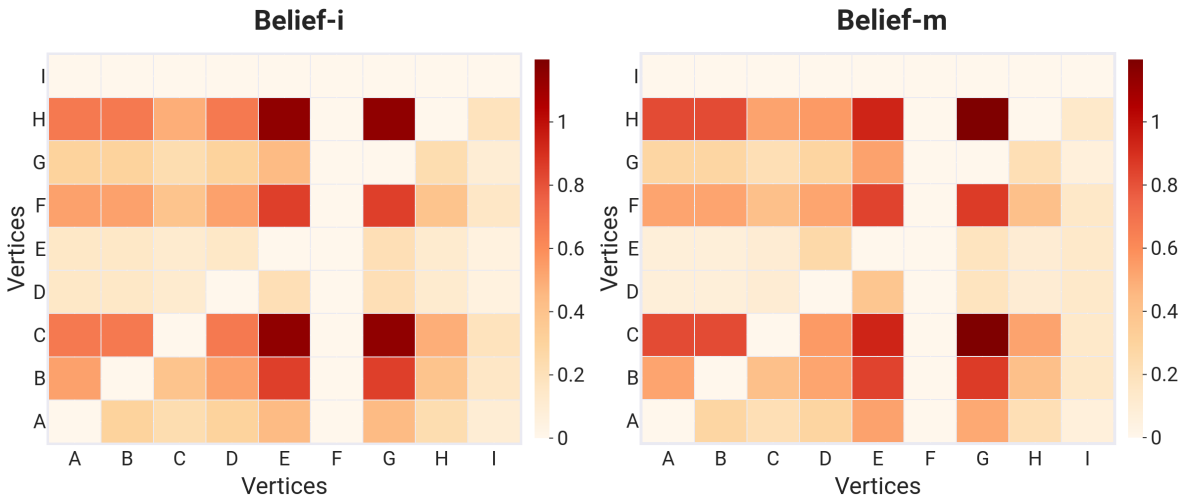


Fig. 3.2. Heatmap showing the expected number of edges between all pairs of vertices ($\mathcal{E}$) for the toy example (Fig. 3.1) w.r.t. Belief-i and Belief-m (dark colour represents higher expectations).

Belief-c results in a uniform distribution with equal expectation for all pairs of vertices. Thus, a subgraph with high average vertex degree would be considered most interesting under this type of belief, which is confirmed by the co-occurrence of high values of both interestingness (**I**-c) and average degree (η) in Table 3.3.

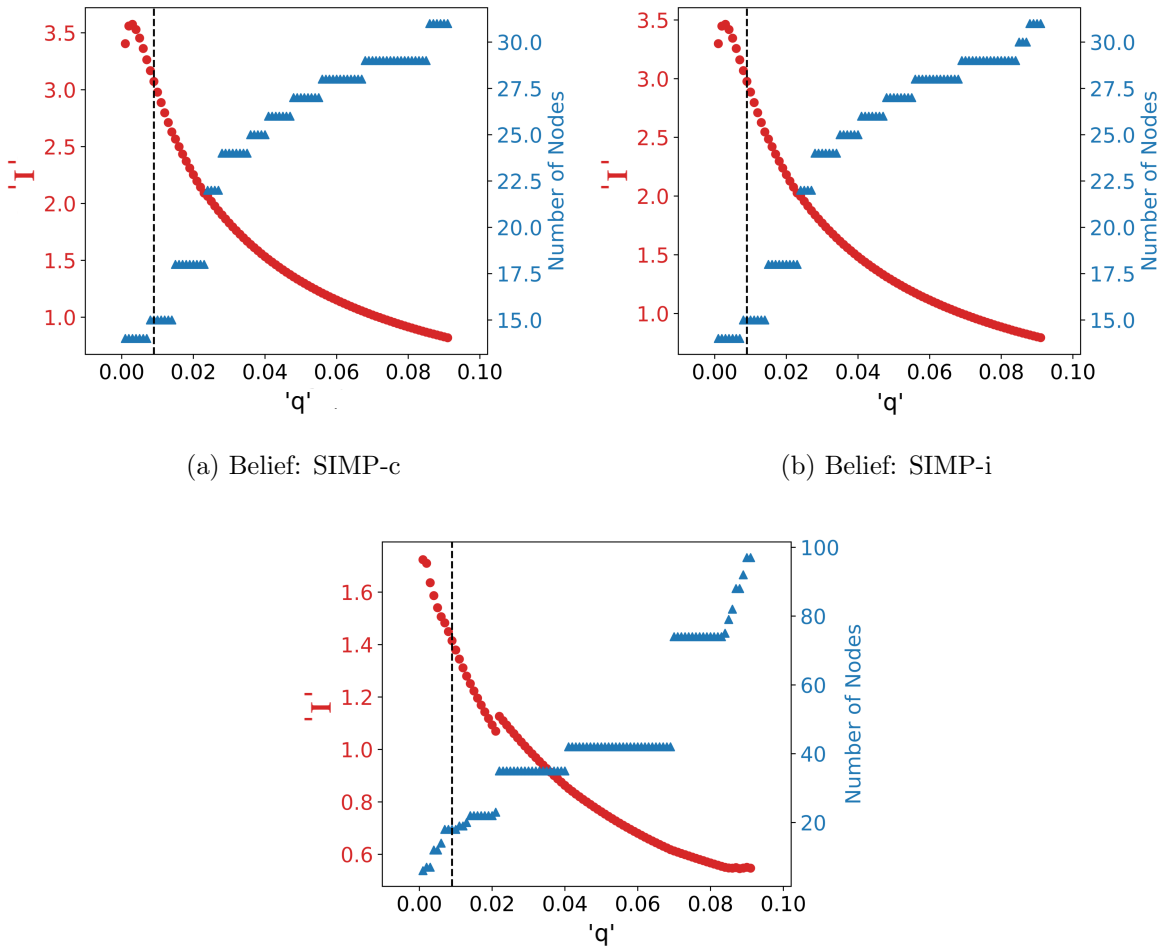
Belief-i and Belief-m represent more extensive forms of prior knowledge than Belief-c. Using the toy data set from Figure 3.1, the expectation between all pairs of vertices is shown in Figure 3.2 for both Belief-i and Belief-m. With Belief-i, it can be seen that

the highest expectation on the number of edges is for vertices C and E, as C has the highest number of outgoing edges and E has the maximum number of incoming edges in the graph. As subjective interestingness is defined as the positive deviations from the expectation, this type of belief usually leads to dense patterns (as can be witnessed from Table 3.3). Belief-m is more profound than Belief-i, as here the analyst has additional information on the number of unique neighbors for each vertex. With the addition of a new constraint, the expectation between vertices C & E decreases, as C has only two successors, which is compensated for by an increased expectation for the number of edges between vertex pairs C & A and C & B. In this particular case, these expectations are much closer to the actual values.

3.6.2 Description Length and Seeding Strategy Evaluation

In this subsection, we empirically demonstrate the effect of the value of parameter q as used in the description length. For most of the datasets, including the larger graphs, a value of 0.01 was found to be robust as it results in moderately sized patterns. Note that this corresponds to a belief that a pattern is expected to consist of 1% of all vertices in a graph. For the DBLP1 dataset, the effect of varying q is shown in Figure 3.3. The plots demonstrate how q can be used to influence pattern size as desired by the analyst. For the remainder of the chapter, we fix q to 0.01.

Next, we perform experiments on datasets SYN1, SYN2, DBLP1 and DBLP2, for different number of independent runs (represented by k) and for each type of seeding strategy. The results, aggregated over the four mentioned datasets, are shown in Table 3.2 (mean interestingness score and sum of the runtimes). We can observe that, in general, the highest mean subjective interestingness (**I**) was found using the *interest*-based seed selection strategy, followed by the *degree* based strategy, for all three types of belief. Further, we observe that the extra runtime needed for using all individual vertices as seeds is substantially larger than the improvement in subjective interestingness. The results show that $k = 10$ provides an adequate trade-off, saving substantially on runtime while hardly giving in on subjective interestingness. Hence, for all remaining experiments, we will use the *interest*-based seeding strategy with $k = 10$ independent runs.



(a) Belief: SIMP-c

(b) Belief: SIMP-i

(c) Belief: SIMP-m

Fig. 3.3. Parameter q vs the number of vertices (triangles) vs subjective interestingness (**I**, circles), for subgraphs found on DBLP1. The vertical dashed line indicates $q = 0.01$.

Table 3.2. Mean subjective interestingness (**I**) of the best pattern found using SIMP-c, SIMP-i, and SIMP-m, for 'Interest', 'Degree' and 'Uniform' seed selection strategies, with corresponding runtimes (in seconds).

Belief	**k**	**1**		**10**		**50**		**All**	
	Seed Type	**I**	**Time**	**I**	**Time**	**I**	**Time**	**I**	**Time**
SIMP-c	**Interest**	1.799	3.84	1.911	23.03	1.915	108.47	1.919	2758
	Degree	1.304	2.02	1.911	19.35	1.916	118.27		
	Uniform	0.844	2.62	1.453	26.75	1.456	124.69		
SIMP-i	**Interest**	1.592	2.26	1.602	3.50	1.602	9.41	1.607	412
	Degree	0.781	0.13	1.511	1.03	1.602	6.32		
	Uniform	0.439	0.33	0.720	1.66	1.156	7.54		
SIMP-m	**Interest**	1.015	2.50	1.170	5.27	1.170	13.47	1.173	591
	Degree	0.628	0.22	1.150	1.68	1.163	9.98		
	Uniform	0.449	0.24	0.717	4.80	1.094	13.67		

3.6.3 Quantitative Evaluation

In this section, we demonstrate that 1) our proposed subjective interestingness measure is different from existing measures designed for simple and multigraphs, and 2) the hill-climber finds subgraphs with large subjective interestingness scores. We empirically compare to 1) the modularity-based approach by Clauset et al. [2004] and 2) subjective interestingness for subgraphs (SSG) [van Leeuwen et al., 2016], as those are the closest to our approach and representative for the classes of methods they belong to. Note that neither is designed for mining patterns from multigraphs; we compare to these methods nevertheless to demonstrate that the task of mining patterns from multigraphs is very different from mining patterns from simple (unweighted or weighted) graphs in important ways, and therefore deserves the attention it gets in this thesis.

Since SSG is designed for simple, unweighted graphs, the datasets are converted to simple graph by removing parallel edges. For fair comparison on the task of mining multigraphs, the evaluation criteria are computed on the original multigraph. For the method by Clauset et al. [2004], to which we will also refer as CNM, we use its implementation in

Table 3.3. Properties (see text) of the best pattern found by each method.

DS	Method	$\|V\|$	$\|E^M\|$	$\|E^S\|$	I-c	I-i	I-m	ρ	η	d	γ
SYN1	SIMP-c	31.66	400.50	205.00	**1.51**	0.78	0.94	1.441	**24.77**	2.90	6.34
	SIMP-i	5.42	40.46	7.22	1.03	**0.94**	1.21	5.468	15.31	2.38	**6.50**
	SIMP-m	4.72	35.18	5.98	0.98	0.76	**1.26**	**5.964**	15.00	2.08	6.39
	SSG-c	10.98	64.23	48.90	0.77	0.21	0.20	1.170	11.69	1.88	1.39
	SSG-i	3.70	6.24	4.58	0.68	0.32	0.28	1.290	3.30	**1.36**	0.46
	CNM	30.54	159.83	98.95	0.32	0.43	0.39	0.360	10.34	3.40	2.00
SYN2	SIMP-c	90.74	1123.52	775.00	**1.65**	0.47	0.48	0.279	**24.75**	3.02	**3.85**
	SIMP-i	18.84	101.44	39.60	1.31	**0.61**	0.59	0.752	10.79	3.88	3.41
	SIMP-m	23.02	147.16	66.96	1.39	0.54	**0.65**	0.691	12.51	3.56	3.56
	SSG-c	24.92	216.14	167.86	1.40	0.18	0.17	0.730	17.25	**2.00**	1.93
	SSG-i	6.22	11.60	9.00	1.07	0.21	0.22	**0.770**	3.61	2.30	0.40
	CNM	116.19	610.35	371.32	0.37	0.52	0.50	0.095	10.45	4.91	2.06
SYN3	SIMP-c	381.60	5806.46	3626.30	**2.09**	0.51	0.54	0.080	**30.44**	3.92	**5.72**
	SIMP-i	175.50	1135.00	546.70	1.45	**0.67**	0.61	0.075	12.92	5.06	3.35
	SIMP-m	161.46	1414.60	735.78	1.41	0.64	**0.77**	0.111	17.55	4.36	4.21
	SSG-c	79.78	956.40	703.62	1.65	0.61	0.43	**0.304**	23.98	**3.00**	3.16
	SSG-i	30.84	58.72	44.48	1.02	0.36	0.31	0.130	3.79	6.36	0.46
	CNM	903.89	4480.05	2589.45	0.38	0.52	0.48	0.010	9.89	6.85	2.09
SYN4	SIMP-c	1052.20	14422.60	9951.30	**1.80**	0.74	0.68	0.030	**27.42**	4.00	4.25
	SIMP-i	324.30	2864.42	541.68	1.43	**0.91**	0.88	0.055	17.59	9.11	6.70
	SIMP-m	418.45	3918.32	898.45	1.66	0.87	**0.99**	0.045	18.73	8.12	**6.81**
	SSG-c	280.36	3535.70	2705.70	1.35	0.14	0.17	**0.090**	25.21	**3.02**	2.96
	SSG-i	164.08	267.60	207.06	1.08	0.15	0.18	0.020	3.28	12.00	0.37
	CNM	4303.55	21044.65	12138.40	0.41	0.51	0.42	0.002	9.73	9.45	2.07
DBLP1	SIMP-c	15	524	105	**2.98**	2.89	1.31	**4.990**	**69.87**	**1.00**	**27.93**
	SIMP-i	15	524	105	2.98	**2.89**	1.31	**4.990**	**69.87**	**1.00**	**27.93**
	SIMP-m	18	406	125	2.91	2.87	**1.38**	2.654	45.11	2.00	15.61
	SSG-c	20	192	190	0.92	0.89	0.76	1.011	19.20	**1.00**	0.10
	SSG-i	20	190	190	0.91	0.90	0.78	1.000	19.00	**1.00**	0.00
	CNM	532	2010	1696	0.45	0.49	0.42	0.014	7.56	7.00	0.59

Table 3.3. (Continued) Properties (see text) of the best pattern found by each method.

DS	Method	$\|V\|$	$\|E^M\|$	$\|E^S\|$	I-c	I-i	I-m	ρ	η	d	γ
DBLP2	SIMP-c	30	448	435	**1.49**	1.46	1.42	1.030	29.87	1.00	0.43
	SIMP-i	30	448	435	1.49	**1.46**	1.42	1.030	29.87	1.00	0.43
	SIMP-m	30	448	435	1.49	1.46	**1.42**	1.030	29.87	1.00	0.43
	SSG-c	30	448	435	**1.49**	1.46	1.42	1.030	29.87	1.00	0.43
	SSG-i	30	448	435	1.49	**1.46**	1.42	1.030	29.87	1.00	0.43
	CNM	307	998	856	0.42	0.45	0.41	0.021	6.50	12.00	**0.58**
DBLP3	SIMP-c	140	14626	9692	**12.23**	9.48	10.31	1.503	**208.94**	2.00	**35.24**
	SIMP-i	142	14780	9843	12.22	**9.49**	10.30	1.476	208.17	2.00	34.77
	SIMP-m	140	14626	9692	12.23	9.48	**10.31**	1.503	**208.94**	2.00	**35.24**
	SSG-c	104	8215	5356	8.58	9.40	9.39	**1.534**	157.98	**1.00**	27.40
	SSG-i	139	14488	9591	12.19	9.45	10.29	1.511	208.46	1.00	35.23
	CNM	369	18320	13283	6.72	5.22	5.04	0.270	99.30	5.00	13.68
DBLP4	SIMP-c	55	1495	1485	**1.14**	0.94	0.91	**1.007**	**54.36**	**1.00**	0.18
	SIMP-i	71	1663	1653	1.05	**1.17**	1.13	0.669	46.85	2.00	0.14
	SIMP-m	71	1663	1653	1.05	1.17	**1.13**	0.669	46.85	2.00	0.14
	SSG-c	55	1495	1485	**1.14**	0.94	0.91	**1.007**	**54.36**	**1.00**	0.18
	SSG-i	55	1495	1485	1.14	0.94	0.91	**1.007**	**54.36**	**1.00**	0.18
	CNM	3905	13455	11255	0.44	0.46	0.43	0.002	6.89	19.00	**0.56**
IMDB	SIMP-c	137	1037	837	**1.05**	0.46	0.43	0.111	**15.14**	4.00	1.46
	SIMP-i	85	560	425	0.91	**0.57**	0.53	0.157	13.18	4.00	**1.59**
	SIMP-m	86	543	451	0.91	0.56	**0.54**	0.149	12.63	**3.00**	1.07
	SSG-c	72	480	410	0.87	0.44	0.43	0.188	13.33	**3.00**	0.97
	SSG-i	11	18	16	0.13	0.13	0.12	**0.327**	3.27	4.00	0.18
	CNM	657	2397	2113	0.42	0.31	0.26	0.011	7.30	7.00	0.43

For each combination of dataset and property the best value obtained by any method is highlighted in bold

iGraph[3.9], which supports weighted graphs. We transform each multigraph to a simple, weighted graph by replacing each 'multi-edge' with a single edge, with the number of edges as weight. Further, to be able to designate a 'most interesting pattern' for CNM, the pattern giving the highest mean score according to SIMP-c, SIMP-i and SIMP-m is used. *Note that this comparison is very favorable for CNM's method*, as we consider all patterns that the method generates, versus only the top-1 pattern discovered by SIMP (!). For synthetic data, we present averages over the most interesting patterns found on 50 different multigraphs, obtained using different seeds for multigraph generation.

Table 3.3 presents the results. The **I**-c, **I**-i and **I**-m columns show that our proposed hill-climber, by optimizing our multigraph interestingness measure on the multigraph data, was able to find subgraphs with higher scores than SSG and CNM, for all prior beliefs. The patterns found by SSG, however, are much smaller and have very few parallel edges, as witnessed by low values for γ. In general, all three of the proposed method—SIMP-c, SIMP-i, and SIMP-m—discover patterns with more parallel edges than the two baseline methods. For DBLP2 and DBLP4; CNM found patterns with the largest γ, but those patterns are very large and sparse, indicating that these are hardly informative. For some of the DBLP and IMDB datasets, the advantage of SIMP is quite large in terms of γ. Finally, the patterns found by SIMP-c, SIMP-i, and SIMP-m do not typically have a high density (ρ), which demonstrates that the proposed measure is different from ('objective') density.

Overall, it is shown that although SSG and SIMP are built on the same principles, they clearly quantify subjective interestingness of patterns differently, which leads to the identification of different patterns. While SIMP focuses on the occurrence of parallel edges, SSG only focuses on patterns with a smaller diameter. CNM provides similar results to SIMP-i, yet it yields large pattern as partitioning the dataset does not provide the user with an option to control the size of the patterns. Moreover, CNM's modularity measure necessarily always assign all vertices to a pattern, while SIMP-i can easily find few patterns containing only part of the graph.

It is also interesting to compare the results obtained by SIMP-c, SIMP-i, and SIMP-

[3.9]https://igraph.org/

Table 3.4. Properties of the top-10 patterns found by SIMP-c, SIMP-i, and SIMP-m, indicating the total computation time, the fraction of the vertices of the multigraph covered by all patterns combined, and the average Jaccard distance between all pairs of vertex sets.

DataSet	Time (in seconds)			Coverage			AvgJaccard		
	SIMP-c	SIMP-i	SIMP-m	SIMP-c	SIMP-i	SIMP-m	SIMP-c	SIMP-i	SIMP-m
SYN1	6.93	6.51	7.49	32.77%	21.82%	24.17%	0.95	0.90	0.96
SYN2	312.5	61.3	108.6	27.97%	16.01%	18.16%	0.93	0.95	0.97
SYN3	2674	2394	2462.9	11.34%	8.78%	9.89%	0.97	0.98	0.99
SYN4	8634	8435	8876	8.57%	6.54%	7.12%	0.94	0.97	0.98
DBLP1	871.8	828.8	835.6	3.09%	3.23%	2.98%	0.99	1.00	0.98
DBLP2	1025	1014	1024	3.16%	3.08%	3.18%	1.00	1.00	1.00
DBLP3	7443	7828	7522	2.66%	2.53%	2.58%	0.97	0.94	0.98
DBLP4	12659	11765	11828	1.08%	1.04%	1.05%	1.00	1.00	1.00
IMDB	493.8	215.1	276.5	12.64%	6.54%	6.98%	0.91	0.94	0.90

m. For almost all datasets, SIMP-c finds the pattern with the largest average multigraph degree, i.e., η, which is as expected since only a prior belief on the total number of edges in the network is assumed; all information on individual vertex degrees is assumed unknown. As expected, η is smaller for SIMP-i and SIMP-m results, and on the synthetic data SIMP-i and SIMP-m typically finds smaller subgraphs with larger densities and diameters. However, there is a trade-off among SIMP-c, SIMP-i, and SIMP-m for the measures ρ, η, $\mathbf{d}$ and γ, which demonstrates the flexibility of our proposed approach, where plugging in different prior beliefs lead to different results.

3.6.4 Qualitative Evaluation

In this section, we first demonstrate how iterative pattern mining results different yet partially overlapping patterns, and then present an external validation of the patterns found on the IMDB dataset.

Iterative pattern mining. As discussed in Subsection 3.4.1, our approach can be naturally utilized for iterative exploratory data mining to identify the top-K patterns in a multigraph. Table 3.4 shows the properties of the top-10 patterns found using SIMP-c, SIMP-i and SIMP-m. The patterns are evaluated based on total computation time taken

to find the ten patterns, coverage (i.e., the percentage of all vertices in a multigraph dataset covered by the union of the found 10 patterns), and average Jaccard (AvgJaccard) distance among the found patterns. The total computation time is mainly dependent on the size of the dataset and the expected size of the pattern by the analyst (altered with the supplied parameter 'q' used in description length; not shown). The results show that the proposed approach can be easily used on moderately large datasets, with around two hours of computation time needed to find the top 10 patterns in the most densely connected graph, SYN4. This time includes the initial computation of the background distribution, searching for the most interesting pattern with ten independent runs (seeds) of the hill-climber, and updating the background distribution after each iteration. The coverage values indicate that the proposed method finds patterns in different regions of the graph; the exact coverage varies depending on the dataset, its topology and its size. At the same time, the high AvgJaccard value indicates that overlap is largely avoided but small overlaps among vertex sets do occur.

In terms of runtime, updating the background distribution hardly affects the performance of the algorithm. The main factor affecting this step is the computation of a Lagrangian multiplier corresponding to the found pattern, which is computed using the bisection method—in practice this method is very fast compared to the overall runtime of the algorithm. Updating the background distribution in every iteration is essential to the process, as we can demonstrate empirically. That is, by updating the background distribution, the code length of the data—i.e., the number of bits required to encode the data under the background distribution—is expected to decrease; this can be regarded to represent the effect of learning based on the found patterns.

To investigate this, Figure 3.4 depicts the decrease in normalized code length of the IMDB dataset, for SIMP-c, SIMP-i, and SIMP-m, after each consecutive update of the background distribution. The code length of data $\mathbf{A}$ is given by $-\log_2 P(\mathbf{A})$, and in the plot, this is normalized by the code length of the data without any update, i.e., the length computed before learning but based only on the prior beliefs. We can observe that the negative loglikelihood of the data decreases over time, as the background distribution is updated using the found patterns. This clearly demonstrates how each consecutive

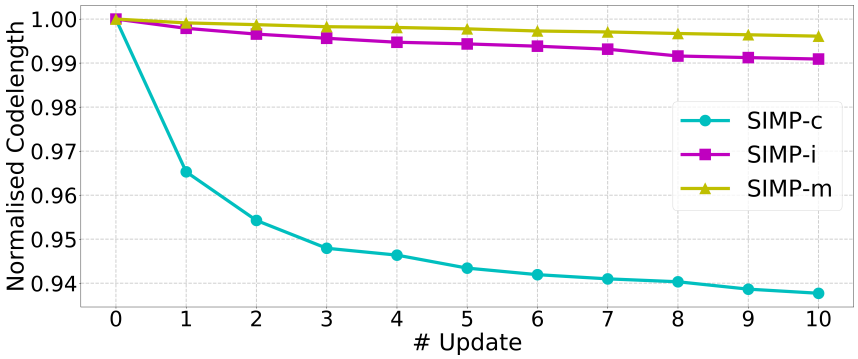


Fig. 3.4. Normalized code length of the IMDB dataset after each performed update, showing how each consecutive pattern adds new information to the set of patterns that is mined and therefore results in a shorter code for the data.

pattern adds new information to the set of patterns that is mined. Further, the relative decrease in code length is larger for SIMP-c than for SIMP-i and SIMP-m, which is also completely in line with our expectations as Belief-i and Belief-m represent more elaborate forms of prior knowledge; hence there is less to learn from the data.

External validation. In Table 3.5 we investigate how the found patterns are different and whether they could be meaningful to a domain expert. In the IMDB co-actor network, each edge corresponds to a movie in which the two actors (represented by the vertex pair) have worked together. Clearly, this naturally fits the multigraph setting, as co-actors can work together in multiple movies and each movie can be of a different genre. Genre information is not considered in the construction of the dataset or the prior belief and we, therefore, use this attribute to externally and objectively validate the semantics of the found patterns. For validation, we consider 26 different genres and the top-10 patterns found by SIMP-i, SIMP-m, SSG-i, and CNM. For each combination of genre and pattern, we conduct a hypergeometric test to assess whether a genre is significantly associated with the pattern. We compute the corresponding p-values and multiply them by the total number of tests per pattern, i.e., 26, as Bonferroni correction. All genres that are positively associated, i.e., have a p-value smaller than the threshold of 1e$-$4, are shown for the top-10 patterns found by each of the four methods.

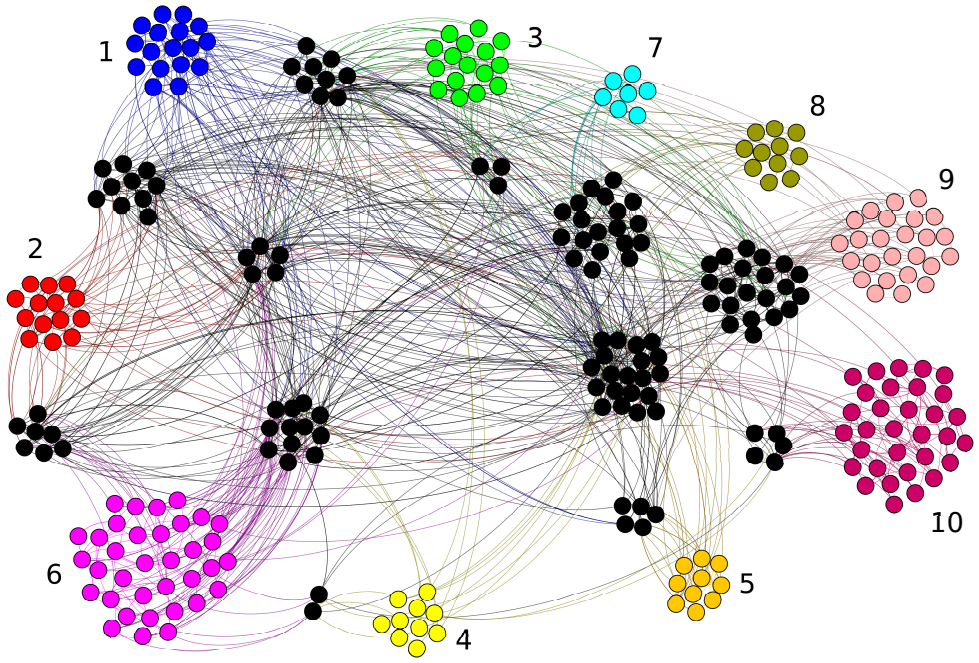
It is observed, in Table 3.5, mostly patterns found by SIMP-i and SIMP-m have more than one positively associated genre. This is mainly because of the presence of parallel

Table 3.5. Genres that are positively and significantly associated with the top-10 patterns found by SIMP-i, SIMP-m, SSG-i, and CNM, from the IMDB dataset, along with their respective Bonferroni corrected p-values ($< 1e{-}4$) (between brackets).

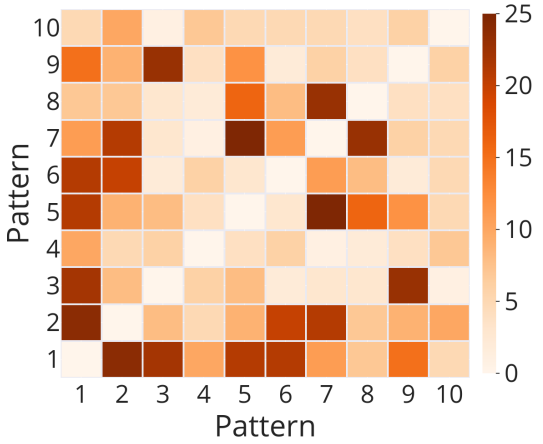
SN	SIMP-i	SIMP-m	SSG-i	CNM
1	Drama(0.0e+0), Crime(5.4e-10), Thriller(6.2e-16), Action(8.4e-6), Romance(2.2e-6)	Adventure(2.7e-12), Action(1.1e-5), Crime(2.8e-8)	Adventure(1.9e-7)	Adventure(2.1e-7), Drama(1.8e-14), Thriller(8.5e-8)
2	Adventure(0.0e+0), War(1.1e-12), Sci-Fi(5.9e-49), Action(1.1e-95), Family(2.0e-45), Thriller(0.0e+0), History(7.2e-10), Crime(9.0e-73), Romance(1.7e-96), Sport(8.7e-9), Biography(2.7e-20)	Sci-Fi(7.9e-12), Action(2.2e-8)	Drama(1.9e-6)	Comedy(1.3e-39)
3	Adventure(3.5e-61), Sport(6.3e-8), Sci-Fi(3.8e-36), Fantasy(2.4e-37), Family(3.6e-39), Action(1.3e-58), Crime(5.3e-52), Horror(1.5e-35), Thriller(1.8e-94)	Adventure(2.0e-5), Action(5.4e-9), Crime(1.2e-5)	—	Music(6.1e-11)
4	Romance(4.7e-10), Comedy(1.6e-11)	Adventure(1.7e-8), Fantasy(7.1e-9), Romance(2.0e-6)	—	—
5	Thriller(1.6e-22), Family(1.4e-7), Fantasy(6.0e-11), Sci-Fi(1.1e-11), Action(2.4e-13), Crime(7.3e-9), Comedy(1.4e-42), Adventure (4.2e-14)	Adventure(7.2e-5), Fantasy(2.0e-16), Family(3.3e-8)	Horror(6.3e-17)	—
6	Action(2.2e-11), Crime(6.0e-21), Sport(3.8e-13)	Sci-Fi(3.8e-15), Action(1.0e-5)	Adventure(2.2e-6), Action(8.4e-5)	—
7	History(8.6e-10), Crime(9.2e-5), Action(6.5e-15), Thriller(9.7e-12)	Adventure(1.1e-10)	Action(7.4e-13)	Action(1.4e-6)
8	Music(4.3e-11), Drama(1.7e-8)	Romance (6.1e-9)	Documentary (8.1e-7)	—
9	Action(4.3e-21), Horror(1.1e-7), Comedy(1.4e-40)	Action(5.3e-17), Crime(7.4e-7)	Western(5.8e-22)	—
10	Fantasy(1.2e-15)	Thriller(4.0e-6)	—	History(7.0e-11), Action(3.4e-12)

edges that correspond to different genres; two actors can work together in numerous movies that belong to different genres. The patterns found by SSG-i are mostly associated with one or no genre. This is indicative of the fact that SSG, by definition, considers patterns with a smaller diameter as more interesting, which is different from the proposed approach for multigraphs. CNM, on the other hand, was able to find patterns with more than one significantly associated genre, but not every pattern was significantly associated with one or more genres. This might be explained by the fact that CNM partitions the entire graph into several communities, which results in relatively large patterns that do not correspond to certain genres. The results show that the patterns found by each method are different; both SIMP variants tend to find patterns that more strongly correspond to genres.

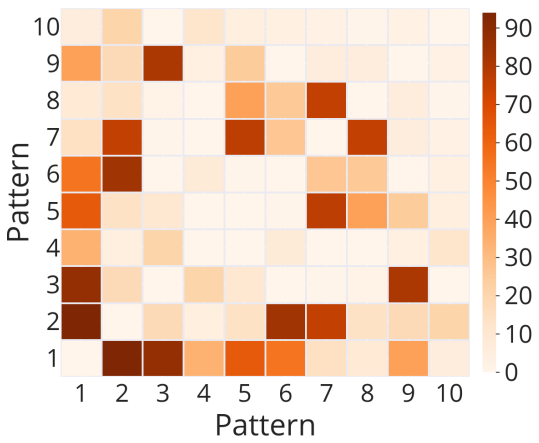
We further investigate the patterns found by SIMP-m by visualizing the resulting patterns in Figure 3.5. From the figures, we can observe that our approach succeeds in exploiting information about multiple edges between vertices, which results in the discovery of distinct yet partially overlapping patterns. From Table 3.5 we observe that patterns 1 and 3 are associated with the same set of genres, which might indicate that they are redundant or might be merged. Figure 3.5 shows that these patterns indeed share some vertices and edges, but are also different. Inspecting the data in more detail, we find that the actors with the highest degree in pattern 1 (but not in pattern 3) include Johnny Depp, Bruce Willis, Julia Roberts, and Robert Duvall. Similarly, actors present only in pattern 3 include Tom Hanks, John Ratzenberger, Delroy Lindo, and Sylvester Stallone. The overlapping region includes actors with very high degrees: Brad Pitt, J.K. Simmons, Morgan Freeman, and Kristen Dunst. Considering *actor's Facebook likes*, another feature present in the data, we find that the actors in pattern 1 (but not in pattern 3) have 8994 likes on average, versus 2973 on average for the actors in pattern 3 (not in pattern 1). The actors shared by both patterns on average have 10453 likes. Further, we also find that the union of patterns 1 and 3 would give an **I**-m of 0.503, which is clearly less than that of pattern 1, i.e., 0.538. All combined, the above analysis provides sufficient evidence to claim that pattern 1 and 3 indeed represent different, non-redundant 'actor communities', and are therefore rightfully considered to be two distinct patterns by our approach.



(a) network representation, with vertices present in more than one pattern shown in black colour (note that multiple edges between vertex pairs are depicted as a single edge to avoid cluttering the graph; see the other subfigures)



(b) pattern overlap in terms of vertices: for each pair of the top-10 patterns, the heatmap shows the number of vertices that are part of both patterns, i.e., $|W_1 \cap W_2|$ for every two mined subgraphs H_1, H_2



(c) pattern overlap in terms of edges: for each pair of the top-10 patterns, the heatmap shows the number of edges that are part of both patterns, i.e., $|E'_1 \cap E'_2|$ for every two mined subgraphs H_1, H_2

Fig. 3.5. Visualization of the top-10 patterns (numbered as per Table 3.5) found by SIMP-m in the IMDB dataset.

3.7 Summary

We proposed a novel subjective interestingness measure for subgraphs in multigraphs, taking into account both the given multigraph and different types of prior beliefs that the analyst may have. For the background distributions we used existing ideas based on the maximum entropy principle, but to quantify interestingness for multigraph patterns we used the properties of the background distribution to derive an expected number of edges for each pair of vertices. Following this, we proposed an effective hill-climber algorithm for mining the most interesting pattern from the data. Our experiments demonstrated that our subjective interestingness measure for multigraphs is different from existing definitions for other types of graphs, highlighting the benefits of taking the specific properties of multigraphs into account. The proposed algorithm was naturally extended for iterative exploratory data mining process. Using this characteristic of the proposed algorithm a number of overlapping yet different patterns were shown to be found. Also, the proposed algorithm was found to be scalable and accurate in iteratively finding interesting patterns. In the following chapters, we extend our approach to dynamic graphs. We also anticipate to explore the application possibilities of the proposed algorithm in the airline domain, which we will do in Chapter 6.

Chapter **4**

Subjective Summarization of Evolving Simple Graphs

Many real-world phenomena can be represented as dynamic graphs, i.e., networks that change over time. The problem of dynamic graph summarization, i.e., to succinctly describe the evolution of a dynamic graph, has been widely studied. Existing methods typically use objective measures to find fixed structures such as cliques, stars, and cores. Most of the methods, however, do not consider the problem of *online* summarization, where the summary is incrementally conveyed to the analyst as the graph evolves, and (thus) do not take into account the knowledge of the analyst at a specific moment in time.

In this chapter[4.1], we address this gap in the literature through a novel, generic framework for subjective interestingness for sequential data. Specifically, we iteratively identify atomic changes, called 'actions', that provide most information relative to the current knowledge of the analyst. For this, we introduce a novel *information gain* measure, which is motivated by the minimum description length (MDL) principle. With this measure, our approach discovers compact summaries without having to decide on the number of patterns. As such, we are the first to combine approaches for data mining based on subjective interestingness (using the maximum entropy principle) with pattern-based summarization (using the MDL principle).

We instantiate this framework for dynamic graphs and dense subgraph patterns, and present DSSG, a heuristic algorithm for the online summarization of dynamic graphs by means of informative actions, each of which represents an interpretable change to the connectivity structure of the graph. The experiments on real-world data demonstrate that our approach effectively discovers informative summaries.

[4.1] This chapter has been published as [Kapoor et al., 2021]: **Kapoor, S.**, Saxena, D.K. & van Leeuwen, M. Online summarization of dynamic graphs using subjective interestingness for sequential data. In *Data Min Knowl Disc*, Springer, **35**, 88–126 (2021).

4.1 Introduction

Many real-world phenomena, including interactions between people (e.g., social media, e-mail), web browsing, transport and logistics operations, and asset management, can be modelled in terms of the relationships between entities. That is, the corresponding data can be naturally represented as a network or *graph*, where vertices represent the entities and edges represent their relationships. When these relationships change over time, the graphs are called *dynamic graphs*.

The problem of *static graph summarization* has been widely studied, e.g., to efficiently store large volumes of data [Navlakha et al., 2008]; improve query efficiency [LeFevre and Terzi, 2010]; visualize large graphs [Koutra et al., 2014]; and provide high-level descriptions [Goebl et al., 2016]. Some of the popular methods rely on compression [Koutra et al., 2014], aggregation of vertices/edges [LeFevre and Terzi, 2010], or finding meaningful patterns [Goebl et al., 2016].

The need to incorporate the temporal dimension has led to the introduction of the problem of *dynamic graph summarization*. Lately, this problem has gained much attention. Here, the focus is on finding a minimal set of temporal structures that describe a dynamic network or graph. A typical way to achieve this is by considering a dynamic network as a sequence of static graph states/snapshots [Adhikari et al., 2017; Shah et al., 2015; Sun et al., 2007] and subjecting those to static graph summarization methods. Such sequences of static graphs, constructed by segmenting a dynamic graph into different states, can be referred to as *sequential data*. For instance, the method proposed by Shah et al. [2015], namely TimeCrunch, extends VoG, a method for static graph summarization by Koutra et al. [2014]. It creates a summary by stitching together the graph structures found in different snapshots while minimizing the global description length of the dynamic network. TimeCrunch uses a predefined vocabulary of graph structures, including cliques, stars, cores, and bipartite cores.

Most existing methods, however, do not consider the problem of *subjective online summarization*, where the summary is iteratively and incrementally conveyed to the analyst as the graph evolves. In that, the analyst is progressively updated on all changes up to

the current state of the network, relative to his/her prior knowledge. This problem has two key characteristics that differentiate it from posthoc summarization and therefore require a different approach. First, at any state, *it is only possible to use data that has been observed until this very moment*; it is impossible to use parts of the dynamic graph that lie in the future. Second, each change that is observed and communicated to the analyst should be *relative to what that analyst already knows about the graph.*

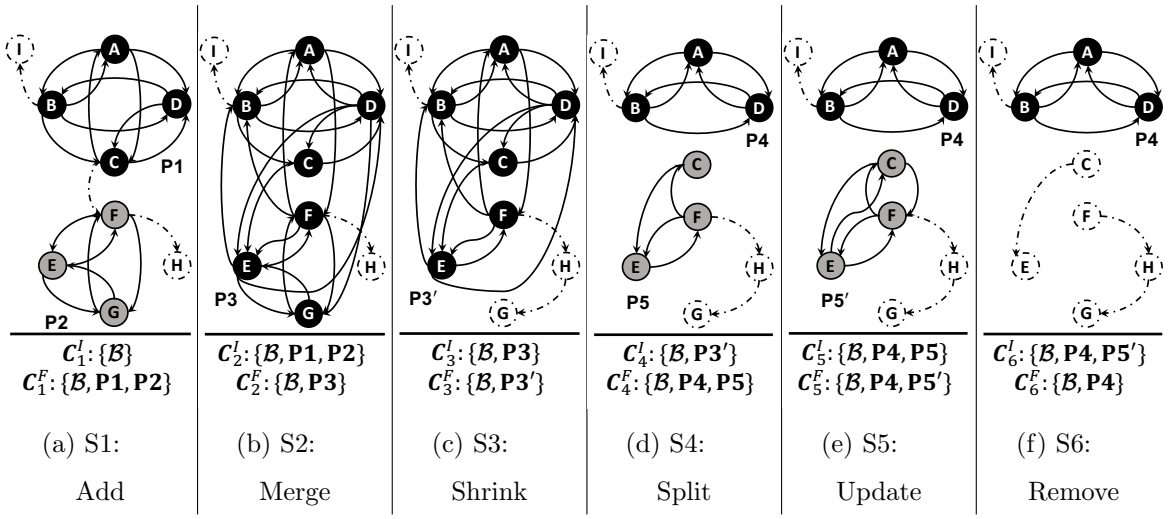
One motivation for such an approach comes from airline network analysis, where vertices represent airports and (directed) edges represent operating flights or routes between two airports. As the edges in an airline network change with time, it can be considered as a dynamic network. Here, an analyst may be interested in learning the *informative changes*, for example, as to how the traffic load is changing in real-time between different airports. An airline schedule is generated based on comprehensive knowledge on air traffic load management [Bazargan, 2016]. Hence, a domain analyst may well have prior knowledge/expectation at the block-hour level, of the total number of routes operated by an airline, total number of flights, number of unique routes from each airport, or even the densely connected set of airports. However, delays are a reality, as the schedules are not necessarily robust enough to perfectly factor and accommodate them. Hence, a compact and *subjective online summarization* bears real-time utility for airliners. It is *critical to note* that the application and utility of this approach is not limited to airline domain but spans across many other real-world scenarios, including evolving co-authorship network, co-actor network, and interaction network.

Our first significant contribution is the introduction of a novel, generic framework for subjective interestingness for sequential data. For this, we build on previous work by De Bie [2011b], who first introduced a formalization of subjective interestingness for exploratory data mining, in which the analyst's prior beliefs are modelled as constraints and a background distribution—representing the current knowledge of the analyst—is derived using the maximum entropy principle. The novelty of our framework for sequential data is two-fold. First, the patterns that we define, called 'actions', represent atomic changes to the data that provide information relative to the current knowledge of the analyst. Second, we introduce a novel *information gain* measure that is motivated by

the minimum description length (MDL) principle [Grünwald, 2007]. With this measure, our approach can automatically discover compact summaries without having to decide on the number of patterns. As such, we are the first to combine approaches for data mining based on subjective interestingness (using the maximum entropy principle) with pattern-based summarization (using the MDL principle).

Our second significant contribution is the instantiation of this generic framework for dynamic graphs. As van Leeuwen et al. [2016] instantiated subjective interestingness for dense subgraph discovery from (static) graphs, indeed we here build on their results. The concrete actions that we define, include `add`, `remove`, `update`, `shrink`, `split`, and `merge`. An instance of each of the action types is presented in Figures 4.1a-4.1f, for a toy example depicting an evolving airline network. Each of these actions adds, updates, and/or removes one or more dense subgraphs to/in/from the current summary, represented by set $\mathbf{C}_s$ for each state s. The set $\mathbf{C}_s$ comprises of the analyst's prior beliefs (represented by $\mathcal{B}$) and the dense subgraphs as patterns (represented by $\mathbf{P_i}$). In Figures 4.1a-4.1f, we indicate the initial summary $\mathbf{C}_s^I$ and final summary $\mathbf{C}_s^F$ after performing the actions in each state. By iteratively communicating these actions to the analyst, the analyst learns about the relevant changes in the graph (as shown in Figure 4.1g) *relative to what they already know*. The use of our information measure ensures that we always communicate actions that provide more information about the data than that is required to describe the patterns and corresponding actions, effectively making sure that the analyst always gains information.

Our third and final significant contribution is DSSG, a heuristic algorithm for the online summarization of dynamic graphs by means of iteratively discovering actions. Guided by the *information gain* criterion, it always considers all possible types of actions but only returns that action that provides the largest gain.



(g) Patterns and their changes over time.

Fig. 4.1. Toy example showcasing an evolving graph over six states (S1-S6), as summarized by background information $\mathcal{B}$ and patterns P1-P5′. (a-f) In each state s the initial and final summary are represented by $\mathbf{C}_s^I$ and $\mathbf{C}_s^F$, respectively; (g) Patterns P1-P5′ and corresponding add / merge / shrink / split / update / remove actions can be used to summarize the six consecutive states of the dynamic graph as depicted in a-f.

The remainder of the chapter is organized as follows. The relevant notation and preliminaries are discussed in Section 4.2. Our framework for subjective interestingness for sequential data and its online summarization is presented in Section 4.3.1 and Section 4.3.2, respectively, leading to the introduction of the problem of online summarization of dynamic graphs in Section 4.3.3. In this context, the DSSG algorithm is presented in Section 4.4. The experimental results on publicly available real-world datasets are discussed in Section 4.5. Important features of the proposed framework, key observations, limitations and future scope are discussed in Section 4.6, after which we conclude in Section 4.7.

4.2 Preliminaries

This section defines the notation adopted in this chapter, and briefly describes the observations from the two most closely related works on which we build in this chapter. These are 1) the framework for FORmalizing Subjective Interestingness in Exploratory Data mining (FORSIED) introduced by De Bie [2011b], and instantiated for different types of data and patterns; and 2) the work on Subjective interestingness of SubGraph patterns (SSG) in static graphs by van Leeuwen et al. [2016].

4.2.1 Data and Notation

A *rectangular dataset* is a matrix $\mathbf{D} \in \mathbb{D}^{M \times N}$, where the dimension of the dataset is given by $M \times N$ and $\mathbb{D}$ is the domain of an individual cell. A (simple) *graph* is denoted as $G^S = (V, E^S)$, where V is a set of vertices and E^S is a set of edges such that $u, v \in V$ for each edge $(u, v) \in E^S$. Its adjacency matrix is a rectangular dataset and hence, represented by $\mathbf{D} \in \mathbb{D}^{|V| \times |V|}$, where $\mathbb{D} = \{0, 1\}$.

A *dynamic dataset* $\mathbf{D}_T$ (rectangular) changes with time, where T is the timespan of the dataset. This time interval can be segmented into a number of consecutive intervals, where each interval $t = (t^b, t^f) \subset T$ represent a state s, such that t^b is the begin time and t^f is the finish time. For any two consecutive states, s and $s+1$, time t_s^f is equal to time t_{s+1}^b. Thus, a sequence of snapshots $\mathbf{D}_1, \ldots, \mathbf{D}_{\mathbb{S}}$ is observed, indexed by state $s \in \{1, \ldots, \mathbb{S}\}$, where $\mathbb{S}$ is the total number of states. Note that, in a sequence of snapshots, each D_s is a static rectangular dataset, such that $\mathbf{D}_s \in \mathbb{D}^{M \times N}$. We refer to such a sequence of snapshots as *sequential data*.

A *dynamic graph*, denoted $G_T = (V, E_T)$, is a graph in which each edge is present for a given period within time interval T, i.e., E_T is the set of edges that occur in time interval T. More specifically, each $e = (u, v, t^b, t^f) \in E_T$ defines an edge $u, v \in V$ that appears at start time t^b and continues to exist until it disappears at finish time t^f. Again, the time interval T can be segmented into a number of intervals, as seen earlier for dynamic datasets. This assumption implies that each $t \subset T$ defines a static state s of the dynamic graph, that is essentially a (simple) graph: each edge either exists or not. We denote the dynamic graph projected to its graph corresponding to a fixed time t by G_s, and

its corresponding adjacency matrix by $\mathbf{D}_s \in \mathbb{D}^{|V|\times|V|}$, such that $\mathbb{D} = \{0, 1\}$. Hence, a *dynamic/evolving simple graph*, G_T^S can be represented as a sequential dataset, $\mathbf{D}_T$ with a sequence of static simple graph snapshots $G_1^S, \ldots, G_{\mathbb{S}}^S$ and a corresponding sequence of adjacency matrices $\mathbf{D}_1, \ldots, \mathbf{D}_{\mathbb{S}}$.

Notably, even when time is not discrete, one can easily discretize it by segmenting it into equal-length intervals (e.g., seconds, minutes, ...). As we will see, the length of these intervals determines the granularity at which the approach will identify changes in the data. For instance, in the airline case, it is very unlikely that (relevant) changes will occur within seconds or even minutes, hence, it may be reasonable to segment time in hours.

4.2.2 Subjectively Interesting Patterns in Static Graphs

Informally, the FORSIED framework [De Bie, 2011b] defines subjective interestingness of a pattern as the information it provides with regard to the analyst's expectations (or prior knowledge), normalized by its complexity. Given a dataset $\mathbf{D}$, the analyst's *background distribution* P^*, is the distribution that maximizes entropy, is given by

$$P^* = \underset{P(\mathbf{D})}{\operatorname{argmax}} - \sum_{\mathbf{D}\in\mathbb{D}} P(\mathbf{D}) \log(P(\mathbf{D})), \tag{4.1}$$

$$\text{s.t.} \underset{\mathbf{D}\in\mathbb{D}}{\mathbb{E}}[f_i(\mathbf{D})] = \sum_{\mathbf{D}\in\mathbb{D}} P(\mathbf{D}) f_i(\mathbf{D}) = c_i, \forall i, \tag{4.2}$$

$$\sum_{\mathbf{D}\in\mathbb{D}} P(\mathbf{D}) = 1. \tag{4.3}$$

The set of constraints enforced in Equation 4.2 is presented in a generalized form, where each constraint $B_i \in \mathcal{B}$ is a pair consisting of a function f_i over $\mathbf{D}$—as properties of the data—and a corresponding constant c_i, i.e., $B_i = (f_i, c_i)$. The set of constraints $\mathcal{B}$ represents the analyst's prior knowledge or expectations on the data. The exact type(s) of constraints and their interpretation depends on the type and nature of the dataset $\mathbf{D}$.

Next, the interestingness of a pattern θ is defined as the ratio of the pattern's self-information (denoted $\mathcal{SI}$) to its description length (denoted $\mathcal{DL}$). Self-information is the negative log-probability that the pattern is present in the data, i.e., $-\log(P(\theta \in \mathbf{D}))$,

while description length is the number of bits required to describe or communicate the pattern to the analyst.

Instantiating these generic concepts for dense subgraph patterns in static simple graphs, van Leeuwen et al. [2016] defined interestingness $\mathbf{I}$ of a static graph pattern, $\theta = (W, k_W)$, denoting a vertex set W having k_W edges, as[4.2]

$$\mathbf{I}\left[(W, k_W)\right] = \frac{\mathcal{SI}\left[(W, k_W)\right]}{\mathcal{DL}\left[W\right]} = \frac{n_W \cdot \mathbf{KL}\left(\frac{k_W}{n_W} || p_W\right)}{|W| \cdot \log\left(\frac{1-q}{q}\right) + |V| \cdot \log\left(\frac{1}{1-q}\right)}, \tag{4.4}$$

where n_W is the number of possible edges in subgraph W, q is a hyperparameter representing the 'expected' probability of a random node to be present in W, and p_W is the probability of the subgraph occurring given background distribution P^*. The latter probability is computed as $p_W = \frac{1}{n_W} \sum_{u,v \in W} p_{u,v}$, where $p_{u,v}$ is the probability that an edge between vertices u and v exists as given by P^*.

Iterative learning. The framework above can be motivated by the observation that *compression equates learning* [Grünwald, 2007]: in order to learn as much as possible about the data, the implicit goal of the analyst is to (internally) represent the data using as few bits as possible. This observation implies minimizing $-\log P^*(\mathbf{D})$, i.e., the length of the data encoded by the background distribution. This can be accomplished by changing the analyst's knowledge on $\mathbf{D}$. Here, change in the analyst's knowledge on $\mathbf{D}$ implies that a new set of constraints $\mathbf{C}$ corresponding to each discovered pattern must be constructed, which is used to update the background distribution P^*. Specifically, when a graph pattern is discovered, a constraint is added to ensure that the updated expectations of the analyst conform with the actual number of edges. For instance, when a graph pattern (W, k_W) is presented to the analyst, a new constraint $C_W = (f_W, k_W)$ is added to $\mathbf{C}$, where f_W is a function over W vertices which counts the number of edges, i.e., $f_W(\mathbf{D}) = \sum_{u,v \in W, u<v} \mathbf{D}[u, v]$, and k_W is the actual number of edges in the vertex-induced subgraph of W vertices. Notably, the solution to the following problem provides the updated background distribution [van Leeuwen et al., 2016]:

[4.2] All logarithms in this chapter are to the base 2.

$$P^{*\prime} = \operatorname*{argmin}_{P} \sum_{\mathbf{D}} P(\mathbf{D}) \log \left(\frac{P(\mathbf{D})}{P^*(\mathbf{D})} \right), \tag{4.5}$$

$$\text{s.t.} \sum_{\mathbf{D}} P(\mathbf{D}) f_W(\mathbf{D}) \geq k_W, \tag{4.6}$$

$$\sum_{\mathbf{D}} P(\mathbf{D}) = 1. \tag{4.7}$$

Hence, the analyst can learn everything about the data by iteratively discovering the most interesting pattern and updating the background distribution after each iteration.

4.3 Proposed Approach

In this section, we introduce our novel framework for subjective interestingness for sequential data, which extends the FORSIED framework but also incorporates crucial changes. We introduce the problem of subjective summarization of sequential data, and to solve this problem we propose the method of online summarization of sequential data. Finally, we instantiate this generic problem for dynamic graphs.

4.3.1 Subjective Interestingness for Sequential Data

Given a sequential dataset $\mathbf{D}_T$, we consider the setting where an analyst is interested in learning informative patterns about the data as the snapshots unfold in an online fashion. As with static data, the analyst may have prior beliefs about the data already before the first snapshot—these are represented by a set of constraints $\mathcal{B}$.

When the snapshot corresponding to the first state is analyzed, we aim to find a compact set of constraints, i.e., patterns, that—together with the prior beliefs—minimize the negative log-probability of the data, given the implied background distribution. To avoid finding either too many or too complex patterns, we draw inspiration from the minimum description length principle [Grünwald, 2007] and use a two-part code to balance the goodness of fit of the data with the complexity of the constraint set. More precisely, we aim to find a new set of constraints $\mathbf{C}_1$ with corresponding background distribution P_1^* that minimizes $-\log P_1^*(\mathbf{D}_1) + L(\mathbf{C}_1)$, where L is a function that computes the encoded length for any given set of constraints. It is of note that we require an additional set of constraints $\mathbf{C}_1$ other than the existing set of constraints $\mathcal{B}$ to achieve the optimal

(feasible) solution of the above problem. The set of constraints $\mathbf{C}_1$ is used to ensure that the knowledge mined by the discovered patterns is reflected in the background distribution P_1^*.

For any consecutive snapshot, we now want to *adapt what the analyst has learned before*; by only providing the analyst with information about changes that have occurred in the data since the previous state, he requires minimal effort, and we obtain a minimal summary. Given the previous, this implies that—for each snapshot s after the first—we need to find a set of constraints $\mathbf{C}_s$ with corresponding background distribution P_s^* that minimizes $-\log P_s^*(\mathbf{D}_s) + L(\mathbf{C}_s|\mathbf{C}_{s-1})$, where L is a function that computes the encoded length for any given set of constraints *given* another set of constraints; i.e., smaller changes require fewer bits.

With the given discussion, we formally introduce the following problem statement.

Problem 4.1 (Subjective Summarization of Sequential Data) *Given a sequential dataset $\mathbf{D}_T$, i.e., sequence of snapshots $\mathbf{D}_1, \ldots, \mathbf{D}_S$, and prior beliefs $\mathcal{B}$, find:*

- *for $\mathbf{D}_1$: a set of constraints $\mathbf{C}_1$ that minimizes $-\log P_1^*(\mathbf{D}_1) + L(\mathbf{C}_1)$, where P_1^* is computed using constraints $\mathcal{B} \cup \mathbf{C}_1$;*
- *for $\mathbf{D}_s$, with $s \in \{2, \ldots, S\}$: a set of constraints $\mathbf{C}_s$ that minimizes $-\log P_s^*(\mathbf{D}_s)$ $+L(\mathbf{C}_s|\mathbf{C}_{s-1})$, where P_s^* is computed using constraints $\mathcal{B} \cup \mathbf{C}_s$.*

4.3.2 Online Summarization of Sequential Data

Apart from the fact that optimally solving each iteration of Problem 4.1 would require to consider a very large search space, i.e., that of all possible constraints sets, we do not want to present unordered sets of constraints to the analyst: this would very likely overwhelm the analyst and therefore cause confusion. Instead, we prefer to present atomic changes to $\mathbf{C}$ to the analyst one by one, as is also done in the framework for static data. We will therefore now derive an approach that heuristically approximates Problem 4.1 by iteratively looking for the largest changes and communicating those to the analyst immediately.

After each atomic change α, also called *action*, the set of constraints $\mathbf{C}$ is updated to a new set $\mathbf{C}'$, and hence the background distribution P^* is updated accordingly. α reduces the negative log-probability of the data by updating the background distribution, and we define this reduction as Information Content, $\mathcal{IC}$.

Definition 4.1 (Information Content) *Given an action α, and constraint sets $\mathbf{C}$ (original) and $\mathbf{C}'$ (updated), we define the* information content *of α, denoted by $\mathcal{IC}$, as the difference between the length of the data encoded by the background distributions specified by constraint sets $\mathbf{C}$ and $\mathbf{C}'$:*

$$\begin{aligned} \mathcal{IC}(\alpha) = \mathcal{IC}(\mathbf{C}'|\mathbf{C}) &= -\log P^*_{\mathbf{C}}(\mathbf{D}) - (-\log P^*_{\mathbf{C}'}(\mathbf{D})) \\ &= \log P^*_{\mathbf{C}'}(\mathbf{D}) - \log P^*_{\mathbf{C}}(\mathbf{D}), \end{aligned} \tag{4.8}$$

where P^*_X is the MaxEnt probability distribution given a set of constraints X (i.e., using Equations 4.1-4.3).

An *action* on $\mathbf{C}$ can be categorized as one of the following:

1. Addition of a new constraint C, i.e., $\mathbf{C}' = \mathbf{C} \cup \{C\}$,
2. Deletion of a constraint C , i.e., $\mathbf{C}' = \mathbf{C} \setminus \{C\}$,
3. Update of an already present constraint $C \in \mathbf{C}$, i.e., replacing C with a constraint C', and hence $\mathbf{C}' = \mathbf{C} \setminus \{C\} \cup \{C'\}$.

Definition 4.2 (Description Length) *The* description length *of an action α, denoted $\mathcal{DL}(\alpha)$, is defined as the (minimum) number of bits required to encode the changes in the set $\mathbf{C}$ when communicated to the analyst.*

Remark 4.1 *Given a set of constraints $\mathbf{C}_{s-1}$, let $\mathcal{A}$ be an ordered set of actions performed on $\mathbf{C}_{s-1}$ to get an updated set $\mathbf{C}_s$, then the encoded length L of $\mathbf{C}_s$ is computed as:*

$$L(\mathbf{C}_s|\mathbf{C}_{s-1}) = \sum_{\alpha \in \mathcal{A}} \mathcal{DL}(\alpha). \tag{4.9}$$

We now have two different quantities associated with each atomic change α, i.e., $\mathcal{IC}$ and $\mathcal{DL}$. Maximizing $\mathcal{IC}$ and minimizing $\mathcal{DL}$ leads to our overall goal of minimizing $-\log P_s^*(\mathbf{D}_s) + L(\mathbf{C}_s|\mathbf{C}_{s-1})$. Thus, we discount $\mathcal{IC}$ with $\mathcal{DL}$ and perform the action with maximal difference at each step. We call this difference *information gain* and denote it by $\mathcal{IG}$.

Definition 4.3 (Information Gain) *Let α be an action that transforms a given set of constraints $\mathbf{C}$ into an updated set $\mathbf{C}'$. Then, the* information gain $\mathcal{IG}$ *on performing α on $\mathbf{C}$ is given by*

$$\mathcal{IG}(\alpha) = \mathcal{IC}(\alpha) - \mathcal{DL}(\alpha). \tag{4.10}$$

The process of online summarization begins with the initialization of background distribution $P_{\mathcal{B}}^*$ using the prior belief(s) $\mathcal{B}$ that an analyst may have. At the start of state 1, no patterns have been discovered yet, i.e., $\mathbf{C}_1 = \emptyset$, which implies $P_{\mathcal{B}\cup\mathbf{C}_1}^* = P_{\mathcal{B}}^*$. Then patterns with maximum $\mathcal{IG}$ (Equation 4.10) are discovered iteratively and for each such pattern a corresponding constraint C is added to $\mathbf{C}_1$ and hence the background distribution $P_{\mathcal{B}\cup\mathbf{C}_1}^*$ is updated (using Equations 4.5-4.7). Note that $\mathbf{C}_1$ is initially an empty set, thus the only action that can be performed on $\mathbf{C}_1$ is the addition of a new pattern. The process continues until no feasible action can be performed on set $\mathbf{C}_1$. Here, a feasible action is any action which satisfies a user-provided criteria, for example, to be in agreement with the MDL principle an action α it is recommended default that α is feasible if $\mathcal{IG}(\alpha) > 0$. The process then moves to the following state. For any state s (except state 1), $\mathbf{C}_s$ is initialized to the final $\mathbf{C}_{s-1}$ and $P_{\mathbf{C}_s}^*$ to the final $P_{\mathbf{C}_{s-1}}^*$. This is followed by iterative actions on $\mathbf{C}_s$ with maximal, $\mathcal{IG}$ until no feasible action can be performed.

4.3.3 Online Summarization of Evolving Simple Graphs

The concept of subjective summarization of sequential data can be directly adapted to dynamic graphs by segmenting such a graph into a sequence of static graph snapshots (see Section 4.2.1). By making the data type more specific, however, we can also instantiate the other components of the generic framework—e.g., actions, prior beliefs, constraints, and description length—with more precise definitions. As discussed earlier, a graph pattern, $\theta = (W, k_W)$ is a subgraph of $W \subset V$ vertices that is connected by k_W edges. Thus, by

definition a graph pattern is *connected*, i.e., there exists a path from every vertex to every other vertex. Note that, since we consider graph patterns, the definition of constraints follows the discussion in Section 4.2.2. Following, we introduce the following problem statement as an instance of Problem 1.

Problem 4.2 (Subjective Summarization of Evolving Simple Graphs) Given an evolving simple graph G_T^S consisting of a sequence of snapshots $G_1, \ldots, G_{\mathbb{S}}$, with $\mathbf{D}_s$ the corresponding adjacency matrix for a state s and prior beliefs $\mathcal{B}$, solve Problem 4.1 such that each pattern in every set $\mathbf{C}_s$ is a connected subgraph pattern.

As discussed previously, optimally solving Problem 4.2 requires to consider a very large number of possible constraint sets. Similarly, we heuristically address Problem 4.2 by iteratively communicating atomic changes, or *actions*, having maximal $\mathcal{IG}$ to the analyst. Based on the properties of a graph pattern and possible structural changes, we now formalize six specific types of actions which we use to communicate changes on graph data, as initially depicted in Figure 3.1.

The `add` action communicates a newly discovered subjectively dense subgraph pattern. In Figure 4.1a, two patterns, **P1** and **P2**, are identified and `add`ed in state S1. A `remove` action deletes a pattern that no longer holds in the current state, i.e., when the pattern is no longer connected and/or its density decreases substantially. An example is shown in Figure 4.1f, where a sparse pattern **P5′** is `remove`d in state S6—removing a constraint is informative when it has a positive $\mathcal{IC}$.

The other actions are `update`, `merge`, `shrink`, and `split`, which all represent modifications of constraint(s) already present in $\mathbf{C}$. When the density of a pattern corresponding to an existing constraint increases, this is communicated via `update`. Thus, a constraint $C = (f_W, k_W) \in \mathbf{C}$ is replaced by a similar but updated constraint $C' = (f_W, k'_W)$. In Figure 4.1e, pattern **P5** is updated to pattern **P5′** in state S5, when its density increases compared its density in state S4 (Figure 4.1d). By applying a `merge` action, two previous patterns are merged to form one new pattern. That is, two constraints $C_i = (f_{W_i}, k_{W_i}), C_j = (f_{W_j}, k_{W_j}) \in \mathbf{C}$ are replaced by a single new constraint $C' = (f_{W_i \cup W_j}, k_{W_i \cup W_j})$, such that the resulting pattern of vertices $W_i \cup W_j$ is connected.

Table 4.1. Conditions that must be met to perform an action α on a constraint C present in constraint set $\mathbf{C}$, with initial pattern θ_i, resultant pattern θ_f and density function ρ (defined as the ratio of the number of edges to the maximum possible number of edges in a graph).

Type of α	**is $C \in \mathbf{C}$?**	**$\rho(\theta_i)$ increases?**	**$\rho(\theta_i)$ decreases?**	**is θ_i connected?**	**is θ_f connected?**
`Add`	✗	—	—	—	✓
`Remove`	✓	✗	✓	✗	—
`Update`	✓	✓	✗	✓	✓
`Shrink`	✓	✗	✓	?	✓
`Split`	✓	✗	✓	✗	✓
`Merge`	✓	?	✗	✓	✓

✓: true, ✗: false, ?: may or may not be true, —: not applicable

An instance is presented in Figure 4.1b, where two patterns, **P1** and **P2**, are merged to create a new pattern **P3** in S2.

Actions `shrink` and `split` either reduce an existing constraint or decompose one into multiple constraints. A constraint is shrunk when the density of a pattern decreases with the evolution of the graph (see Figure 4.1c, where pattern **P3** shrinks to form pattern **P3′** in state S3). Similarly, a constraint can be decomposed into multiple new constraints if the pattern corresponding to an original constraint consists of two or more connected components (see Figure 4.1d, where pattern **P3′** splits into two new patterns **P4** and **P5** in state S4). In `shrink`, the original constraint $C = (f_W, k_W) \in \mathbf{C}$ is replaced by a new reduced constraint $C = (f_{W'}, k_{W'})$ such that $W' \subset W$. In `split`, on the other hand, a constraint $C = (f_W, k_W) \in \mathbf{C}$ is replaced by τ new constraints, $C_1 = (f_{W_1}, k_{W_1}), \ldots, C_\tau = (f_{W_\tau}, k_{W_\tau})$, such that $W_1, \ldots, W_\tau \subset W$ and $W_i \cap W_j = \emptyset, \forall i, j \in \{1, \ldots, \tau\}$.

The different conditions that must be satisfied for each of the six types of actions to be applicable are summarized in Table 4.1.

Next, the formulation of information content $\mathcal{IC}$ and description length $\mathcal{DL}$ of each action type is summarized in Table 4.2. We extend the abstract definition of description length given in the previous section (Definition 4.2). The description length of an action is the summation of two parts, the first of which encodes the type of action, represented by

Table 4.2. Shown are the formulation of Information Content ($\mathcal{IC}$) and Description Length ($\mathcal{DL}$) for each defined atomic change, α.

α	$\mathcal{IC}$	$\mathcal{DL}$
`Add`	$\log P^*_{\mathbf{C}\cup C}(\mathbf{D}) - \log P^*_{\mathbf{C}}(\mathbf{D})$	$\mathcal{T}_\alpha + \mathcal{T}_{k_W} + \mathcal{T}_W$
`Remove`	$\log P^*_{\mathbf{C}\setminus C}(\mathbf{D}) - \log P^*_{\mathbf{C}}(\mathbf{D})$	$\mathcal{T}_\alpha + \mathcal{T}_C$
`Update`	$\log P^*_{\mathbf{C}\setminus C\cup C'}(\mathbf{D}) - \log P^*_{\mathbf{C}}(\mathbf{D})$	$\mathcal{T}_\alpha + \mathcal{T}_C + \mathcal{T}_{k_{W'}}$
`Shrink`	$\log P^*_{\mathbf{C}\setminus C\cup C'}(\mathbf{D}) - \log P^*_{\mathbf{C}}(\mathbf{D})$	$\mathcal{T}_\alpha + \mathcal{T}_C + \mathcal{T}_{k_{W'}} + \mathcal{T}_r + \mathcal{T}_{rw}$
`Merge`	$\log P^*_{\mathbf{C}\setminus\{C_i,C_j\}\cup C'}(\mathbf{D}) - \log P^*_{\mathbf{C}}(\mathbf{D})$	$\mathcal{T}_\alpha + 2\times\mathcal{T}_C + \mathcal{T}_{k_{W'}}$
`Split`	$\log P^*_{\mathbf{C}\setminus C\cup\{C_1,\ldots,C_\tau\}}(\mathbf{D}) - \log P^*_{\mathbf{C}}(\mathbf{D})$	$\mathcal{T}_\alpha + \mathcal{T}_C + \mathcal{T}_\tau + \mathcal{T}_{\lvert W_\tau\rvert} + \mathcal{T}_{W_\tau} + \mathcal{T}_{k_{W_\tau}}$

$\mathcal{T}_\alpha = \log(l)$, $\mathcal{T}_C = \log(|\mathbf{C}|)$, $\mathcal{T}_{k_W} = \mathrm{L}_{\mathbb{N}}(n_W - k_W + 1)$, $\mathcal{T}_{k_{W'}} = \mathrm{L}_{\mathbb{N}}(n_{W'} - k_{W'} + 1)$,
$\mathcal{T}_r = \mathrm{L}_{\mathbb{N}}(|\Psi|)$, $\mathcal{T}_{rw} = \log(|W|) + \log(|W|-1)\cdots + \log(|W| - |\Psi| + 1)$,
where Ψ is the set of vertices removed, $\mathcal{T}_W = |W|\log(q) + (|V| - |W|)\log(1-q)$,
$\mathcal{T}_\tau = \mathrm{L}_{\mathbb{N}}(\tau)$, $\mathcal{T}_{|W_\tau|} = \sum_{i=1}^{\tau} |\mathrm{L}_{\mathbb{N}}(W_i)|$, $\mathcal{T}_{W_\tau} = \log(|W|) + \log(|W|-1) + \cdots + \log(|W| - x + 1)$
—where $x = |\bigcup_{i=1}^{\tau} W_i|$, $\mathcal{T}_{k_{W_\tau}} = \sum_{i=1}^{\tau} \mathrm{L}_{\mathbb{N}}(n_{W_i} - k_{W_i} + 1)$

type(α), and the second of which encodes the details, represented by *details*(α). For all quantities where the upper limit is not known, we use the universal integer code [Rissanen, 1983], which is given by $\mathrm{L}_{\mathbb{N}}(n) = \log(2.865064) + \log(n) + \log\log(n)\ldots$ and sums over all positive terms. If the upper limit is known then we use the uniform code [Grünwald, 2007], given by $\log(n)$. Note that all logarithms are to base two.

In the description length of α, to describe the type of action we use the uniform code over all possible action types as there is no priority or bias towards any action. Thus, $\mathcal{DL}(type(\alpha)) = \mathcal{T}_\alpha = \log(l)$, as we require $-\log\frac{1}{l}$ bits. Here, $l = 6$ as we have defined six action types above. The computation of $\mathcal{DL}(details(\alpha))$ for each action type is shown in Table 4.2. That is, *details*(`add`) is the summation of the number of bits required to describe the set of vertices ($\mathcal{T}_W = \mathcal{DL}[W]$, see Equation 4.4), and the number of edges in the corresponding vertex-induced subgraph. Instead of describing the number of edges in a subgraph, we describe the number of edges short in a subgraph when compared to a clique of same number of vertices. That is, for a subgraph having W vertices, n_W is the maximum number of edges possible between W vertices, and k_W is the number of

edges, then we describe the difference between n_W and k_W, given by $\mathcal{T}_{k_W}$. Thus, a dense subgraph with high number of edges would have smaller description length, which favours discovery of dense subgraph patterns. Note that, the hyperparameter 'q' in $\mathcal{T}_W$ can be used to influence the size of pattern (see Section 4.2.2).

In `remove`, `update`, `shrink`, and `split`, the index of the constraint to be removed is communicated in $\mathcal{T}_C$ bits. Similarly, in case of `merge` the index of two constraints are communicated in $2 \times \mathcal{T}_C$ bits. Since we only consider the merge of two constraints at a time, the term $\mathrm{L}_{\mathbb{N}}(|2|)$ is omitted. In addition, for all the actions except `remove` the information about the edges is communicated in $\mathcal{T}_{k_{W'}}$ bits. In case of `shrink`, terms $\mathcal{T}_r$ and $\mathcal{T}_{rw}$ indicate the number of bits required to describes the number of vertices removed from the original pattern and the removed vertices, respectively. In `split`, the number of resulting constraints is described in $\mathcal{T}_\tau$ bits, number of vertices in each constraint in $\mathcal{T}_{|W_\tau|}$ bits, vertices in each constraint in $\mathcal{T}_{W_\tau}$ bits, and information about edges in each component using $\mathcal{T}_{k_{W_\tau}}$ bits.

Lemma 4.1 *For an action α, which updates a set of constraints $\mathbf{C}$ to $\mathbf{C}'$, $\mathcal{IC}(\alpha)$ as defined in Definition 4.1 is equal to*

$$\mathcal{IC}(\alpha) = \log P^*_{\mathbf{C}'}(\mathbf{R}) - \log P^*_{\mathbf{C}}(\mathbf{R}), \tag{4.11}$$

where $\mathbf{R}$ is a submatrix of $\mathbf{D}$ given by $\mathbf{R} = \mathbf{D}[W_1, \ldots, W_M; W_1, \ldots, W_M]$, such that W is the set of M vertices covered by the affected constraint(s)[4.3]*, C_α.*

Proof 4.1 *The proof is straightforward, however, for completeness we provide the following details. In Equation 4.8, $\log P^*_X(\mathbf{D}) = \sum_{i,j \in V} \log P^*_X(\mathbf{D}_{ij})$ is the sum over all pairs of vertices. These pairs can be categorized into three groups, which are 1) both vertices lie in W, 2) neither of the vertices lie in W, and 3) either (but not both) of the vertices lie in W. It is only in the first case that the probability is updated on performing the action α, while the rest of the probability terms remains unchanged and hence, these terms cancel out each other, i.e., $\log P^*_{\mathbf{C}'}(\mathbf{D}_{ij}) = \log P^*_{\mathbf{C}}(\mathbf{D}_{ij})$. Thus, the result follows.*

[4.3]The affected constraints C_α are those constraints (both original and updated) that are affected by action α. That is, if α transforms $\mathbf{C}$ to $\mathbf{C}'$ the C_α is defined to be all constraints in either $\mathbf{C}$ or $\mathbf{C}'$ that are not in both $\mathbf{C}$ and $\mathbf{C}'$

By virtue of Lemma 4.1, we come up with the following result.

Theorem 4.1 *The complexity of computing information content $\mathcal{IC}$ of an action α is $\mathcal{O}(|W|^2)$, where W is the set of vertices included in C_α.*

Proof 4.2 *The proof follows Equation 4.11 which is sum over all pair of vertices, $(i, j) : i, j \in W$. Hence, this requires a complexity of $\mathcal{O}(|W|^2)$.*

As discussed above, we solve Problem 4.2 by iteratively performing that action (of one of the six types defined above) with maximal $\mathcal{IG}$. Thus, we introduce the problem of online summarization of dynamic graphs (Problem 4.3). Hence, we heuristically unfold Problem 4.2 by iteratively solving Problem 4.3 at each step.

Problem 4.3 (Online Summarization of Dynamic Graphs) *Given the current state s, simple graph snapshot G_s, corresponding adjacency matrix $\mathbf{D}_s$, and current constraint set $\mathbf{C}_s$, perform that action 'α' from the set of all possible actions having maximal information gain $\mathcal{IG}$, such that the pattern(s) obtained after performing 'α' are connected subgraph(s).*

4.3.4 Additional Details

Prior Beliefs. We consider two different types of prior beliefs to constitute the set $\mathcal{B}$, which are direct adaptations of the beliefs proposed by van Leeuwen et al. [2016], as follows:

1. Belief-c: In this case, we model the analyst's knowledge about the total number of edges in the initial snapshot of the data. In other words, the analyst has prior knowledge about the relative edge density of the graph dataset. Solving Equations 4.1-4.3, De Bie [2011b] showed that P^* turns out to be product of independent Bernoulli distributions for each random variable $a_{u,v}$ and is given by

$$P^*(\mathbf{D}) = \prod_{u<v} \frac{exp((2 \cdot \lambda) \cdot a_{u,v})}{1 + exp(2 \cdot \lambda)}. \tag{4.12}$$

   This distribution is best represented as a matrix $P^* \in [0, 1]^{|V|\times|V|}$ with row and column indices indicating the vertices, such that $p_{u,v} = \frac{exp(2\cdot\lambda)}{1+exp(2\cdot\lambda)} = \rho(G_0)$ suggests the probability of $a_{u,v} = 1$, i.e., an edge between vertex u and v.

2. Belief-i: Similarly, here, the user possesses a belief about the individual degree of each vertex in a snapshot of the data. The maximum entropy distribution turns out to be a similar product of independent Bernoulli distributions, given as

$$P^*(\mathbf{D}) = \prod_{u<v} \frac{exp((\lambda_u + \lambda_v) \cdot a_{u,v})}{1 + exp(\lambda_u + \lambda_v)}, \tag{4.13}$$

where $p_{u,v} = \frac{exp(\lambda_u+\lambda_v)}{1+exp(\lambda_u+\lambda_v)}$ is the probability of random variable $a_{u,v} = 1$.

Updating the background distribution. When a pattern $\theta = (W, k_W)$ is discovered (through action `add`), a constraint $C = (f_W, k_W)$ is added to the set $\mathbf{C}$, and P^* is updated using Equations 4.5-4.7 [van Leeuwen et al., 2016], where the updated P^* is given as

$$P^*(\mathbf{D}) = \prod_{u<v} {p'_{u,v}}^{a_{u,v}} \cdot \left(1 - p'_{u,v}\right)^{1-a_{u,v}}, \tag{4.14}$$

where

$$p'_{u,v} = \begin{cases} \frac{exp(\lambda_u+\lambda_v+\lambda_W)}{1+exp(\lambda_u+\lambda_v+\lambda_W)} \ \textit{if } u, v \in W, \\ \frac{exp(\lambda_u+\lambda_v)}{1+exp(\lambda_u+\lambda_v)} \ \textit{otherwise} . \end{cases} \tag{4.15}$$

Thus, for all pairs $(u, v) : u, v \in W$ a unique Lagrangian multiplier, λ_W is introduced (using the bisection method) upon updating the background distribution. Similarly, if multiple constraints are present in $\mathbf{C}$, then $p'_{u,v}$ is computed as $\frac{exp(\lambda_u+\lambda_v+\sum_{C\in\mathbf{C}:u,v\in W}\lambda_W)}{1+exp(\lambda_u+\lambda_v+\sum_{C\in\mathbf{C}:u,v\in W}\lambda_W)}$. Hence, it is efficient to store only the Lagrangian multipliers and compute the probability whenever required.

If a `remove` action is performed then the corresponding Lagrangian multiplier is removed from the list to update the background distribution. Similarly, for all other actions, first the corresponding Lagrangian multiplier(s) to the original constraint(s) are removed and then using Equations 4.5-4.7, new Lagrangian multiplier(s) are computed. Hence, this is an efficient way to update the background distribution.

Feasibility Constraint. In order to provide the user with a concise summary we introduce a feasibility constraints to limit the number of actions performed in each state. That is, we consider an action feasible if the *information gain* is positive, i.e., $\mathcal{IG}(\alpha) > 0$.

Algorithm 4.4 DSSG

```
1: procedure DSSG(G_T, C, B)
2:     Compute Maximum Entropy Distribution for G_0 given B as P
3:     for each G_s ∈ G_T do          ▷ here G_T is a sequence of static graphs (snapshots)
4:         repeat
5:             A ← EvaluateAdd(G_s, P)
6:             R ← EvaluateRemove(G_s, P, C)
7:             U ← EvaluateUpdate(G_s, P, C)
8:             S ← EvaluateShrink(G_s, P, C)
9:             M ← EvaluateMerge(G_s, P, C)
10:            T ← EvaluateSplit(G_s, P, C)
11:            B ← GetBestAction(A, R, U, S, M, T)      ▷ Returns action with max. IG
12:            if B ≠ ∅ then
13:                Update C and P using B and Communicate B to the analyst
14:        until B ≠ ∅          ▷ Move to next snapshot if nothing is to be learned
```

Although, it may be altered as per user preference, this choice is motivated by MDL principle and ensures that an action always provide more information about the data than that it costs to describe the action.

4.4 The DSSG Algorithm

In this section, we introduce an algorithm called DSSG, of which the step by step procedure is outlined in Algorithm 4.4. DSSG is a heuristic approach to solve Problem 4.2 that works in an iterative manner, solving Problem 4.3 in each step. The overall procedure of DSSG can be summarized as follows.

DSSG starts with an initial graph snapshot G_0, an initial set of constraints $\mathcal{B}$ (as the analyst's prior belief), and a set of constraint $\mathbf{C}$ (which is usually $\emptyset$ initially). Given this, the maximum entropy distribution P is then computed (Line 2). For each state s (Line 3) actions are performed iteratively to solve Problem 4.3 (Lines 5-10). The process continues until no action can be performed (Line 14). Each performed action consists of an update of the background knowledge (updating P and $\mathbf{C}$) followed by communication

of the performed action B, to the analyst (Line 12). An example can be seen in Figure 3.1, where in each state the initial and final (represented by superscript I and F respectively) set of constraints is represented by $\mathbf{C}$ (indexed by subscript $s \in [1, T]$).

The feasibility constraint comes into effect while searching for the best action to be performed in each step (Line 11). The overall best action with the maximal value of $\mathcal{IG}$ is selected and returned. If the best action violates the feasibility constraint, then null is returned and the process continues with the next graph snapshot.

Algorithm 4.5 Find the most interesting pattern for addition

```
1: procedure SearchPattern(G_s, P, H*, I*)
2:     H ← H*, I ← i*
3:     for u ∈ Neighbors(H, G_t) \ W do                ▷ try if adding a vertex increases I
4:         W' ← W ∪ {u}, I' ← IG(add)
5:         if I' > I then W ← W', I ← I', H ← (W', k_W')
6:     if I > I* then
7:         return SearchPattern(G_s, P, H, I)
8:     else
9:         for u ∈ W do                                ▷ try if removing a vertex increase I
10:            W' ← W \ {u}, I' ← IG(add)
11:            if I' > I then W ← W', I ← I', H ← (W', k_W')
12:        if I > I* then
13:            return SearchPattern(G_s, P, H, I)
14:        else
15:            return (H*, I*)          ▷ If nothing increases I* return the found graph pattern
```

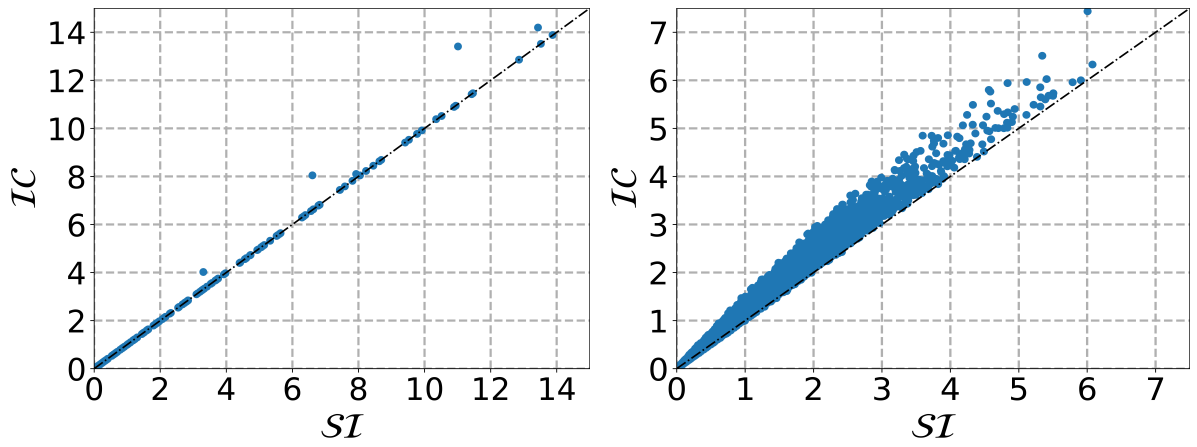
The EvaluateAdd procedure is used to discover the best new subgraph pattern with maximum $\mathcal{IG}$, which is a complex problem. This can be realized by the fact that the discovery of new pattern requires the evaluation of all possible $2^{|V|}$ candidate subgraphs. Hence, we use a hill climber based search algorithm (SearchPattern, see Algorithm 4.5) based on the SSG algorithm [van Leeuwen et al., 2016], which is proposed for finding a subjective interesting subgraph in a static simple graph. This algorithm starts with a

seed pattern H^* and recursively adds (Line 3-6) or removes (Line 10-13) vertices to find a pattern with a maximal value of $\mathcal{IG}$. This search stops if neither a vertex can be added nor removed (Line 17). To ensure the connectedness constraint, while adding vertices only vertices neighboring to vertices present in the pattern are checked (Line 3). As this hill climber is likely to suffer from convergence to local optima, we independently run the Algorithm 4.5 for a list of seed patterns [van Leeuwen et al., 2016] and select the single best pattern as search result. Further, note that the computational cost of naïvely computing $\mathcal{IG}(\texttt{add})$ at each step of the hill climber would be prohibitive, as it would require to compute a new Lagrangian multiplier to update the background distribution at each step. As this is the same problem as van Leeuwen et al. [2016] faced, we also adapt the same solution. That is, information content $\mathcal{IC}$ of a pattern $\theta = (W, k_W)$, as defined in Equation 4.8, is approximated by

$$\mathcal{IC}(\texttt{add}) \approx \mathcal{SI}(\theta) = n_W \cdot \mathbf{KL}\left(\frac{k_W}{n_W} || p_W\right). \tag{4.16}$$

We empirically show that Equation 4.16 is an adequate approximation of Equation 4.8 in Figure 4.2. To obtain Figure 4.2, we created a random graph of 20 vertices using the Barabási-Albert model and computed the values of $\mathcal{SI}$ (Equation 4.16) and $\mathcal{IC}$ (Equation 4.8) of all possible connected subgraphs, considering the two types of prior belief as discussed in Section 4.3.4. It is observed that for all candidate subgraphs (and for both types of prior belief) the value of $\mathcal{SI}$ is always less than or equal to $\mathcal{IC}$. Although they are not exactly equal, the correlation $r = 0.9999$ (in Figure 4.2a) and $r = 0.9948$ (in Figure 4.2b) are high enough to suggest that $\mathcal{SI}$ can be successfully used as proxy for $\mathcal{IC}$, as is also argued by van Leeuwen et al. [2016]. Moreover, computing $\mathcal{SI}$ is clearly much faster than computing $\mathcal{IC}$, as it does not require updating the background distribution at each step. Hence, this allows to discover surprisingly densely connected graph patterns from snapshots of the graph in an efficient way.

EvaluateRemove and EvaluateUpdate are used to evaluate each constraint in $\mathbf{C}$ to, either `remove` or `update` a constraint, respectively. In these procedures, each constraint in $\mathbf{C}$ is independently evaluated by computing the corresponding $\mathcal{IG}$. To compute $\mathcal{IC}$ (as in Table 4.2), we update the background distribution assuming that the action would take



(a) Belief-'c' (b) Belief-'i'

Fig. 4.2. Plots of $\mathcal{IC}$ vs $\mathcal{SI}$ of all connected subgraphs of a Barabási-Albert random graph of 20 vertices

place. Of note, the update in the background distribution is rolled back after evaluation of each constraint. Both of these method return the respective constraint with maximal $\mathcal{IG}$.

Similarly, EvaluateMerge returns two constraints (in **C**) or patterns which, when `merged`, result in a connected graph pattern with maximal $\mathcal{IG}$.

EvaluateShrink is used to evaluate each constraint in **C** for `shrink` and the reduced constraint with maximal $\mathcal{IG}$ is returned. To shrink a pattern or constraint, we use the procedure ShrinkPattern (Algorithm 4.6), which recursively removes vertices (Line 2-7) until no increase in $\mathcal{IG}$ is observed (Line 9).

EvaluateSplit is used find the constraint which produces maximal $\mathcal{IG}$ upon `split`. Note that, a new pattern that is the result of `split` may `shrink` in a next iteration; hence we also evaluate a possible reduction of each resulting pattern upon `split` using procedure ShrinkPattern. Thus, EvaluateSplit contains two parts: 1) first the different connected components in the original pattern are identified (each component acts as a new pattern or constraint), and 2) then each new pattern is evaluated for `shrink`.

Complexity. In a single iteration of DSSG, six different procedures are executed sequentially; hence, we discuss the complexity of each procedure. The EvaluateAdd

Algorithm 4.6 Find a candidate shrink pattern

```
1: procedure ShrinkPattern(H*, P, I*)
2:     for u ∈ W do
3:         W' ← W \ {u}, I' ← IG(α)
4:         if I' > I then W ← W', I ← I', H ← (W', k_W')
5:     if I > I* then
6:         return ShrinkPattern(H, P, I)
7:     else
8:         return (H*, I*)
```

procedure runs the hill climber SearchPattern independently, k times for k different seeds. In each iteration of this hill climber, the computation of $\mathcal{IG}$ is the most computationally expensive part, with time complexity of $\mathcal{O}(|W|^2)$ (from Theorem 4.1), where W is the set of vertices in a pattern. This hill climber is a direct adaptation of SSG and van Leeuwen et al. [2016] showed that this complexity can be reduced to $\mathcal{O}(|W|)$. Hence, if the number of neighbors in Algorithm 4.5 is (let's say) $\mathcal{N}$, then each iteration takes $\mathcal{O}(\mathcal{N}|W|)$. Thus, the worst-case complexity of running SearchPattern becomes $\mathcal{O}(\mathcal{I}\mathcal{N}|W|)$, assuming that the hill climber runs for at most $\mathcal{I}$ iterations.

In the other procedures, to evaluate each constraint in $\mathbf{C}$ requires the computation of $\mathcal{IG}$, which takes $\mathcal{O}(|W|^2)$ (from Theorem 4.1). Note that computing the Lagrangian multiplier corresponding to a revised constraint in $\mathbf{C}$ requires to run the bisection method, which has a complexity of $\mathcal{O}(n|W|^2)$. In this, n is the number of iterations required, computed as $\log \frac{\epsilon_0}{\epsilon}$, where ϵ is the given error or tolerance and ϵ_0 is the initial bracket size. Thus, the other procedures have a complexity of $\mathcal{O}(n|W|^2)$.

Given that the complexity of the overall algorithm strongly depends on the actual number of iterations, which cannot be computed in advance, we will instead mention empirical runtimes in the experiment section.

4.5 Experiments

In this section, we will demonstrate the efficacy of the proposed framework and the proposed algorithm, DSSG, by means of quantitative (Section 4.5.3) and qualitative (Sec-

tion 4.5.5) results on seven publicly available real-world datasets (Section 4.5.1).

4.5.1 Datasets

In this section, we will use the following seven publicly available datasets, also summarized in Table 4.3.

Table 4.3. Datasets along with some of their properties. Type indicates if the dataset is a Directed (D) or Undirected (U) graph, $|V|$ is the total number of nodes in the graph, $|E_S|$ is the total number of unique edges without timestamp, $|E_T|$ is the total number of unique edges with timestamp, T is the total time period for which the edges in the graph are considered, t is the time period covered by each individual state, and $|\mathbb{S}|$ is the total number of states considered for each dataset.

Dataset	**Type**	$\|V\|$	$\|E_S\|$	$\|E_T\|$	T	t	$\|\mathbb{S}\|$
High-School	U	327	5 818	20 448	5 days	1 hour	41
Workplace	U	217	4 274	11 730	10 days	1 hour	91
MathOverFlow	U	24 818	187 978	231 465	6.5 years	1 quarter	26
Reuters	U	7 403	105 343	159 977	66 days	1 day	66
TheMovieDB	U	8 292	236 691	249 324	10 years	1 year	10
DBLP	U	27 400	83 509	98 330	10 years	1 year	10
WebClicks	D	80 306	90 435	231 055	22 days	1 day	22

High-School Interaction[4.4]: This dataset has a total timespan of 5 days. In all 9 hours of interaction is available per day, except for the first day with 5 hours, and the total timespan is segmented into 41 different states of 1 hour each.

Workplace Interaction[4.4]: This is an interaction network of employees at a workplace. It has a total timespan of 10 working days, where interactions for 9 hours are available for each day, except for the first day where 10 hours of interactions are available. It is segmented into 91 different states of 1 hour each. Although the interactions are instantaneous in nature, an edge exists for each interaction which occurred in a state (snapshot).

[4.4]source: http://www.sociopatterns.org/

MathOverFlow[4.5]: This network captures the communication between users on the MathOverFlow website. A timestamped undirected edge exists between two users if one user answers another user's question, comments on another user's question, or comments on another user's answer to any question. The dataset has a total duration of 2 560 days. Here we consider a total timeperiod of 6.5 years, segmented into 26 states of 1 quarter (3 months) each. The lifespan of any edge is considered to be three months, i.e., an edge disappears 3 months after the time it appeared in the network.

Reuters Terror Network[4.6]: This dataset contains words that are present in each news article following the 9/11 terror attack. We build a network of words (as vertices) with a link between them (undirected edge) wherever they appear in the same article. The total time period considered is 66 days, with segments (snapshots) of 1 day each. In each state, the snapshot of the network contains all the words (and edges between them) if they appeared in any news article published on that day.

TheMovieDB: A network of actors (vertices) is considered, with an edge corresponding to a co-acted movie. The data is fetched using the TheMovieDB API[4.7]. The time period of the network is from year 2009 to 2016, and is segmented into 8 states of 1 year each. All movies in the 8 year time period having actors with popularity score more than 2 are included. Each snapshot contains edges corresponding to movies released in the same year.

DBLP: This is a co-author network, created using the DBLP[4.8] data of all publications in top-20 Machine Learning and Data Mining conferences[4.9] over a period of 10 years. The dataset is segmented into 10 states of 1 year each, adding an edge between two authors if they have co-authored at least one publication in the given year.

WebClicks: A network of click requests (directed edges) is created from referrer host to target host (nodes) for the time period between 1 November 2009 to 22 November 2009. To prune the data[4.10], we only consider edges with more than 25 requests in a day.

[4.5]source: https://snap.stanford.edu/data/sx-mathoverflow.html
[4.6]source: http://vlado.fmf.uni-lj.si/pub/networks/data/CRA/terror.htm
[4.7]source: https://www.themoviedb.org/documentation/api
[4.8]source: https://dblp.uni-trier.de/
[4.9]source: https://scholar.google.co.in/citations?view_op=top_venues&hl=en&vq=eng
[4.10]source: http://carl.cs.indiana.edu/data/websci2014/web-clicks-nov-2009.tgz

Also, the network is segmented into 22 states of 1 day each. That is, the edge remains only for 1 day, given that at least 25 requests were made from referrer host to target host.

4.5.2 Experimental Setup

The prior belief for each of the datasets, except for the TheMovieDB dataset, used in this chapter is type belief-c. For TheMovieDB type belief-i is used.

Since, we use an adaptation of the hill climber given by van Leeuwen et al. [2016], we fix the following parameters as suggested by van Leeuwen et al. [2016].

1. The parameter 'q' used in computation of the description length of pattern (see Table 4.2) is fixed at 0.01.

2. We use the 'interestingness' based 'TopK' seeding strategy with $k = 10$.

The experiments are executed on an Apple Macbook Pro 2018, with 2.3 GHz Quad-Core Intel Core i5 processor and 8GBs of RAM.

4.5.3 Quantitative Analysis

In this subsection, we demonstrate the performance of the proposed method on the above mentioned datasets. We evaluate the results in terms of 1) the type of *actions* performed in each state, 2) the number of patterns (or constraints) required to summarize each state, 3) the densities of the patterns found in each state, 4) the ratio of the vertices covered by the patterns in the dataset in each state, and 5) the compression ratio between the encoding cost of the data given the initial background distribution and given the final background distribution in each state. We also showcase the feasibility of the proposed approach by presenting the time taken for online summarization of all states in each graph dataset. Table 4.4 presents the results and summarizes the set of found patterns for each dataset by the proposed method.

Number of patterns required to summarize each state. We observe the lowest median number of patterns, i.e., 5 for Workplace and most, i.e., 62, for DBLP. This is expected as Workplace has the smallest number of vertices and DBLP has the second most number of vertices among all considered datasets. However, WebClicks has the largest number of vertices but surprisingly very few patterns are found to summarize each

Table 4.4. Properties of the found set of patterns (or constraints) $\mathbf{C}_s$ in each state s.

DataSet		$\lvert\mathbf{C}_s\rvert$	$\lvert\mathcal{A}\rvert$	$\Omega_{\mathbf{C}}$	Ω_s	$\bar{\rho}$	CR	Coverage	Runtime
	min	6	6	60	326	0.4104	0.47%	10.70%	
High-School	*median*	14	12	250	819	0.6247	8.01%	28.13%	307
	max	22	16	755	1247	0.7600	25.01%	58.41%	
	min	1	1	5	35	0.5176	0.36%	2.16%	
Workplace	*median*	5	5	50	193	0.7000	1.23%	3.23%	96
	max	11	11	626	997	1.0000	36.44%	28.57%	
	min	1	1	4 151	8 223	0.0070	0.63%	2.59%	
MathOverFlow	*median*	10	7	6 147	16 703	0.0179	4.38%	10.22%	74 849
	max	23	17	14 138	24 292	0.1226	46.20%	12.56%	
	min	5	1	177	322	0.0353	0.16%	2.20%	
Reuters	*median*	18	11	796	3 827	0.2630	5.44%	4.71%	79 049
	max	32	27	6 550	13 494	0.6693	19.50%	16.52%	
	min	2	9	3 725	22 145	0.0062	12.79%	6.07%	
TheMovieDB	*median*	27	45	6 379	42 586	0.3125	21.82%	10.93%	18 908
	max	118	118	15 556	74 499	0.9968	48.30%	17.62%	
	min	2	1	345	7 959	0.6208	1.03%	2.19%	
WebClicks	*median*	6	2	539	11 759	0.6814	2.41%	3.58%	14 576
	max	7	3	3 384	12 751	0.7104	18.50%	4.24%	
	min	10	10	462	4 125	0.6207	7.49%	0.49%	
DBLP	*median*	62	68	3 599	16 705	0.6814	9.28%	2.53%	72 548
	max	142	160	6 604	29 704	0.7104	11.51%	5.13%	

Minimum, median, and maximum value of each property is shown among all states in each dataset, where the number of constraints in each state is shown by $|\mathbf{C}_s|$; total number of performed actions in each state by $|\mathcal{A}|$; difference in two sets of constraints ($\mathbf{C}_{s-1}$ and $\mathbf{C}_s$) in terms of the number of edges added and removed covered by patterns in either set by $\Omega_{\mathbf{C}}$; overall changes in the dataset between two consecutive states ($s-1$ and s) in terms of number of edges added and removed by Ω_s; average of the average density of all the patterns in $\mathbf{C}_s$ by $\bar{\rho}$; compression ratio by CR; and coverage, i.e., the fraction of vertices of the dataset covered by all patterns combined. Runtime (in seconds) is the time required to process all states of the dataset, i.e., to obtain a complete solution of Problem 4.2.

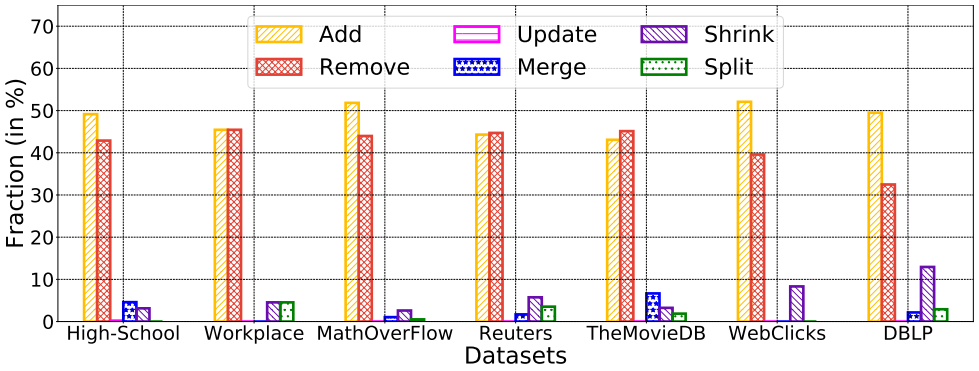


Fig. 4.3. The fraction of each type of action used to summarize each dataset.

state ranging between $2-7$. This is because WebClicks is sparsely dense with the number of unique edges $|E_s|$ almost equal to number of vertices $|V|$ (see Table 4.3). For TheMovieDB, a high number of patterns are observed in the summary of each state as TheMovieDB is relatively dense dataset.

We also observe that the patterns found covers a high number of vertices despite of performing only limited actions. The largest coverage of 58.41% is observed for High-School and smallest of 0.49% in DBLP. In case of WebClicks, reasonable coverage in the range of $2.19\% - 4.24\%$ is observed. Hence, we conclude that depending upon the size and density of a dataset, our method adequately identifies the number of patterns required summarize each state of a dynamic graph.

Number and type of actions performed. We observe that the number of actions ($|\mathcal{A}|$) performed in each state is consistent with number of changes taking place in the network upon evolution from one state to another (i.e., total number of new edges added and old edges removed, shown by Ω_S). That is, when Ω_S is smaller, a smaller value of $|\mathcal{A}|$ is observed, and vice versa. For example, in Reuters only 1 action is performed when changes in the network are small, i.e., a total of 322 edges are either added or removed, and 27 actions are performed when the changes are much larger, i.e., 13 494 edges either appeared or disappeared from the network. The fraction of each type of action performed can be seen in Figure 4.3. It is found that `add` and `remove` are the two most frequently performed actions, whereas the other types of actions depend on the nature of evolution of the network. It is seen that `update` is performed only for the High-School network.

For WebClicks, no `merge` or `split` actions are observed. Hence, the type of actions carried out are dependent on the topology of the network and the nature of evolution, to which our proposed algorithm effectively adapts itself.

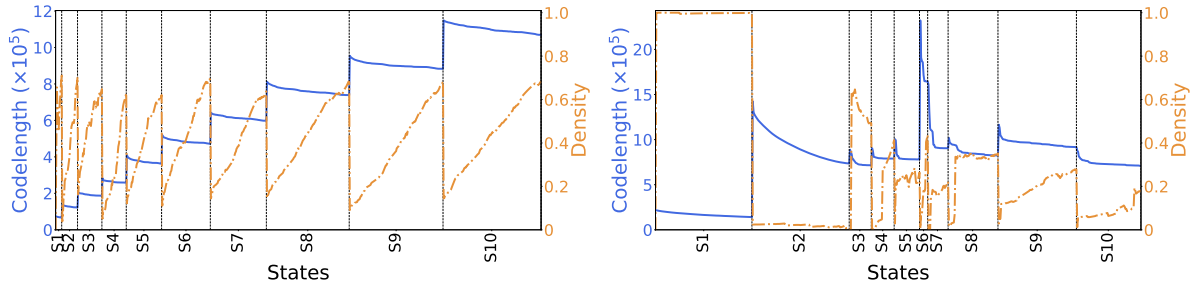
Quality of patterns. We assess the identified patterns through average density[4.11] ρ and compression ratios CR[4.12]. Minimizing the encoding cost of the data is only one part of our objective, and we use it to signify the information contained in the patterns: the higher the compression ratio, the more information about the data is provided by the patterns. The maximum compression ratio is observed for TheMovieDB, which is 48.30%, and the minimum of 0.16% is obtained for Reuters. This is accompanied by the observed high values for the average of the average densities of all identified patterns, including the minima of 0.0062 and 0.0070 in case of TheMovieDB and MathOverflow respectively, which are also higher than the average densities of snapshots of the data. Thus, our method finds subjectively dense and informative patterns.

We also observe for TheMovieDB where a more sophisticated belief, i.e., belief-i is used. That is, the background distribution closely represents a snapshot of the dataset, and with the change of state, any action would results in high compression ratios, which is also observed in Table 4.4.

Quality of actions performed. We next investigate the sequential approach taken in Problem 3. From the nature of the problem, it is expected that with each performed action, the codelength of the data should decrease and the average of average densities of identified set of patterns should increase. This is confirmed by Figure 4.4, where the codelength is found to be always decreasing and the density is mostly increasing for the DBLP and TheMovieDB networks. We also observe in Table 4.4, that there is a correlation between changes captured by the actions (i.e, Ω_C) and changes in the overall state (i.e., Ω_S) compared to the previous state. For TheMovieDB, we observed a relatively larger value of Ω_C, i.e., 15 556, when Ω_S is also large, i.e., 74 499; for Workplace we observed smaller value of Ω_C, i.e., 5, when Ω_S is smaller, i.e., 35. Therefore, the actions

[4.11] For a graph $G = (V, E)$, $\rho = \frac{|E|}{|V|*(|V|-1)}$(directed) or $= \frac{2*|E|}{|V|*(|V|-1)}$ (undirected)

[4.12] CR is 1 minus the ratio of the encoding cost (number of bits, computed as $-\log_2 P(\mathbf{D})$) of the data given the initial background distribution and given the final background distribution.



(a) DBLP (b) TheMovieDB

Fig. 4.4. Codelength (blue solid line) vs average of the average densities of patterns in set $\mathbf{C}_s$ (orange dashed dotted line) vs state s. The vertical dashed lines indicates the change of state and the horizontal axis represent from left to right all iterations, where a series of actions is performed for each consecutive state.

capture the changes in the graph state appropriately.

Runtime Analysis. Last, we discuss the (computation) time taken to run the experiment for each dataset. This is comprised of various factors, including the time required to compute the background distribution, executing the hill climber with different number of seeds to discover patterns, creating a candidate list for each type of atomic change to be performed, and updating the background distribution. In Table 4.4, the factors visibly affecting runtime are the size and density of a dataset, and the number of segments considered in a dataset. Overall, the maximum runtime of 79 049 seconds, which is approximately 22 hours for Reuters, appears practical. However, this could be further reduced upon optimization and parallelization of the proposed algorithm. Also note that all experiments have been run on a standard laptop.

4.5.4 Comparison with other methods

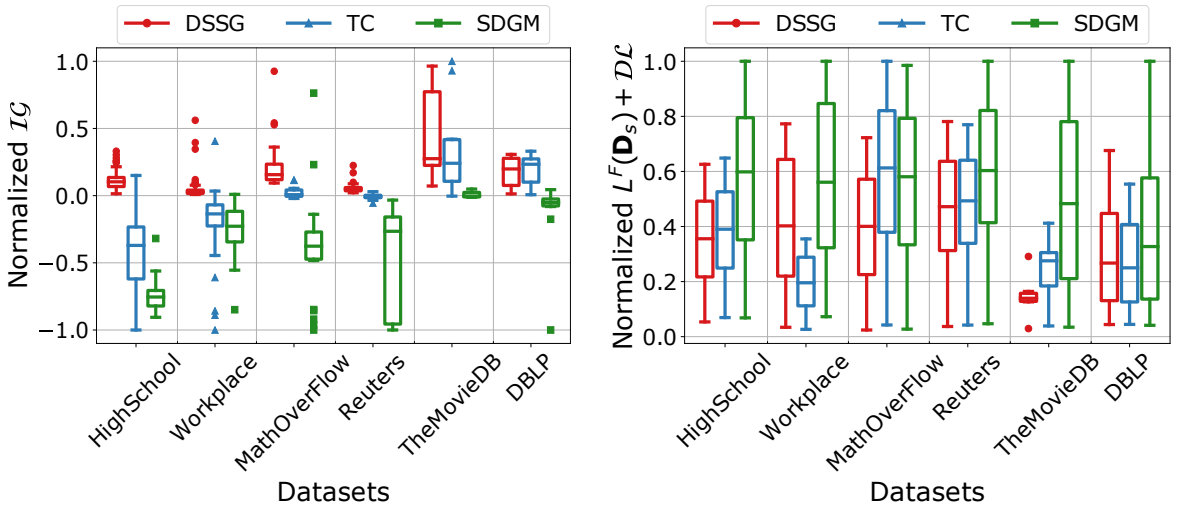
In this subsection, we compare the summaries generated by DSSG, TimeCrunch (TC) [Shah et al., 2015] and Scalable Dynamic Graph summarization Method (SDGM) [Tsalouchidou et al., 2020]. Note that, even though each of these methods summarizes a dynamic graph, they each solve a different problem when compared to DSSG. These methods summarize a dynamic graph but solve a different problem, unlike DSSG which summarizes a dynamic graph by discovering state-to-state relative changes by means of

evolving patterns and updating the analyst's knowledge after presentation of each graph snapshot. Since the other methods are different from DSSG, hence we present two baseline methods using TC and SDGM, aligned with DSSG to provide a fair comparison.

Baseline Methods. For TC, at each state s we compute two summaries using TC, which are, 1) for graph sequence $G_1, \ldots, G_{s-1}$, and 2) for graph sequence $G_1, \ldots, G_{s-1}, G_s$. The difference in the above two summaries is considered as the knowledge discovered in a state s. At each state s, the difference in the two summaries is communicated to the analyst using only two action types similar to the setting of DSSG, namely, `add` (to communicate the patterns that appeared in state s) and `remove` (to communicate the patterns which are present in state $s-1$ but not in state s). To encode an action we use the same strategy proposed for DSSG (see Table 4.2) such that the number of actions here is 2. Notably, SDGM also provides online summary, i.e., the summary is presented after each state. Thus, we use the same two action types and encoding strategy for SDGM to communicate the changes in the summary provided at each state when compared to the previous state. However, neither of the two methods, unlike DSSG, consider any type of analyst's prior knowledge. Thus, we assume that the goal of the other two methods is also to update the analyst's knowledge considering the same initial background distribution for each dataset as considered in case of DSSG. The background distribution is updated, after each performed action for each method, using the same principle as proposed for DSSG (Equations 4.5-4.7, 4.14). To prune the number of patterns/constraints to summarize the changes, DSSG performs actions with positive $\mathcal{IG}$ only. This is not applicable in TC, hence for a fair comparison we perform only a certain number of actions[4.13], such that the resulting number of patterns is equal to the number of patterns provided by DSSG in each state for each dataset. In SDGM[4.14], we choose the maximum number of resultant patterns among all states by DSSG as the number of supernodes in each dataset, as it identifies preset number of supernodes while producing a summary in each state.

[4.13] As recommended in Shah et al. [2015], we fix the Jaccard similarity threshold as 0.5.

[4.14] Tsalouchidou et al. [2020] suggests choosing a high number of microclusters compared to number of supernodes, therefore, the number of microclusters is chosen as 10 times the required number of supernodes/clusters. The required window size is selected as 2 to take into consideration only the previous state while summarizing each state.



(a) Normalized $\mathcal{IG}$

(b) Normalized $L^F(\mathbf{D}_s) + \mathcal{DL}$

Fig. 4.5. Plots showing the distribution of different measures over six datasets for each of the three method including DSSG, TC and SDGM. Each measure for each dataset is normalized by the maximum absolute value observed among all 3 methods.

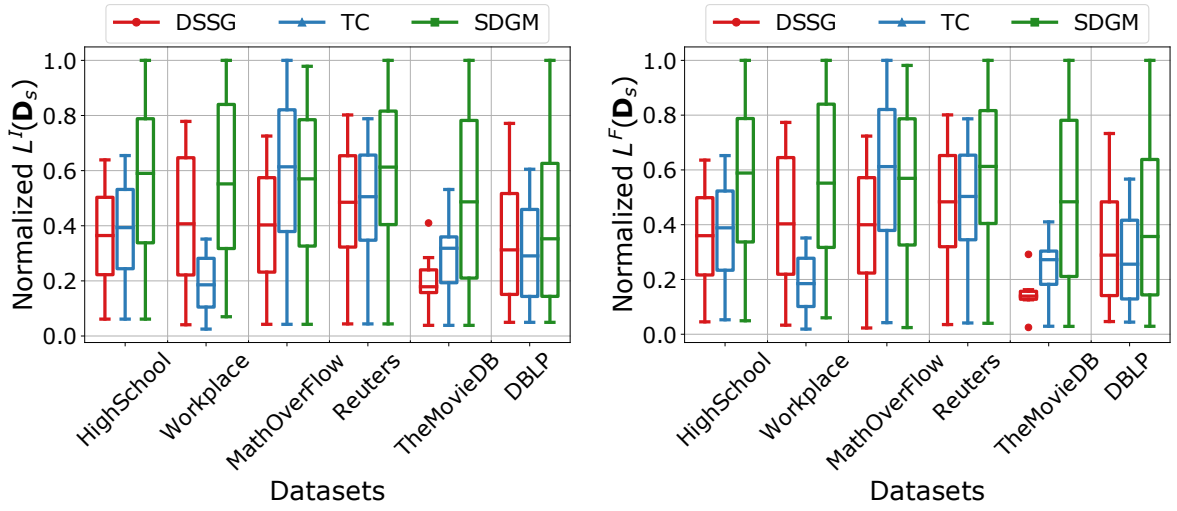
Comparison. The objective is to minimize $-\log P^*_{\mathbf{C}_s}(\mathbf{D}_s) + L(\mathbf{C}_s|\mathbf{C}_{s-1})$ to summarize each state s of a dynamic graph (see Problem 4.2). Hence, we evaluate each method using this as a measure. For simplicity, we denote the number of bits required to encode a graph snapshot $\mathbf{D}_s$ given the background distribution $P^*_{\mathbf{C}_s}$, i.e., $-\log P^*_{\mathbf{C}_s}(\mathbf{D}_s)$ by $L(\mathbf{D}_s)$, and the number of bits required to encode all the atomic changes, i.e., $L(\mathbf{C}_s|\mathbf{C}_{s-1})$ by $\mathcal{DL}$. At places, we use superscript I and F to represent the initial and final values of each state s respectively. The $L^F(\mathbf{D}_s) + \mathcal{DL}$ values of all states in each dataset for each method are presented as distribution (shown as box plots) in Figure 4.5b. It is observed in this figure that DSSG yields lower values across all states in four datasets except in WORKPLACE and DBLP, where TC is better than DSSG. To understand this, we compute the difference between the initial and the final values as the extent of decrease by each method. This decrease is equivalent to the total $\mathcal{IG}$ achieved by performing all actions in state s, and is represented as:

$$\mathcal{IG} = \left(\overbrace{-\log P_{C_s^I}(\mathbf{D}_s)}^{\mathbf{L}^I(\mathbf{D}_s)} + \overbrace{L(\mathbf{C}_s^I|\mathbf{C}_{s-1}^F)}^{=0,\text{ as } \mathbf{C}_s^I=\mathbf{C}_{s-1}^F} \right) - \left(\overbrace{-\log P_{C_s^F}(\mathbf{D}_s)}^{L^F(\mathbf{D}_s)} + L(\mathbf{C}_s^F|\mathbf{C}_{s-1}^F) \right)$$
$$= \underbrace{\log P_{C_s^F}(\mathbf{D}_s) - \log P_{C_s^I}(\mathbf{D}_s)}_{\mathcal{IC}} - \underbrace{L(\mathbf{C}_s^F|\mathbf{C}_{s-1}^F)}_{\mathcal{DL}}.$$

The similar distribution for $\mathcal{IG}$ (Figure 4.5a) indicates that for DSSG the value of $\mathcal{IG}$ is always positive in each dataset, unlike TC and SDGM, where it can be negative. This is in line with Problem 4.3 where DSSG performs actions with positive information gain only, where other methods do not explicitly solve a similar problem. It shall also be seen in case of WORKPLACE that $\mathcal{IG}$ is mostly negative for TC but is always positive for DSSG. Hence, it may not be seen as a fact that lower value of $L^F(\mathbf{D}_s) + \mathcal{DL}$ alone indicates effectiveness of a method. However, lower value of $L^F(\mathbf{D}_s) + \mathcal{DL}$ along with higher positive value of $\mathcal{IG}$ can be considered as parameter to evaluate the efficiency of a method. We also observe that in case of DBLP, TC is better than DSSG is both measures of $L^F(\mathbf{D}_s) + \mathcal{DL}$ and $\mathcal{IG}$. To further investigate, we analyse the components of $\mathcal{IG}$, namely, $L^I(\mathbf{D}_s)$, $L^F(\mathbf{D}_s)$, $\mathcal{IC}$ and $\mathcal{DL}$ in Figures 4.5c-4.5f respectively.

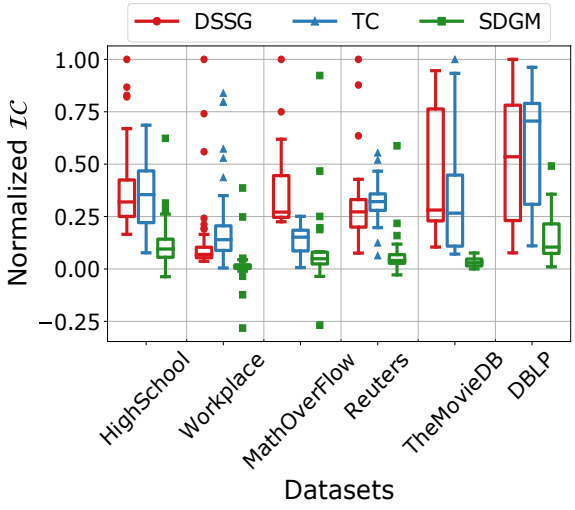
In Figure 4.5c, we do not observe similar distributions of $L^I(\mathbf{D}_s)$, which indicates that all methods start from different points implying that each method have in principle different measure of interestingness of a pattern. For any method, lower values of $L^F(\mathbf{D}_s)$ (Figure 4.5d) may indicate that the summary is more informative considering the analyst's prior knowledge. The observed trend in $L^F(\mathbf{D}_s)$ is similar to the trend observed in case of $L^F(\mathbf{D}_s) + \mathcal{DL}$ (Figure 4.5b) which suggest that lower value of $L^F(\mathbf{D}_s)$ is a major contributing factor. We observe lower values of $L^F(\mathbf{D}_s)$ in DBLP for TC among all three methods.

Although, higher values of $\mathcal{IC}$ are desirable, it is observed in only 2 datasets for DSSG, i.e., MATHOVERFLOW & THEMOVIEDB, and in rest 4 datasets for TC. The high values of $\mathcal{IC}$ for TC comes at a cost of higher values of $\mathcal{DL}$ as compared to DSSG (Figure 4.5f) which at many times lead to lower or even negative value of $\mathcal{IG}$. In Figure 4.5f, in most dataset except DBLP, least values of $\mathcal{DL}$ for DSSG are observed. However, in case of

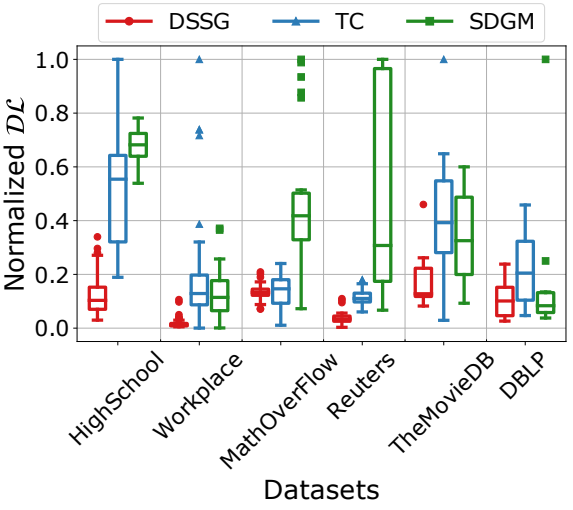


(c) Normalized $L^I(\mathbf{D}_s)$

(d) Normalized $L^F(\mathbf{D}_s)$



(e) Normalized $\mathcal{IC}$



(f) Normalized $\mathcal{DL}$

Fig. 4.5. (Continued) Plots showing the distribution of different measures over six datasets for each of the three method including DSSG, TC and SDGM. Each measure for each dataset is normalized by the maximum absolute value observed among all 3 methods.

DBLP, SDGM yields relatively lower values of $\mathcal{DL}$ and TC yields higher values of $\mathcal{DL}$ compared to DSSG. Moreover, the lower values of $\mathcal{DL}$ by SDGM in DBLP do not describe patterns with high information content $\mathcal{IC}$, in Figure 4.5e. It is also argued, that even though TC performs better in case of DBLP compared to DSSG, this, however, comes at a cost of discovering the patterns with higher value of $\mathcal{DL}$. The high value of $\mathcal{DL}$ indicates two possibilities, either the patterns found are less dense compared to DSSG, or have size either larger or smaller compared to DSSG and opposite to the expected size of pattern which is fixed using parameter 'q' in DSSG.

From the above observations, it may be concluded that DBLP is an exceptional case where TC performs better than DSSG. We also observe that even in the initial state, where the background distribution is the same for each DSSG and TC, TC finds patterns with higher $\mathcal{IG}$. This exception could be due to the limitation of the hill-climber, in DSSG, used to discover new patterns, is likely to suffer from the problem of local convergence. This limitation can be resolved by increasing the number of independent seed runs (here, fixed at 10 for all dataset) of the hill climber.

We highlight the observed key differences in the summaries obtained using DSSG compared to TC and SDGM in the following points:

1. The prior knowledge of the analyst including the knowledge obtained in the previous state is not considered in either TC or SDGM. We draw this inference from the observed lower values of $\mathcal{IG}$ and higher value of both $\mathcal{DL}$ and $L^F(\mathbf{D}_s)$ (hence $L^F(\mathbf{D}_s) + \mathcal{DL}$) by TC and SDGM compared to DSSG.

2. DSSG finds dense patterns of size as expected by the analyst, with much lower value of $\mathcal{DL}$.

3. Unlike TC and SDGM, in DSSG the number patterns to summarize a dynamic graph is not required to be explicitly fixed beforehand. Rather, DSSG by itself identifies the number of patterns sufficient to summarize a dynamic graph by only performing actions which yields positive information gain.

Besides the above-mentioned differences, neither TC nor SDGM captures the process of evolution of a dynamic graph in contrast to DSSG as discussed in the following subsection.

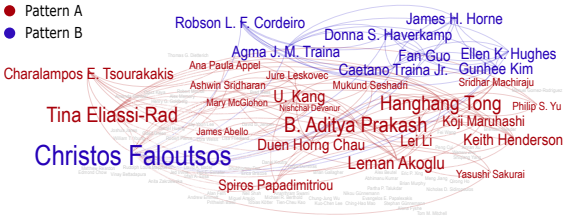
Thus, it can be seen that the summary provided by DSSG is different from the summaries provided by other methods such as TC and SDGM.

4.5.5 Qualitative Analysis

In this subsection, we discuss how the summary created by our proposed approach can be meaningful to a domain expert. Since we provide a summary of the changes in a dataset, the effectiveness of the discovered patterns can be assessed by the information captured in the sets of patterns and the actions performed on them. We analyze the results[4.15] obtained for DBLP, Reuters and TheMovieDB.

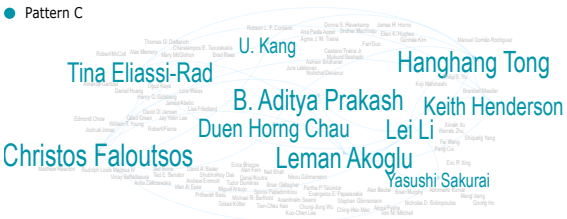
DBLP. For the DBLP graph, we discuss one of the various captured chains of subgraph patterns, which demonstrates the evolution of the communities of 92 authors centered mainly around Christos Faloutsos from Year 2010 to 2015. This evolution is shown in Figure 4.6. Initially (Figure 4.6a) in Year 2010, two surprisingly dense communities shown as pattern 'A' and 'B' are discovered, where Christos Faloutsos is a common link between the two communities. These two different communities have been condensed in the following year and merged to form a single community, shown as pattern 'C' (Figure 4.6b) with Christos Faloutsos and U Kang being some of the prominent names. This collaboration network shrinks the next year (Figure 4.6c). In Year 2013 another very densely connected set of authors is discovered, shown as pattern 'D' in Figure 4.6d. Surprisingly, in the subsequent year, this set of authors got split into two different communities of three authors each, i.e., Lisa Friedland, David D. Jensen & Amanda Gentzel and Christos Faloutsos, Jay Yoon Lee & Danai Koutra. However, the latter set of authors got merged with a newly discovered densely connected set of authors centered around Christos Faloutsos and Evangelos E. Papalexakis, shown by pattern 'H' in Figure 4.6e. Finally, in year 2015 the two different communities where Christos Faloutsos is the common link, i.e., pattern 'C' and a part of pattern 'H', merge to form one community with Neil Shah starting the collaboration with Leman Akoglu and others. In short, we captured how the community around one author with a large number of collaborations evolve over time.

[4.15] In the following figures, graphs are shown such that the text size of a vertex label is proportional to its degree. That is, if vertex degree is higher then text size is larger and vice versa.



(a) Y2010: Add ‘A’ & ‘B’.

(b) Y2011: Shrink ‘A’ & ‘B’; Merge ‘A’ & ‘B’ to ‘C’.



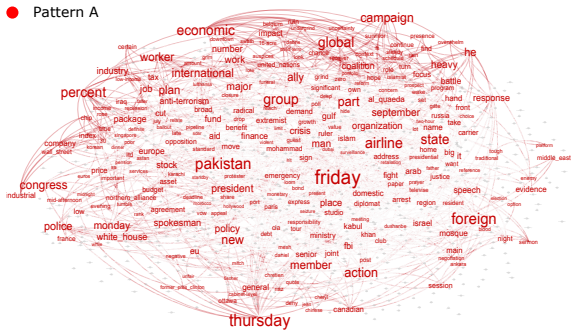
(c) Y2012: Shrink ‘C’.

(d) Y2013: Add ‘D’.

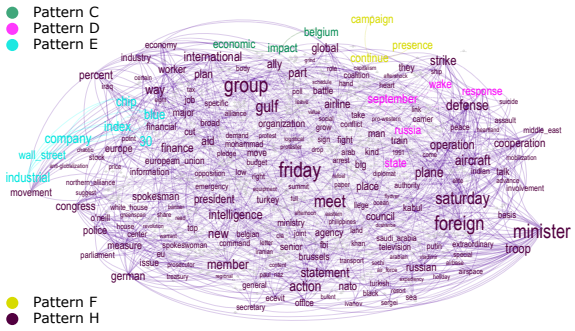
(e) Y2014: Split ‘D’ to ‘E’ & ‘F’; Add ‘G’; Merge ‘F’ & ‘G’ into ‘H’.

(f) Y2015: Split ‘H’ to ‘I’ & ‘J’; Remove ‘E’; Shrink ‘C’; Merge ‘J’ & ‘C’ into ‘K’.

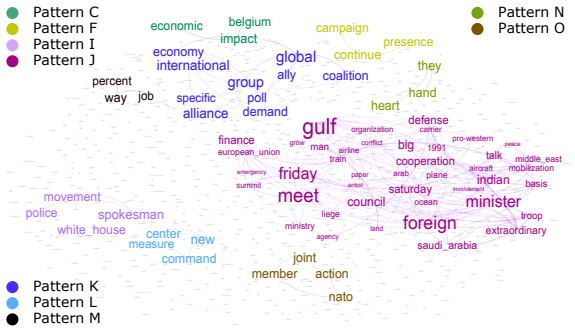
Fig. 4.6. Shown is the evolution of patterns in the DBLP network from Year 2010 to 2015.



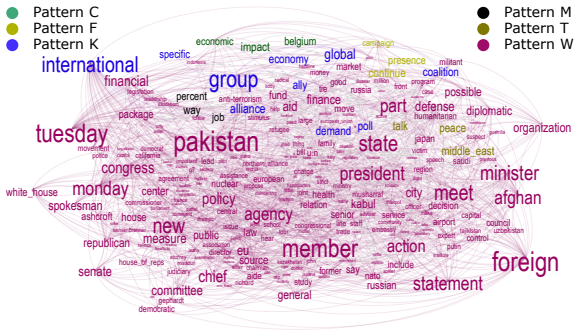
(a) D11: Add ‘A’.



(b) D12: Split ‘A’ into ‘B’, ‘C’, ‘D’, ‘E’, ‘F’; Add ‘G’; Merge ‘B’ & ‘G’ to ‘H’.

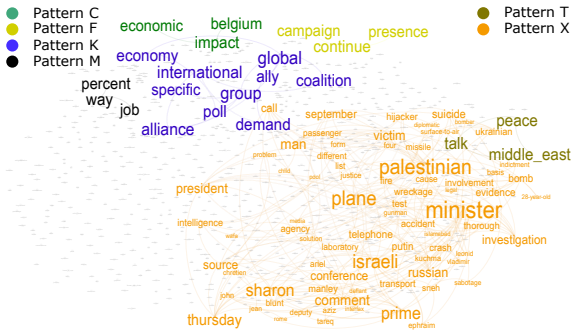


(c) D13: Split ‘H’ into ‘I’, ‘J’, ‘K’, ‘L’, ‘M’, ‘N’, ‘O’; Remove ‘D’ & ‘E’.



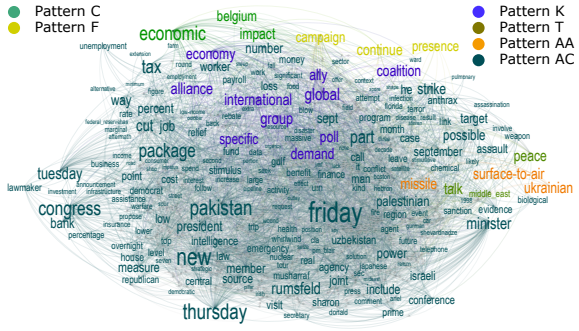
(d) D15: Add ‘P’; Split ‘J’ to ‘S’ & ‘T’; Remove ‘N’; Merge ‘I’, ‘L’, ‘P’, ‘O’, ‘S’ to ‘W’.

(e) D17: Remove ‘W’.

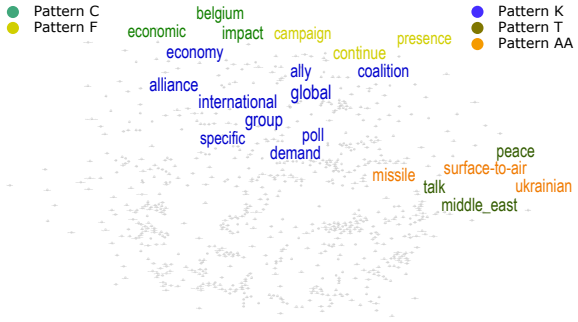


(f) D24: Add ‘X’.

Fig. 4.7. Shown is the evolution of pattern(s) in REUTERS network from Day 11 to Day 26.



(g) D25: Add 'Y'; Split 'X' to 'Z' & 'AA'; Merge 'M', 'Y', 'Z' to 'AC'.



(h) D26: Remove 'AC'.

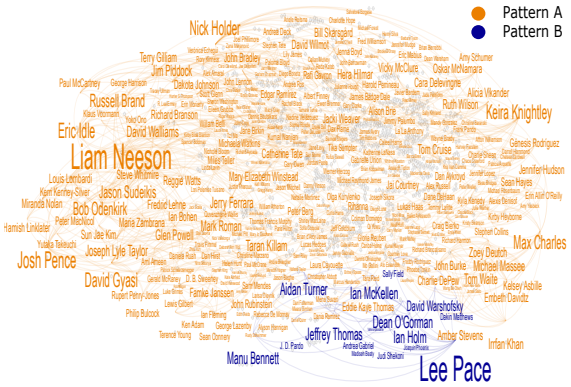
Fig. 4.7. (Continued) Shown is the evolution of pattern(s) in REUTERS network from Day 11 to Day 26.

Reuters. For the REUTERS network, we discuss the evolution of the set of 736 identified words in the dataset over a time span of 16 days, i.e., from day 11 to day 26, as shown in Figure 4.7. On day 11, a pattern ('A') of 269 words is discovered, where some of the prominent words include Friday, Airline, Global, Thursday, Foreign, Economic, Pakistan, State and Crisis (Figure 4.7a). Since the 11th day from the 9/11 attacks is Friday, it is hardly surprising to see Friday as a prominent word in this pattern. However, inclusion of words like Lufthansa, Jean, Unfair, etc. is interesting, as these words are not densely connected in the dataset. This pattern gets subsequently decomposed into five different patterns the following day, as shown in Figure 4.7b, followed by a merge with a newly discovered pattern (resulting pattern 'H'). Words such as Economic, Impact and Continue, which are densely connected in the original pattern, are no longer associated with any of the high degree words, such as Saturday and Minister in the current state. A further shrink of pattern 'H' is observed on the 13th day (Figure 4.7c), which is Sunday and can be explained by the fact that fewer articles are published on that day. On the 15th day, after a series of actions, a pattern 'W' emerges (Figure 4.7d). This pattern however vanishes on the 17th day (Figure 4.7e). Some of the noteworthy words in this pattern are Tuesday, Member, Foreign, Pakistan, Group, State, and President. Following this state, the next pattern 'X', is found on the 24th day (Figure 4.7f), which grew in size

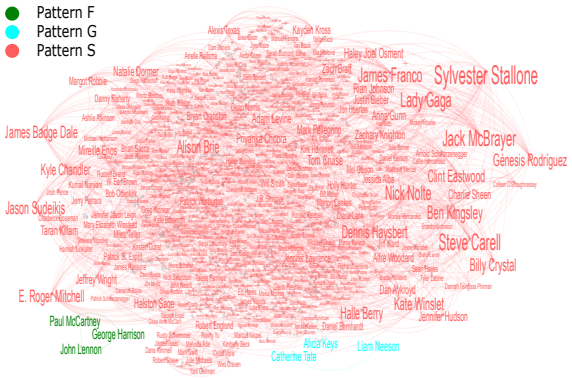
(via `merge`) resulting in a new pattern, 'AC', on the following day 25 (Figure 4.7g). This pattern existed only for a day and is removed on day 26 (Figure 4.7h). The notable words in pattern 'X' are Minister, Prime, Plane, Palestine and Israel; while in pattern 'AC' are Friday, Tax, Cut, Congress and Minister.

The presence of the day of the week as the most prominent word in a pattern each day is rather expected. However, the associations with other words are potentially interesting. One can also easily identify the most important topic of each day in the network from the discovered patterns, for e.g., Economics on day 11, Gulf on day 12 and 13, Palestine and Israel on day 24, and Tax Cut on day 25.

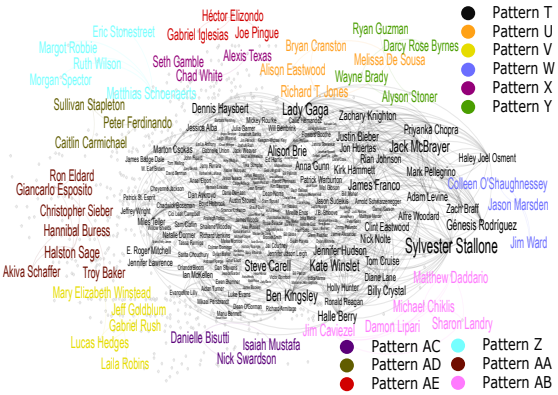
TheMovieDB. In this network, we discussed the discovered evolution of different patterns or communities of 1019 actors from Year 2012 to 2017, as shown in Figure 4.8. For each found pattern, we also find the associated genres using the hypergeometric test. A genre is considered to be significant if the p-value after Bonferroni Correction (with factor 19) is less than $1e-1$. During the Year 2012, two patterns 'A' and 'B' are discovered (Figure 4.8a). Pattern 'A', with significant genres Action and Comedy, includes vertices such as Liam Neeson, Josh Pence, David Gyasi and Nick Holder, all with high vertex degree. Pattern 'B' comprises of Sally Field and Lee Pace as high degree vertices and has Adventure and Fantasy as significant genres. In Year 2013, Pattern 'A' splits into 8 resulting patterns ('C', 'D', 'E', 'F', 'G', 'H', 'I', 'J'). This suggests that these 8 patterns represents 8 different communities of actors. Surprisingly, among these 8 patterns (which are all non-overlapping disjoint patterns), 6 patterns (excluding 'F' and 'G') got merged to form Pattern 'S' only after pattern 'K' is discovered (Figure 4.8b). Hence, it is found that the actors of the pattern 'C', 'D', 'E', 'H', 'I' and 'J' are indirectly connected through the actors in pattern 'K'. Some of the notable actors of pattern 'S' include James Badge Dale, Kyle Chandler, Kirsten Dunst and Will Smith. Pattern 'S' has Romance, Crime and Western as the three significantly associated genres. Pattern 'S' is decomposed into 12 different patterns in Year 2014 (Figure 4.8c). All the 12 resulting patterns have different significantly associated genres such as, Action with pattern 'T', Science Fiction with 'U', Documentary with 'W', Fantasy with 'X', Animation & Family with 'Y' and 'AE', War with 'Z' & 'AD', Crime with 'AA', Drama with 'AB' and Comedy with 'AC'. Most of the



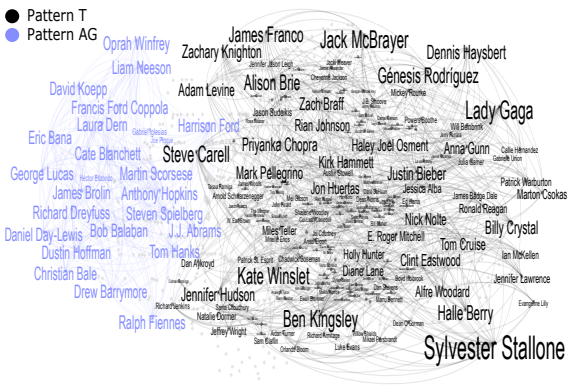
(a) Y2012: Add 'A' & 'B'.



(b) Y2013: Split 'A' into 8 patterns named in alphabetical order from 'C' to 'J'; Add 'K'; Merge all patterns 'B' to 'K' except 'F' and 'G' to form 'S' with 'L' to 'R' as intermediate patterns.



(c) Y2014: Split 'S' into 12 patterns shown in order from 'T' to 'AE'; Remove 'F' & 'G'.



(d) Y2017: Add 'AF'; Merge 'AE' & 'AF' to 'AG'.

Fig. 4.8. Shown is the evolution of patterns in TheMovieDB network from Year 2012 to 2017.

patterns disappear in the following two years, i.e., 2015 and 2016, except patterns 'T' and 'AE'. In Year 2017, pattern 'AE' merges with a newly discovered pattern 'AF', resulting in pattern 'AG'. Thus, pattern 'T' and 'AG' are observed in Year 2017 (Figure 4.8d). Some of the prominent actors in pattern 'T' are Sylvester Stallone, Lady Gaga, Ben Kingsley and Alison Brie. Pattern 'AG' includes actors like Dustin Hoffman and Oprah Winfrey and has Animation, Fantasy and Adventure as significant genres.

This case presents how the collaboration between actors evolves over time. The genres which are significantly associated to each pattern implies that our algorithm successfully identifies different and evolving subgroups (or communities) in the network.

4.6 Discussion

We propose a framework for summarizing sequential datasets in an online setting. We define *information gain* using both the maximum entropy principle and minimum description length principles. This measure enables not only to quantify the informativeness of a pattern, but also of the proposed actions (or atomic changes) in our framework, which enables to capture the evolution in a graph by evolving patterns. The proposed generic framework for subjective summarization of sequential data can be further instantiated for different types of evolving datasets, such as event sequence databases. In this paper, we instantiated the proposed generic framework for dynamic (simple) graphs.

This work focuses on the discovery of an *online* summary of dynamic graphs, by iteratively identifying actions with maximum information gain. The summary of a dynamic network contains a set of subgraph patterns (or constraints) along with captured changes in those (chains of) patterns over time. The findings from the experiments performed on different networks indicate that 1) the generated summaries are informative with regard to the analyst's prior knowledge about the data, with relatively high observed compression ratios; 2) the sets of subgraph patterns identified to summarize the networks are found to be relatively dense; and 3) the discovered evolving patterns provide an informative sequence that can be further inspected and analyzed. Also, with the proposed measures of information gain and information content, our method can be used to rank the found patterns.

We observe during the experiments that a pattern might appear regularly or sporadically in different snapshots of a dynamic network. This leads to a situation where our method learns and forgets the same pattern multiple times. However, on each occasion, our method treats the same pattern as newly acquired knowledge. It would be interesting to identify these instances while summarizing a network over time. A way to address this limitation could be to label each subgraph pattern and explore the similarity between two subgraph patterns. Thus, similar to TimeCrunch [Shah et al., 2015], the periodicity of a pattern could be explored. Another limitation of our work is the consideration of prior belief of the analyst. In this setting, we only consider that the analyst has prior knowledge on the initial snapshot and is interested in observing the changes in the network. A different setting may consider that the analyst knows about the different snapshots of the network.

One future opportunity includes improving the scalability of the proposed framework. The runtime of the proposed algorithm is currently higher than the two methods used to compare the summaries provided by DSSG, including TimeCrunch [Shah et al., 2015] and SDGM [Tsalouchidou et al., 2020]. Notably, the other two methods have a highly optimized implementation using parallel and distributed computing capabilities. For now, DSSG sequentially executes multiple procedures, including the number of independent seed runs of the hill climber. These procedures are highly independent and could be executed simultaneously. Hence, DSSG has several inherent features which may allow a parallelized implementation. This would significantly reduce the runtime and improve the scalability of the algorithm. Another future opportunity includes the development of a tool based on the proposed framework, for interactive visualization and exploration of changes identified in a dynamic network. This tool would further provide a user-friendly platform for analysts to learn how a network evolves with time.

4.7 Summary

We presented the novel problem of subjective summarization of sequential data in an online manner. As a specific instance of this generic problem, online summarization of dynamic graphs was introduced. We presented a framework to solve this problem, which

has been built on the existing ideas related to maximum entropy principle, the minimum description length principle, and subjectively interesting subgraph patterns. We then introduced an efficient algorithm, called DSSG, which is followed by extensive experiments on real-world datasets. Through experimental results, we demonstrated the effectiveness of the proposed algorithm. The generated summaries are found to be informative with regard to the analyst's prior knowledge about the data. We conclude this from the observed substantial compression ratios and the fact that *compression equates learning*. We have also found different sequences of patterns, which evolved over time in a network. As a part of future work, it would be interesting to extend the proposed method to incorporate a feature to capture periodicity of the patterns; another is to extend this method to multigraphs, weighted graphs, and attributed graphs. The proposed framework is tuned for evolving multigraphs in the following chapter. Finally, as a part of our ongoing/future work, we aim to develop a tool for interactive visualization and exploration of the found patterns. However, a brief idea of visualizing the generated summary is presented as a case study on airline network in Chapter 6.

Chapter **5**

Subjective Summarization of Evolving Multigraphs

Many real-world phenomena signify interaction (edges) between vertices which evolve and are best represented as dynamic graphs. At any instance or time interval, more than one edge may likely exist between two vertices — for example, two or more flights may be operational from one airport to another at any given time interval. In such case, upon segmentation of a dynamic graph into a sequence of graph snapshots, each snapshot is ideally represented by a *multigraph* — having parallel edges. This type of dynamic graphs is referred to here as *evolving multigraphs*.

Summarizing dynamic graphs has been a widely studied problem, but to the best of our knowledge, the problem of summarizing evolving multigraphs is not yet addressed explicitly. We address this gap in the literature through an approach for subjective summarization of evolving multigraphs, which summarize an evolving multigraph considering the analyst's knowledge. For this, we instantiate our previous proposed generic framework for subjective interestingness for sequential data. Here, we propose a new efficient measure to approximate the most interesting pattern in an evolving multigraph heuristically. We also propose a new encoding schema for evolving multigraphs to encode the atomic changes (or 'actions') suggested on the analyst's current knowledge at each step.

We refer to the algorithm proposed for evolving multigraphs as DSIMP. The structure of this algorithm is the same as the DSSG algorithm. We will also demonstrate its efficacy through experiments on some of the real-world cases of evolving multigraphs.

5.1 Introduction

Lately, different methods have been proposed in the literature to solve the problem of *dynamic graph summarization*. Most methods consider a dynamic graph as a sequence of simple graphs. In the previous chapter, we proposed a novel generic framework for *online summarization of sequential data using subjective interestingness*. This framework is then instantiated for dynamic graphs — presented as a sequence of simple graph snapshots. However, it has been observed that for any given instance or time interval, the corresponding graph snapshot may not necessarily be a simple graph and may contain parallel edges between any vertex pair. Following instances can realize this observation: two authors may co-author more than one publication (in a co-authorship network), two-actor may work together in more than one movie (in a co-actor network), and two flights may be scheduled from one airport to another (in an airline network). In turn, these instances evince that the graph snapshot is indeed a *multigraph*; and a sequence of multigraphs is referred to as an *evolving multigraph*.

In this chapter, we propose an instantiation of our generic framework of *subjective online summarization* for *evolving multigraphs*. However, the definitions proposed for *evolving simple graphs* cannot be directly adapted for *evolving multigraphs* due to the existence of parallel edges. Thus, a different encoding scheme is required to encode both a multigraph pattern and the atomic changes (or 'actions'). The same proxy measure used to discover subgraph patterns in evolving simple graphs cannot be directly used in evolving multigraphs either. Note that in Chapter 3 we used the measure of aggregate deviation to discover multigraphs patterns; however, we will show in this chapter that this measure cannot be adapted. Thus, we propose a new efficient measure for the same.

Thus, the contributions of this chapter are two-fold. First, we propose a novel measure which is used as a proxy to discover multigraph patterns. Second, we propose a new coherent encoding schema to encode actions in case of evolving multigraphs.

This chapter is organized as follows. The data and notation with the formal problem definition of subjective summarization of evolving multigraphs are presented in Section 5.2. We discuss the proposed approach is Section 5.3 followed by experiments in

Section 5.4. Finally, we conclude this chapter in Section 5.5.

5.2 Preliminaries

In this section, we introduce the evolving multigraph data and define the notation adopted in this chapter. We also briefly describe the problem of subjective online summarization.

5.2.1 Data and Notation

A *multigraph* is denoted as $G^M = (V, E^M)$, where V is a set of N vertices and E^M is a multiset of edges, where each edge $e \in E^M$ is an element of $V \times V$. Notably, there can be repetitive elements indicating parallel edges between two vertices in an edge multiset of a multigraph. An *evolving multigraph* is denoted as $G_T^M = (V, E_T^M)$, where E_T^M is the multiset of edges that occurs in time interval T. More precisely, each element $e = (u, v, t^b, t^f) \in E_T$ represents an edge between vertices u and v which exists between the time period represented by t^b (start time) and t^f (end time). There exist edges between the same pair of vertices, but with the same, different or overlapping time periods. Also, there can exist repetitive elements in E_T, indicating multiple edges between two vertices in the same time period.

An evolving multigraph can also be treated as *sequential data*, i.e., a sequence of snapshots, after segmenting the time interval T into multiple shorter time intervals (for example, seconds, minutes, hours and so on) of same or different length. For simplicity, we consider the segmentation of time into multiple shorter intervals of equal lengths. Thus, any edge that exists in a given time interval is considered to appear in the corresponding snapshot. Hence, we project an evolving multigraph into a sequence of different multigraph snapshots $G_1, G_2, \ldots, G_{\mathcal{S}}$, where $\mathcal{S}$ is the total number of snapshots upon segmentation. The corresponding adjacency matrix of a multigraph snapshot G_s is represented by $\mathbf{D_s} \in \mathbb{D}^{N \times N}$, where $\mathbb{D} = \mathbb{N}_0$.

5.2.2 Subjective Online Summarization of Evolving Multigraphs

As proposed earlier in Chapter 4 (Problem 4.2), we propose a similar problem of subjective summarization of evolving multigraphs. The problem is formally presented as follows.

Problem 5.1 (Subjective Summarization of Evolving Multigraphs) *Given an*

evolving multigraph G_T^M *consisting of a sequence of snapshots* $G_1, \ldots, G_{\mathcal{S}}$ *, with* $\mathbf{D}_s$ *the corresponding adjacency matrix for a state* s*, and prior beliefs* $\mathcal{B}$*, find:*

- *for* $\mathbf{D}_1$*: a set of constraints* $\mathbf{C}_1$ *that minimizes* $-\log P_1^*(\mathbf{D}_1) + L(\mathbf{C}_1)$*, where* P_1^* *is computed using constraints* $\mathcal{B} \cup \mathbf{C}_1$*;*
- *for* $\mathbf{D}_s$*, with* $s \in \{2, \ldots, S\}$*: a set of constraints* $\mathbf{C}_s$ *that minimizes* $-\log P_s^*(\mathbf{D}_s)$ $+L(\mathbf{C}_s|\mathbf{C}_{s-1})$*, where* P_s^* *is computed using constraints* $\mathcal{B} \cup \mathbf{C}_s$

such that each pattern in any set $\mathbf{C}_s$ *is a connected multigraph pattern.*

Similar to Chapter 4, we propose to solve Problem 5.1 by iteratively solving Problem 5.2, which is presented as follows.

Problem 5.2 (Online Summarization of Evolving Multiraphs) Given the current state s, multigraph snapshot G_s, corresponding adjacency matrix $\mathbf{D}_s$, current constraint set $\mathbf{C}_s$, and background distribution $P_{\mathbf{C}_s}^*$ perform that action 'α' from the set of all possible actions having maximal information gain $\mathcal{IG}$, given by

$$\mathcal{IG}(\alpha) = \mathcal{IC}(\alpha) - \mathcal{DL}(\alpha),$$

such that the pattern(s) obtained after performing 'α' are connected (sub-) multigraph(s).

We present the actions on constraints set $\mathbf{C}_s$ along with their encoding scheme in the following sections.

5.3 Proposed Approach

In this section, we present a new measure to discover multigraph patterns efficiently and then present the possible actions to update the analyst's knowledge of the data with the proposed encoding scheme.

5.3.1 Self Information of a Multigraph Pattern

In Problem 5.2, we propose to perform an action having maximal information gain, which is the difference of information content and description length. Notably, when an action is proposed the information content is a non-negative quantity representing that the

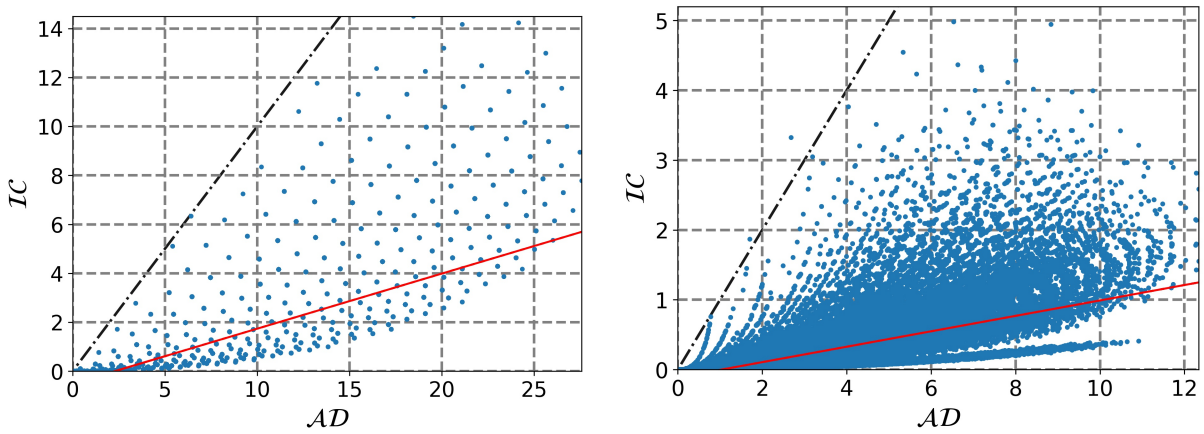
codelength required to encode the data reduces (see Definition 4.1). The action with maximal information gain ensures that the codelength required to encode the data is reduced maximally in a minimal complexity.

As seen in the previous chapter to identify the best action with maximum information gain to perform, `add` is the most computationally expensive task. The search for the most interesting pattern in the worst case requires evaluating a total of $2^{|V|}$ possible candidate patterns. In case of a static multigraph, we observed the same in Section 3.5. Thus, an efficient way is to use a heuristic search method such as the hill climber proposed in Algorithm 3.1. Further, in Section 4.4, we realized that in hill climber heuristic search, naïvely computing $\mathcal{IG}$ at each step is also cost-inefficient. Given this, the use of proxy measure has been proposed in place of $\mathcal{IC}$, which requires practical computation complexity. However, the proxy proposed for evolving simple graph data cannot be used in evolving multigraph data. The reason has been earlier discussed and highlighted in Chapter 3, i.e., due to different structural properties of a multigraph, the definitions proposed for simple graphs are not applicable.

The above challenge can be remedied by the use of the measure of aggregate deviation proposed for static multigraphs in Chapter 3, Definition 3.1, as a potential alternative. In Section 4.4, for simple graphs, it has been suggested that the self-information proposed in van Leeuwen et al. [2016] can be used as a proxy measure as it is positively correlated to the $\mathcal{IC}$.

Although the measure of aggregate deviation $\mathcal{AD}$ is efficient in defining a multigraph pattern's usefulness, we observed that it might not be a good proxy measure for $\mathcal{IC}$. In Figure 5.1, we observed that $\mathcal{IC}$ and $\mathcal{AD}$ are somewhat correlated but $\mathcal{AD}$ may not considered as a suitable proxy measure of $\mathcal{IC}$. Hence, we next explore another suitable measure to quantify the interestingness of a multigraph pattern. Since the previously suggested proxy measure for simple graphs is a self-information based measure, it is evident that a self-information based measure for multigraphs can be an effective proxy measure.

We know that the maximum entropy distribution for a multigraph is a product of independent geometric distributions. A pattern can be considered informative if it has



(a) Belief-c, $r = 0.8732$

(b) Belief-i, $r = 0.7487$

Fig. 5.1. Shown are the plots of $\mathcal{AD}$ vs $\mathcal{IC}$ for different type of prior beliefs. In each plot, $\mathcal{AD} = \mathcal{IC}$ line is shown (dash-dotted) with a linear fit curve (red solid dashed). The value of Pearson's correlation coefficients r are mentioned in the subcaptions. We used a synthetic multigraph of 20 vertices, which is synthesized using the method given in Section 3.6, such that $p_b = 0.8$, $p_g = 0.4$ & $l = 10$. $\mathcal{AD}$ and $\mathcal{IC}$ is computed for all subgraphs such that $\mathcal{AD} > 0$.

at least k edges. For a multigraph pattern θ^m having W vertices and k^m_W edges, we consider the probability of having a sum of k^m_W edges between n^s_W unique vertex pairs. The expected number of edges between each vertex pair is represented by an independent geometric distribution with probability of success $p_{u,v}$. This fact implies that for n^s_W random variables, we are interested in having at least k^m_W combined failures before having a first success in each case. In much simpler words, for a multigraph pattern, this is the case of the probability of having k^m_W edges. Notably, n^s_W is also the maximum possible number of edges in an equivalent simple graph of the same number of vertices.

Thus, we compute the probability using the upper bound for the upper tail probability for a sum of multiple independent geometric random variables, possibly with different parameters. The resulting theorem is as follows.

Theorem 5.1 *Let $X_1, X_2, \ldots, X_n$ be n independent geometric random variables with possibly different distributions: $X_i \sim \mathbf{Ge}(p_i)$ with $0 < p_i \leq 1$, where $\mathbb{E}[X_i] = \frac{1-p_i}{p_i}$. Further-*

more, let $X = \sum_{i=1}^{n} X_i$ *and* $\mu = \sum_{i=1}^{n} \mathbb{E}[X_i]$*, then for any* $\lambda \geq 1$*:*

$$\mathbf{Pr}(X \geq \lambda\mu) \leq \exp\left(-\mu p_*(\lambda - 1) - p_*(\mu + n)\ln\left(\frac{\mu + n}{\lambda\mu + n}\right)\right),$$

where, $p_* = \min_i p_i$*.*

For brevity, the proof of Theorem 5.1 is relegated to Appendix B.1.

The above theorem can be translated for the case of a multigraph pattern, such that n is now n_W^s, $\mu = \sum_{u,v \in W, u<v} \frac{1-p_{u,v}}{p_{u,v}}$ is the sum of expected number of edges between n_W^s vertex pairs, and $\lambda\mu = k_W^m$. Thus, we get the following result.

$$\mathbf{Pr}([W, k_W^m]) \leq \exp\left(-p_*(k_W^m - \mu) - p_*(\mu + n_W^s)\ln\left(\frac{\mu + n_W^s}{k_W^m + n_W^s}\right)\right),$$

where, $p_* = \min_{u,v \in W, u<v} p_{u,v}$.

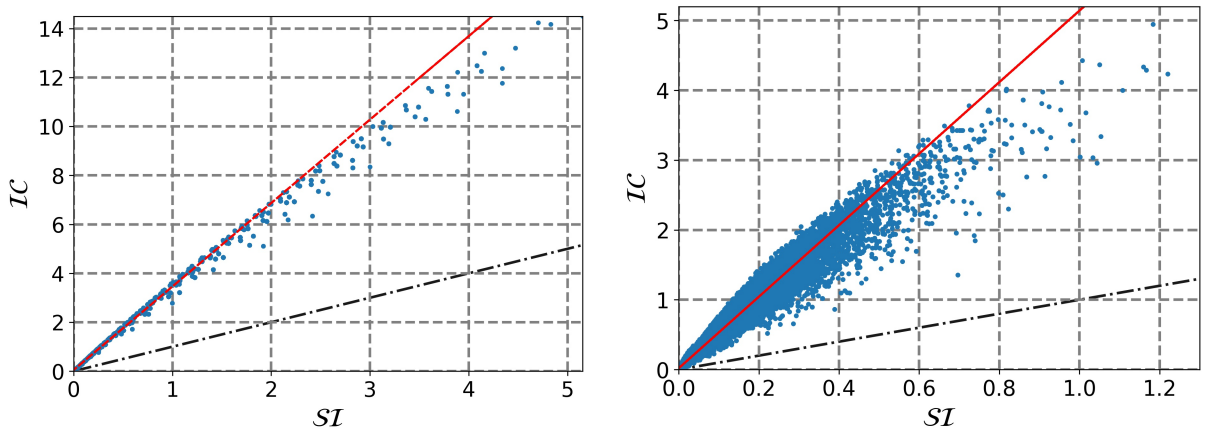
Since the self information of a pattern is defined as the negative logarithm of the probability [De Bie, 2011b], for a multigraph pattern the same can be defined:

$$\begin{aligned} \mathcal{SI}(\theta^m) &= -\log\left(\mathbf{Pr}([W, k_W])\right) \\ &\geq p_*(k_W^m - \mu) + p_*(\mu + n_W^s)\ln\left(\frac{\mu + n_W^s}{k_W^m + n_W^s}\right) \end{aligned} \tag{5.1}$$

In Figure 5.2, for the same synthetic multigraph, we observe that $\mathcal{SI}$ is highly correlated with $\mathcal{IC}$, having a correlation of $r = 0.9988$ for prior belief-c and $r = 0.9848$ for prior belief-i. Therefore, $\mathcal{SI}$ is a suitable proxy measure for $\mathcal{IC}$. It is of note that $\mathcal{SI}$ is always less than $\mathcal{IC}$, but both are increasing functions.

Computationally, $\mathcal{SI}$ requires a worst-case complexity of $\mathcal{O}(|W|^2)$, which is still inefficient. However, in a heuristic framework, all the terms in Equation 5.1 can be efficiently computed by maintaining a list of potential vertices that can be added to a subgraph in each step. In the list of potential vertices, the potential change—on adding a vertex—in the values of k_W^m, μ, n_W^s and q can be easily maintained. The update of this list requires—in a worst-case—a linear complexity of $\mathcal{O}(|V|)$. Hence, the computation of $\mathcal{SI}$ is cost-effective compared to $\mathcal{IC}$.

Thus, for the evolving multigraph setting, we propose to use $\mathcal{SI}$ as a proxy measure to only discover a multigraph pattern, i.e., executing the proposed hill climber based search



(a) Belief-c, $r = 0.9988$ (b) Belief-i, $r = 0.9848$

Fig. 5.2. Shown are the plots of $\mathcal{SI}$ vs $\mathcal{IC}$ for different type of prior beliefs. In each plot, $\mathcal{SI} = \mathcal{IC}$ line is shown (dash-dotted) with a linear fit curve (red solid dashed). The value of Pearson's correlation coefficient is mentioned in the subcaption. We used a synthetic multigraph of 20 vertices, which is synthesized using the method given in Section 3.6, such that $p_b = 0.8$, $p_g = 0.4$ & $l = 10$. $\mathcal{SI}$ and $\mathcal{IC}$ is computed for all subgraphs such that $\mathcal{AD} > 0$.

algorithm. On the contrary, to decide which action to be performed, we recommend computing the actual $\mathcal{IC}$ value, as in Figure 5.2 we observed that $\mathcal{SI} << \mathcal{IC}$. This will ensure that each action is given a fair chance to compete.

5.3.2 Analysis of Aggregate Deviation versus Self-Information

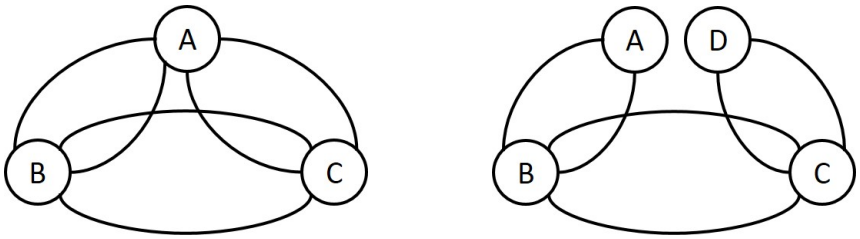
As evident in Section 5.3.1, aggregate deviation and self-information are two different measures to define interestingness of a multigraph pattern. The aggregate deviation is the difference between the actual number of edges and the expected number of edges in a multigraph pattern. On the other hand, self-information is the negative logarithm of the probability of a pattern, and is given as

$$\begin{aligned} \mathcal{SI}(\theta^m) &= -\log\left(\mathbf{Pr}([W, k_W])\right) \\ &\geq p_*(k_W^m - \mu) + p_*(\mu + n_W^s) \ln\left(\frac{\mu + n_W^s}{k_W^m + n_W^s}\right). \end{aligned} \tag{5.2}$$

Upon close inspection, it is realized that the aggregate deviation is itself a part of the above equation. That is, the above equation can be re-written as

$$\mathcal{SI}(\theta^m) = p_* \times \mathcal{AD} + p_*(\mu + n_W^s) \ln\left(\frac{\mu + n_W^s}{k_W^m + n_W^s}\right). \tag{5.3}$$

Since we consider relatively dense patterns as interesting, i.e., k_W is greater than μ. Hence, the second term in the above equation is a negative quantity. It shall also be noted that p_* is always less than or equal to 1. Therefore, it can be concluded from the above equation, that $\mathcal{SI} \leq \mathcal{AD}$ (also observed in Figures 5.1 and 5.2).



(a) $|W| = 3,\ k_W^m = 6,\ n_W^s = 3$ (b) $|W| = 4,\ k_W^m = 6,\ n_W^s = 6$

Fig. 5.3. Toy example of multigraph patterns.

Let us consider an example of two different multigraph patterns, as shown in Figure 5.3. Upon considering a hypothetical situation, where μ and k_W^m are the same for both cases, then the usefulness of either example cannot be determined with $\mathcal{AD}$; not considering the description length. Interestingly, evaluation of the $\mathcal{SI}$ value for both cases may distinguish the usefulness of a pattern to the analyst. This is because n_W^s is greater for the case shown in Figure 5.3b than for the case in Figure 5.3a. Also, the second term in Equation 5.3 is strictly decreasing for $n_W^s > 0$, when μ and k_W^m are constant such that $k_W^m \geq \mu$. Thus, $\mathcal{SI}$ for the pattern in Figure 5.3a would be larger than $\mathcal{SI}$ for the pattern in Figure 5.3b. Thus, we can anticipate discovering denser patterns with the measure of $\mathcal{SI}$ compared to the measure of $\mathcal{AD}$.

In another case, we consider two patterns, θ_1 and θ_2 having the same number of vertices but different edges. In a scenario where $\mathcal{AD}$ is equal for both patterns, then $\mathcal{AD}$ cannot decide between either pattern, but $\mathcal{SI}$ can. The reason for this is the different values of p_* and μ for both patterns. If k_W^m is greater for θ_1 than θ_2, then for $\mathcal{AD}$ to be equal, μ for θ_1 has to be greater than θ_2. Also, to accommodate a larger μ, p_* has to be smaller.

Thus, we can say that $\mathcal{SI}$ for θ_1 will be less than θ_2. Therefore, as per $\mathcal{SI}$, a pattern is more informative, if fewer edges are expected, given that $\mathcal{AD}$ is the same.

Therefore, it can be inferred that $\mathcal{SI}$ is a better measure than $\mathcal{AD}$ to define subjectively dense multigraph patterns. Further, we confirm our analysis with the experiments on various static multigraph datasets in the latter part of this chapter.

5.3.3 Actions and their Encoding Strategy

Since the problem of online summarization for the evolving multigraphs is the same as for the evolving simple graph, the same type of actions defined for simple graphs can be used in case of multigraphs. However, when the number of edges is also described, the maximum number of possible edges in a multigraph are unknown beforehand. Thus, a multigraph pattern cannot be described using the same complexity as required in simple graphs. We now discuss an optimal encoding schema for a static multigraph pattern and for the actions to summarize an evolving multigraph.

A simple graph pattern is described by encoding the vertex set and the number of edges in a subgraph. A multigraph is described similarly, but with a difference in the strategy to encode the number of edges. In a simple graph with W vertices, the number of edges k^s_W is minimally described by encoding the number of edges short in a subgraph compared to a clique of the same number of vertices. That is, the maximum possible number of edges n^s_W in a simple graph is known; thus, we encode the term $n^s_W - k^s_W$. This method allows us to describe the number of edges in a subgraph, which is surprisingly dense, using minimal complexity.

The same strategy cannot be used in case of multigraphs having W set of vertices as the maximum possible number of edges n^m_W in a multigraph cannot be defined. To overcome, we realized that the number of edges k^m_W in a multigraph pattern is always greater than or equal to k^s_W (the number of edges in a simple graph equivalent). In other words, the relationship between k^m_W and k^s_W is given as

$$k^m_W = \sigma \times k^s_W, \tag{5.4}$$

where σ is a non-negative real constant. With this relation, we can encode k^m_W using

Table 5.1. Shown are the formulation of Description Length ($\mathcal{DL}$) for each defined atomic change, α or action for the case of evolving multigraphs.

α	$\mathcal{DL}$
`Add`	$\mathcal{T}_\alpha + \mathcal{T}_{k_W^m} + \mathcal{T}_W$
`Remove`	$\mathcal{T}_\alpha + \mathcal{T}_C$
`Update`	$\mathcal{T}_\alpha + \mathcal{T}_C + \mathcal{T}_{k_{W'}^m}$
`Shrink`	$\mathcal{T}_\alpha + \mathcal{T}_C + \mathcal{T}_{k_{W'}^m} + \mathcal{T}_r + \mathcal{T}_{rw}$
`Merge`	$\mathcal{T}_\alpha + 2 \times \mathcal{T}_C + \mathcal{T}_{k_{W'}^m}$
`Split`	$\mathcal{T}_\alpha + \mathcal{T}_C + \mathcal{T}_M + \mathcal{T}_{\lvert W_\tau \rvert} + \mathcal{T}_{W_\tau} + \mathcal{T}_{k_{W_\tau}^m}$

$\mathcal{T}_\alpha = \log(l), \quad \mathcal{T}_C = \log(|\mathbf{C}|), \quad \mathcal{T}_{k_W^m} = \mathrm{L}_\mathbb{N}\left(n_W^s - k_W^s + 1\right) + \log(\tau_W) - \log(\delta),$
$\mathcal{T}_{k_{W'}^m} = \mathrm{L}_\mathbb{N}(n_{W'}^s - k_{W'}^s + 1) + \log(\tau_{W'}) - \log(\delta), \quad \mathcal{T}_r = \mathrm{L}_\mathbb{N}(|\Psi|)$
—where Ψ is the number of vertices removed,
$\mathcal{T}_{rw} = \log(|W|) + \log(|W|-1) \cdots + \log(|W| - |\Psi| + 1), \quad \mathcal{T}_\tau = \mathrm{L}_\mathbb{N}(\tau), \quad \mathcal{T}_{|W_\tau|} = \sum_{i=1}^{\tau} \mathrm{L}_\mathbb{N}(|W_i|),$
$\mathcal{T}_{W_\tau} = \log(|W|) + \log(|W|-1) + \cdots + \log(|W| - x + 1)$ — where $x = |\bigcup_{i=1}^{\tau} W_i|,$
$\mathcal{T}_{k_{W_\tau}^m} = \sum_{i=1}^{\tau} \mathrm{L}_\mathbb{N}(n_{W_i}^s - k_{W_i}^s + 1) + \log(\tau_i) - \log(\delta), \quad \mathcal{T}_W = |W| \log(q) + (|V| - |W|) \log(1-q)$

minimal codelength by encoding k_W^s and σ separately. Here, k_W^s is an integer number which can be minimally encoded using codelength $\mathrm{L}_\mathbb{N}(n_W^s - k_W^s + 1)$ [Rissanen, 1983], and σ is a positive real number which can be minimally encoded in $\log(\sigma) - \log(\delta)$ bits [Lee, 2001]. Note that, in a simple graph equivalent of a multigraph n_W^s is known, and to encode σ, δ is the given precision such that $|\sigma - \sigma_\delta| < \delta$, where σ_δ is a truncated number. The only limitation of using this method is that some accuracy is inevitably lost.

The method proposed in Chapter 4, to encode the set of vertices in a pattern and the information required to encode an action, can also be used for the case of evolving multigraphs. Hence, most encoding strategies to encode a sub-component in each action remain as given in Chapter 4 - Table 4.2. The only encoding strategy that is unique for multigraphs is the encoding of the number of edges.

The encoding strategies proposed for each type of action in case of evolving multigraphs are summarized in Table 5.1. We encode the type of an action to be performed in $\mathcal{T}_\alpha$ bits, which is required for each type of action. The information to identify the constraint on

which an action is performed is required in case of all actions except `add`, and is encoded using $\mathcal{T}_C$ bits. The set of vertices in a pattern for `add` action is encoded using $\mathcal{T}_W$ bits, the vertices removed in `shrink` action using $\mathcal{T}_r + \mathcal{T}_{rw}$ bits, and vertices in each component after `split` using $\mathcal{T}_{|W_\tau|} + \mathcal{T}_{W_\tau}$ bits. The number of edges in a resulting pattern(s) is encoded in $\mathcal{T}_{k^m_W}$ and $\mathcal{T}_{k^m_{W_\tau}}$ bits for `add` and `split` actions, respectively. While for actions such as `update`, `shrink` and `merge`, the number of edges in the final pattern is encoded using $\mathcal{T}_{k^m_{W'}}$ bits. Finally, in `split` the information about the number of resulting new patterns is also encoded using $\mathcal{T}_\tau$ bits, where τ is the number of resulting patterns.

5.3.4 The Algorithm

The problems of online summarization of evolving multigraph and evolving simple graph are almost the same. Hence, the DSSG algorithm proposed in the previous chapter, Section 4.4, can be used for evolving multigraphs. However, to discover a new pattern or constraint, the new proposed heuristic measure in Section 5.3.1 will be used. For clarity, in the case of evolving multigraph, we will refer to this algorithm as DSIMP.

It shall be noted, since the two algorithms —DSSG and DSIMP— are the same; hence, the overall runtime complexity also remains the same.

5.4 Experiments

In this section, we will demonstrate the efficacy of the proposed framework of online summarization for evolving multigraphs and the proposed algorithm DSIMP. We will also investigate the difference between the two measures of interestingness for a multigraph pattern, i.e., one based on aggregate deviation and the other based on self-information. For this, we incorporate the definition of self-information for multigraphs in SIMP framework.

5.4.1 Results on Static Multigraphs

In this section, we analyse the patterns observed using the $\mathcal{SI}$ measure with the SIMP algorithm, proposed in Chapter 3. Hence, we use the same real-world examples, as given in Section 3.6. For the experiments, we use interest-based seeding strategy with 10 independent seed runs and q=0.01.

In Table 5.2, the properties of the most interesting pattern discovered by the SIMP

Table 5.2. Shown are the properties* of the most interesting multigraph pattern discovered using different configurations of SIMP on each dataset.

DS	Configuration	$\|V\|$	k_W^m	k_W^s	ρ^m	ρ^s	η^m	**d**	γ	**best**
DBLP1	$\mathcal{SI}$-c	14	501	91	**5.51**	**1.00**	**35.79**	**1.00**	**29.29**	*
	$\mathcal{SI}$-i	14	501	91	**5.51**	**1.00**	**35.79**	**1.00**	**29.29**	*
	$\mathcal{AD}$-c	15	524	105	4.99	**1.00**	34.93	**1.00**	27.93	
	$\mathcal{AD}$-i	15	524	105	4.99	**1.00**	34.93	**1.00**	27.93	
	$\mathcal{AD}$-m	18	406	125	2.65	0.82	22.56	**1.00**	15.61	
DBLP2	$\mathcal{SI}$-c	16	226	120	1.88	**1.00**	14.13	**1.00**	6.63	
	$\mathcal{SI}$-i	9	142	27	**3.94**	0.75	**15.78**	2.00	**12.78**	*
	$\mathcal{AD}$-c	30	448	435	1.03	**1.00**	14.93	**1.00**	0.43	
	$\mathcal{AD}$-i	30	448	435	1.03	**1.00**	14.93	**1.00**	0.43	
	$\mathcal{AD}$-m	30	448	435	1.03	**1.00**	14.93	**1.00**	0.43	
DBLP3	$\mathcal{SI}$-c	137	13916	9279	1.49	**1.00**	101.58	**1.00**	33.85	
	$\mathcal{SI}$-i	102	10034	5151	**1.95**	**1.00**	98.37	**1.00**	**47.87**	*
	$\mathcal{AD}$-c	140	14626	9692	1.50	**1.00**	**104.47**	2.00	35.24	
	$\mathcal{AD}$-i	142	14780	9843	1.48	0.98	104.08	2.00	34.77	
	$\mathcal{AD}$-m	140	14626	9692	1.50	**1.00**	**104.47**	2.00	35.24	
DBLP4	$\mathcal{SI}$-c	55	1495	1485	**1.01**	**1.00**	**27.18**	**1.00**	**0.18**	*
	$\mathcal{SI}$-i	55	1495	1485	**1.01**	**1.00**	**27.18**	**1.00**	**0.18**	*
	$\mathcal{AD}$-c	55	1495	1485	**1.01**	**1.00**	**27.18**	**1.00**	**0.18**	*
	$\mathcal{AD}$-i	71	1663	1653	0.67	0.67	23.42	2.00	0.14	
	$\mathcal{AD}$-m	71	1663	1653	0.67	0.67	23.42	2.00	0.14	
IMDB	$\mathcal{SI}$-c	4	18	6	3.00	**1.00**	4.50	**1.00**	3.00	
	$\mathcal{SI}$-i	3	14	3	**4.67**	**1.00**	4.67	**1.00**	**3.67**	*
	$\mathcal{AD}$-c	137	1037	837	0.11	0.09	**7.57**	4.00	1.46	
	$\mathcal{AD}$-i	85	560	425	0.16	0.12	6.59	4.00	1.59	
	$\mathcal{AD}$-m	86	543	451	0.15	0.12	6.31	3.00	1.07	

* Here, $|V|$, k_W^m, ρ^m, η^m and **d** represent the number of vertices, number of edges, density, average vertex degree, and diameter, respectively, observed for a multigraph pattern, along with a new defined parameter γ. We also show the properties of a simple graph equivalent of a multigraph pattern, i.e., a vertex induced simple subgraph of same set of vertices, V. The properties shown for simple subgraph equivalent are number of edges k_W^s and density ρ^s. For each dataset the best value for each property is highlighted in bold and the best pattern by a star symbol.

algorithm on each dataset are shown. For each dataset, we consider 5 different configurations of SIMP, i.e., the combination of 2 proposed measures of interestingness ($\mathcal{SI}$ and $\mathcal{AD}$ based) and the types of prior belief used. Note that in Chapter 3, we proposed 3 types of prior beliefs, but $\mathcal{SI}$ cannot be used with belief-m, as the corresponding background distribution is a product of independent distributions, where each distribution is not a natural form of the geometric distribution.

We observed in Table 5.2 that the patterns found using $\mathcal{SI}$ for either type of belief are much denser with a high value of ρ^m than the patterns found using the $\mathcal{AD}$ measure. For example, in DBLP2, the value of ρ^m for $\mathcal{SI}$-i observed is 3.94, which is much higher than the maximum value of 1.03 with $\mathcal{AD}$ measure. The observed low values of diameter d and high value of γ in most of the datasets for the $\mathcal{SI}$-c and $\mathcal{SI}$-i configuration indicates that with the $\mathcal{SI}$ measure SIMP discovers patterns having densely connected vertices with a high number of parallel edges. Hence, these observations assert our analysis that patterns found using the $\mathcal{SI}$ measure are denser than the patterns found using $\mathcal{AD}$.

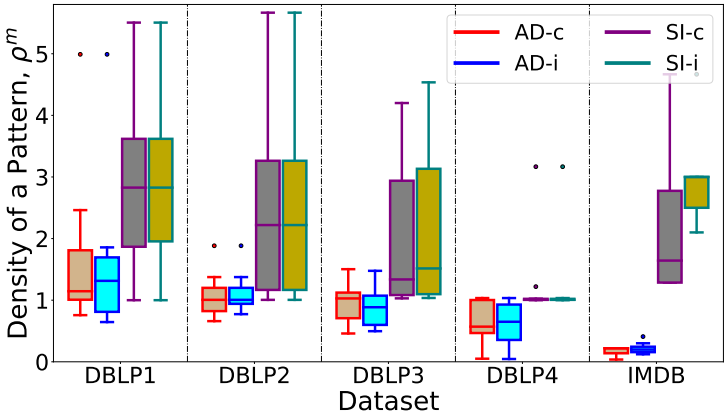
For $\mathcal{SI}$, among the two types of prior beliefs, the densest patterns are observed for belief-i. This is because the background distribution is not uniform, unlike belief-c, and hence, one may expect a high number of edges between a pair of vertices. Thus, p_* in case of belief-i is even smaller than belief-c, which leads the SIMP algorithm with $\mathcal{SI}$ measure to find even denser patterns for belief-i.

Among the two measures, i.e., $\mathcal{AD}$ and $\mathcal{SI}$, patterns corresponding to $\mathcal{SI}$ are smaller in size, with fewer vertices compared to patterns found using $\mathcal{AD}$. We argue that one of the reasons is the fact discussed in the previous section, that $\mathcal{SI} \leq \mathcal{AD}$. Also, a hill climber based algorithm will allow the pattern to grow in size only when the increase in $\mathcal{SI}$ or $\mathcal{AD}$ is more than the complexity required to encode one additional vertex, i.e., change in $\mathcal{DL}$. Thus, under similar conditions, a hill climber during the search is more likely to add a vertex in a seed pattern when the measure of $\mathcal{AD}$ is used, in place of $\mathcal{SI}$.

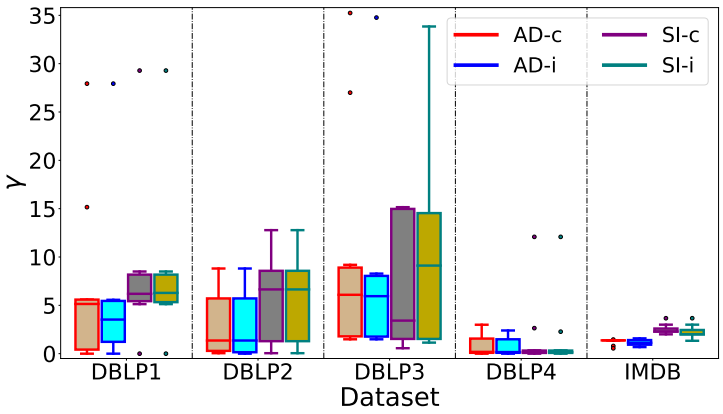
We also observed that IMDB is an exceptional case, in terms of the size of the patterns found with $\mathcal{SI}$ compared to patterns found with $\mathcal{AD}$. In this case, the size of patterns corresponding to $\mathcal{SI}$ is much smaller than the patterns found with $\mathcal{AD}$. This is because among the five datasets, IMDB is least dense, with sparsely connected vertices (see Ta-

ble 3.1). Since $\mathcal{AD}$ is only the difference between the actual and expected sum of edges, not considering the size of a pattern. Hence, it allows the algorithm to grow a pattern during the search process. On the other hand, $\mathcal{SI}$ has the term of n_W^s which reflects the pattern's size, allowing a pattern to grow only if a pattern is densely connected with a relatively high number of edges.

It shall also be seen that even though with $\mathcal{SI}$ we discover highly dense patterns, these may not be every time qualitatively useful. As seen for IMDB, patterns with a size of 4 and 3 vertices with belief-c and belief-i, respectively, may not be a piece of useful information. Hence, in this case, $\mathcal{AD}$ is a decently better measure qualitatively, as observed in Section 3.6.4.



(a) Density of a multigraph pattern, ρ^m in each dataset



(b) γ of a multigraph pattern in each dataset

Fig. 5.4. Shown are the properties of the top-10 patterns found by each configuration of SIMP.

The observations made above are also noticed for the top-10 patterns. In Figure 5.4, we observe that among different configurations of SIMP, the densest patterns are found using $\mathcal{SI}$-i configuration, followed by $\mathcal{SI}$-c, $\mathcal{AD}$-c and $\mathcal{AD}$-i, in non-ascending order. In Figure 5.4a, the distribution of density of all top-10 patterns for each configuration is shown. While, in Figure 5.4b, the distribution of γ value for all top-10 patterns in each case is shown. In Figure 5.4a, in case of configuration $\mathcal{SI}$-i, the median value of density is greater than the other cases in each dataset. However, in Figure 5.4b, the median value of γ is greatest for $\mathcal{SI}$-i configuration only in 3 out of 5 datasets. The parameter γ is also dependent on the size of a pattern, similar to the average vertex degree and may not always represent a generalized trend.

5.4.2 Results on Evolving Multigraphs

In this section, we demonstrate the efficacy of the proposed framework and DSIMP on some real world examples of evolving multigraphs. The datasets used are shown in Table 5.3.

Table 5.3. Shown are the properties of the evolving multigraph datasets used in this section. Here, $|V|$ is the total number of nodes in the graph, $|E_S|$ is the total number of unique edges without timestamp (also removing parallel edges), $|E_T|$ is the total number of edges with timestamp, T is the total time period for which the edges in the graph are considered, t is the time period covered by each individual state, and $|\mathbb{S}|$ is the total number of states considered for each dataset.

Dataset	$\lvert V\rvert$	$\lvert E_S\rvert$	$\lvert E_T\rvert$	T	t	$\lvert\mathbb{S}\rvert$
DBLP[5.1]	44000	123979	154719	10 years	1 year	10
TheMovieDB[5.2]	8292	236691	254467	10 years	1 year	10

The two datasets in Table 5.3 are constructed, where vertices represent authors or actors, and an undirected edge between two vertices represents a co-authored publication or co-acted movie, in DBLP and TheMovieDB, respectively. Each dataset is constructed for the total timespan between the year 2009 and 2018, i.e., 10 years. In DBLP, we

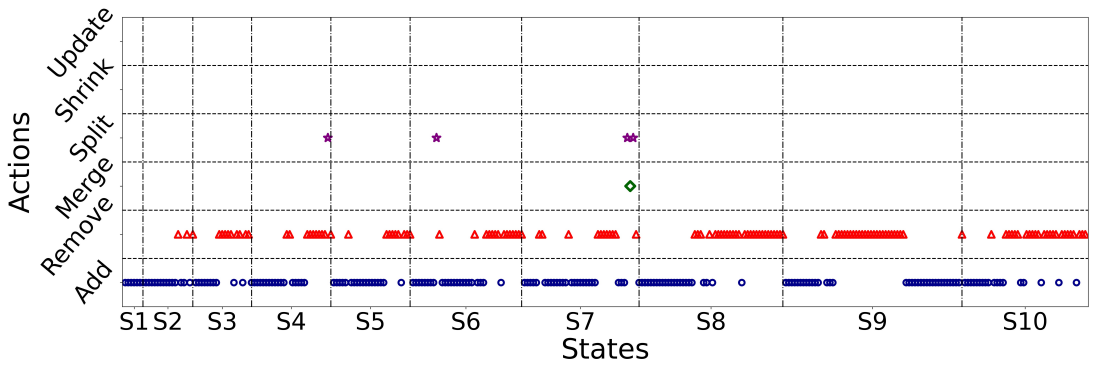
consider only the publications in any of the top-20 Machine Learning and Data Mining conferences and journals. In TheMovieDB, we consider only those actors who have a popularity score of at least 2. Since each state's timespan is one year in each dataset, there may exist more than one publication or movie—co-authored or co-acted—by any two authors or actors, respectively. Hence, DBLP and TheMovieDB datasets are best represented as evolving multigraphs where each snapshot is a static multigraph.

We use the prior belief of type 'c' for each dataset for experiments, and to discover new constraints or patterns, we fix the parameter 'q' to 0.01 with 30 independent 'interest' based seed runs.

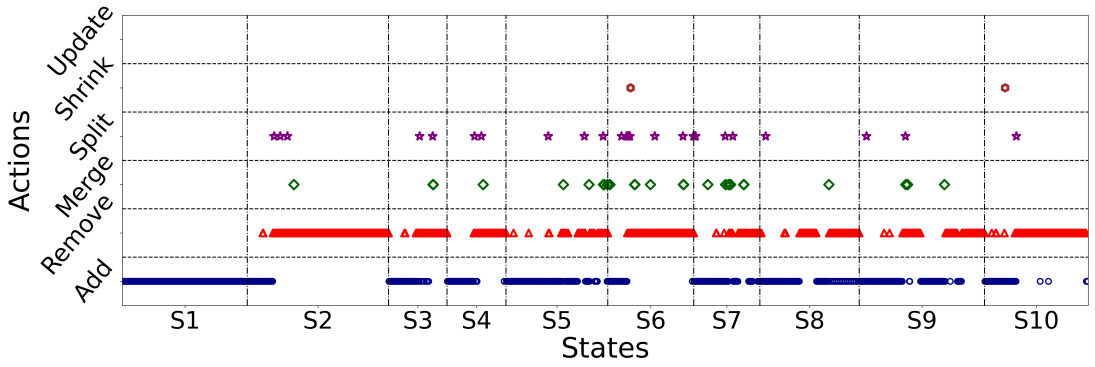
In Figure 5.5, the fingerprint of the summary generated is presented for both DBLP and TheMovieDB. The fingerprint is a visualization of actions suggested at each iteration of the DSIMP algorithm. It is observed that `add` and `remove` are the two most common suggested actions among the 6. This observation indicates that most of the subjectively interesting communities are short-lived (usually for one year) or are constant not evolving with time, i.e., the pattern's size is constant and is similarly dense during this period. However, there are also patterns identified which dwindled or grew with time via `shrink`, `split` and `merge` actions. Though the number of such actions performed in DBLP are significantly lower is number. It shall also be seen that no `update` action is suggested for either of the datasets. One reason is that the density of the initial patterns found is high, and the pattern may not be found to grow beyond it.

In Figure 5.6, we observe that with each action performed the codelength—required to encode the data given the background distribution—decreases and the mean of the average density of all patterns corresponding to the constraints in the set $\mathbf{C}_s$ increases. This observation is in line with the observations for evolving simple graphs in the previous chapter. This is because DSSG and DSIMP have the same structure, and both work on similar ideas.

Interestingly, in Figure 5.7, we observe that DSIMP discovers the set of constraints for each state with a high compression ratio but lower coverage. For example, in DBLP, the maximum compression ratio of above 15% is observed with coverage around 0.6% of 28 constraints in state S3. Similarly, the smallest compression ratio for TheMovieDB is

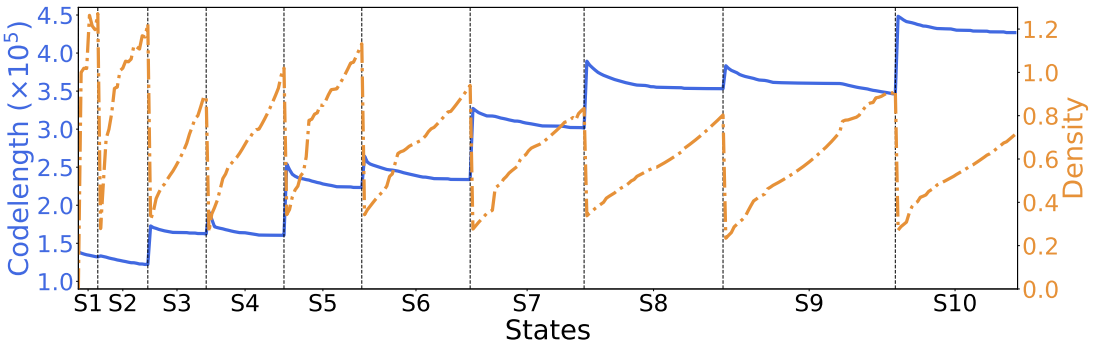


(a) DBLP

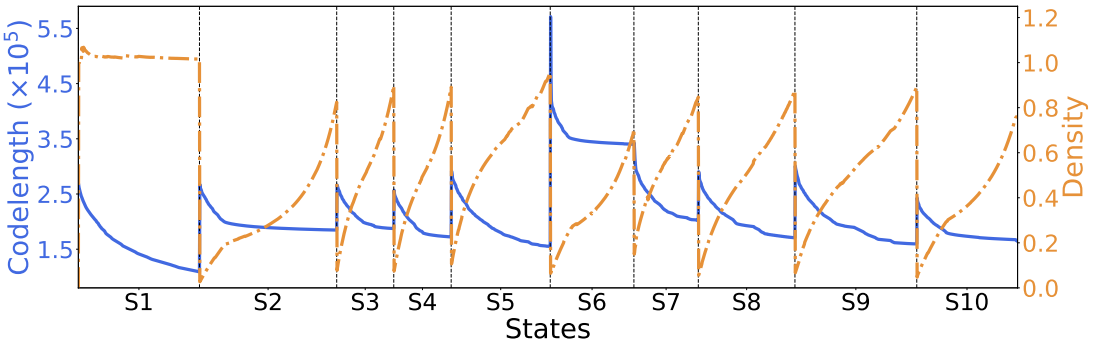


(b) TheMovieDB

Fig. 5.5. Shown are the fingerprints of the summaries generated for evolving multigraphs, i.e., the horizontal axis represents the iteration in each state and the vertical axis represents the actions performed during the corresponding iteration.



(a) DBLP



(b) TheMovieDB

Fig. 5.6. Codelength (blue solid line) vs average of the average densities of patterns in set $\mathbf{C}_s$(orange dashed dotted line) vs states. The vertical dashed lines indicates the change of state and the horizontal axis represent from left to right all iterations, where a series of actions is performed for each consecutive state.

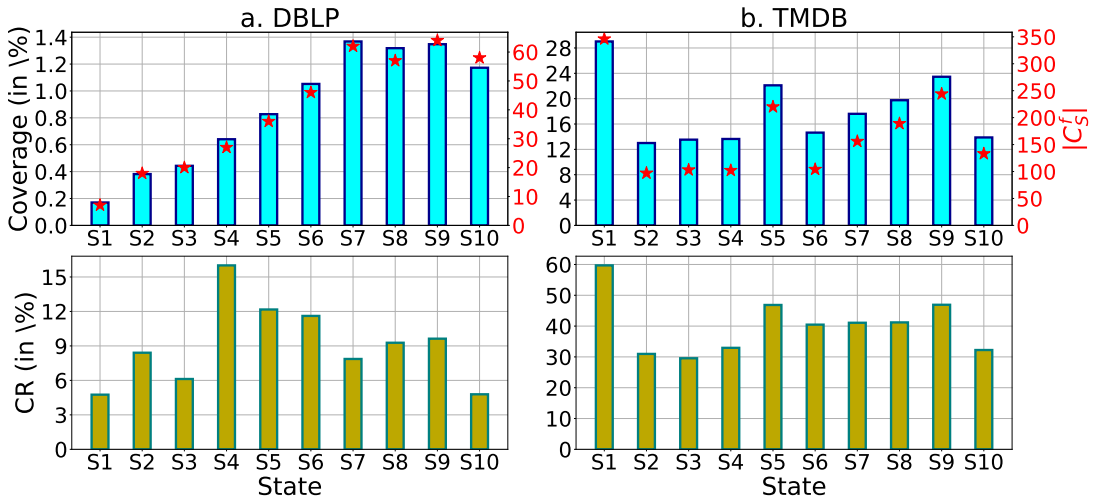
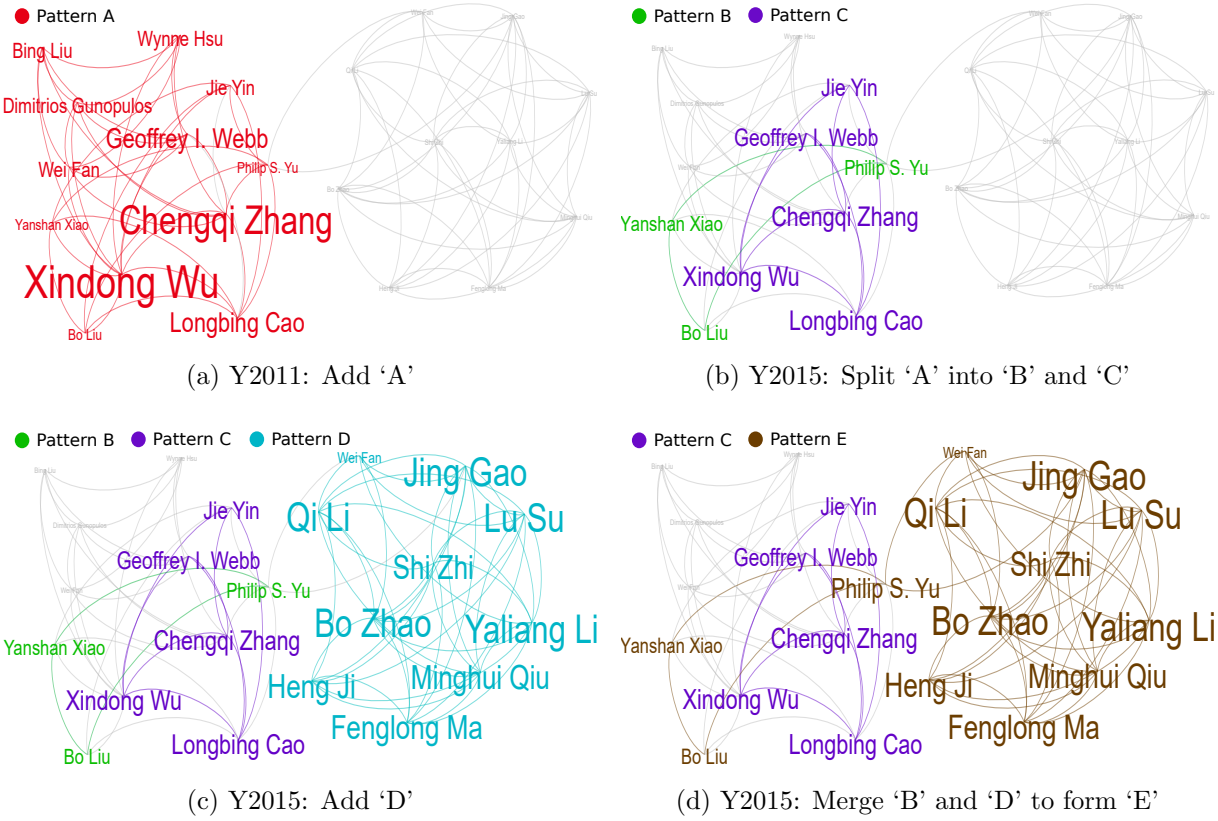


Fig. 5.7. Properties of the found set of patterns (or constraints) $\mathbf{C}_s$ in each state s. Here, $|\mathbf{C}_s^f|$ represents the number of constraints after each state (represented by red star symbols), coverage represents the fraction of vertices of the dataset covered by all patterns combined, and CR represents the compression ratio, i.e., 1 minus the ratio of the encoding cost (number of bits, computed as $-\log_2 P(\mathbf{D})$) of the data given the initial background distribution and given the final background distribution.

observed in state S2, where with 12% coverage via 118 patterns more than 30% of compression is observed. This observation indicates that the proposed methodology discovers information which is surprising or not known to the analyst.

We now discuss the information captured by performing the changes in the constraint set at each step. In this process, we identified a total of 188 and 1346 chains of evolution or evolving patterns (see Figure 4.1g) in DBLP and TheMovieDB network, respectively. We discuss one example each of an evolving pattern found in both networks.

a. DBLP**:** In this case, we present an example of a closely connected community of 22 authors (see Figure 5.8). In the year 2009, a pattern of 12 authors is discovered having a high number of co-authored publications, say Pattern A. In the year 2015, the group is now found to be working in two different subgroups, and hence, DSIMP suggested `split`ting the pattern into two, i.e., Pattern B & C of size 3 and 5 vertices, respectively. However, it is also discovered that one of the subgroups is closely connected to a new group of 10 authors, represented as Pattern D. Thus, DSIMP performed a `merge` action to form



(a) Y2011: Add 'A'

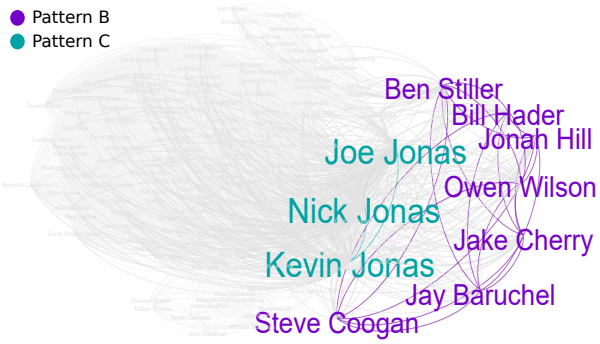
(b) Y2015: Split 'A' into 'B' and 'C'

(c) Y2015: Add 'D'

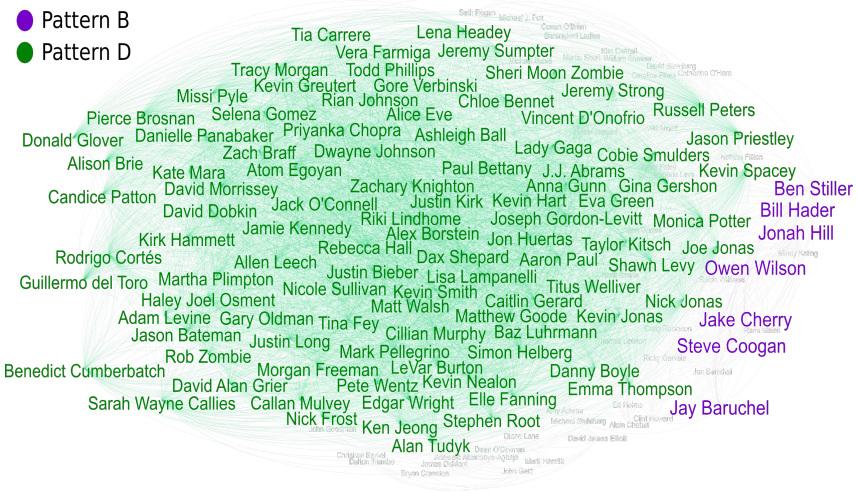
(d) Y2015: Merge 'B' and 'D' to form 'E'

Fig. 5.8. Shown is the evolution of patterns in the DBLP network from Year 2009 to 2015. To avoid cluttering, parallel edges are not shown.

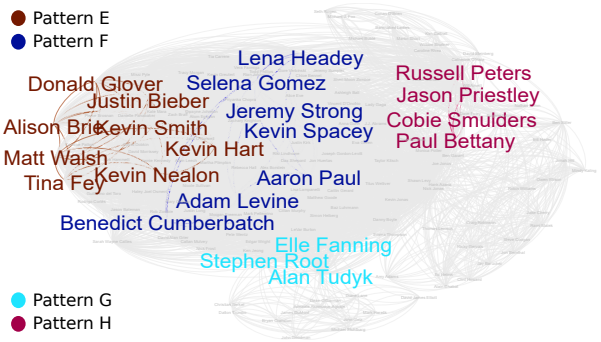
(a) Y2009: Add 'A'



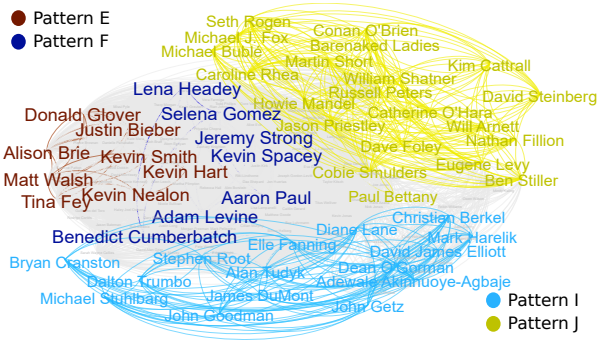
(b) Y2010: Split 'A' into 'B' & 'C'



(c) Y2014: Add a pattern and merge with 'C' to form 'D'



(d) Y2015: Split 'D' into 'E', 'F', 'G' & 'H'



(e) Y2015: Add a pattern and merge with 'G' to form 'I'; Add another pattern and merge with 'H' to form 'J'

Fig. 5.9. Shown is the evolution of patterns in TheMovieDB network from Year 2009 to 2016. To avoid cluttering, parallel edges are not shown.

Pattern E. The three significant patterns in the chain of evolution are Patterns A, C and E, which are densely connected with respective densities of 0.94, 1.0 and 0.91. Hence, this example demonstrates the evolving dynamics and associations between authors with time.

b. TheMovieDB**:** In Figure 5.9, we show an example of how a group of 145 actors may have worked during the time between the year 2009 and 2016. In the year 2009 (see Figure 5.9a), a community of 25 actors is discovered who worked together in 7 movies that released this year with comedy being the most popular genre of 6 movies. In the following year, a `split` is suggested to decompose the pattern A into two patterns 'B' and 'C'. The actors of pattern 'B' worked together in 1 movie of a music genre, while actors of pattern 'C' worked together in a total of 5 movies having family, comedy and drama as prominent genres. In Figure 5.9c, the resultant pattern 'D' of `add` followed by merge with pattern 'C' is shown, as suggested by DSIMP. Pattern 'D' represents that the community now consists of 97 actors who are connected for working together in a total of 33 movies. Among these movies, the three most common observed genres are comedy, drama and thriller. In year 2015 (see Figure 5.9d and 5.9e), pattern 'B' is suggested to be `removed` and `split` is performed on pattern 'D', decomposing it into 4 different patterns 'E', 'F, 'G', and 'H'. Also, in the same year two `merge` actions are also performed following two `add` actions. In the first `merge`, pattern 'G' is merged with a newly found pattern to bring about pattern 'I'. Similarly, pattern 'H' is merged with another newly discovered pattern to result in pattern 'J'. The most commonly observed genre in each pattern, i.e., 'E', 'F', 'I' and 'J', are comedy, drama, action and drama, respectively. Finally, in the year 2016, all the 4 patterns are suggested to be `removed`. Thus, this example demonstrates how the community of actors evolved where they usually worked in the movies of comedy and drama genres.

From the above experiments, it is evident that DSIMP is equally capable as DSSG, for summarizing evolving multigraphs. The DSIMP algorithm efficiently captures the evolution of different interesting communities in the multigraph networks discovered when found to be surprisingly dense. It shall be noted that even for the same dataset, i.e., TheMovieDB, in either scenario—considering it as an evolving simple graph (for DSSG)

or as an evolving multigraph (for DSIMP)—the summaries generated are significantly different.

5.5 Summary

In this chapter, we introduced a new instance of the novel problem of subjective summarization of sequential data, as presented in Chapter 4. Here, we presented the case of subjective summarization of evolving multigraphs. We argued that the measures to quantify the interestingness of a simple subgraph pattern and the strategy to encode the corresponding information cannot be used for multigraph patterns. Hence, we proposed a novel self-information based interestingness measure as proxy to information content in order to discover multigraph patterns in a network. The proposed measure can be efficiently computed with lower time complexity compared to information content and discover patterns which are densely connected. Besides, we also proposed a unique encoding strategy to encode a multigraph pattern. The efficacy of the proposed measures is demonstrated by incorporating them in an algorithm termed DSIMP and performing experiments on real-world examples of evolving multigraph networks.

Chapter **6**

Investigating Airline Networks

In the previous chapters, we saw that subjectively interesting patterns reveal information relative to what the analyst already knows. In this chapter[6.1], we use this fact to study airline networks to learn unexpected patterns. In airline operations, decisions are made on scheduling and other operational problems after gathering insights from the historical data. Thus, subjective interestingness based data mining methods provide a means to learn beyond what airliners may already know. Hence, in this chapter, we present various applications of the different proposed subjective interestingness based algorithms in this thesis.

[6.1] Parts of this chapter has been published in

[Kapoor et al., 2020]: **Kapoor, S.**, Saxena, D.K. & van Leeuwen, M. Discovering subjectively interesting multigraph patterns. In *Mach Learn*, Springer, **109**, 1669–1696 (2020); and

[Kapoor et al., 2021]: **Kapoor, S.**, Saxena, D.K. & van Leeuwen, M. Online summarization of dynamic graphs using subjective interestingness for sequential data. In *Data Min Knowl Disc*, Springer, **35**, 88–126 (2021).

6.1 Introduction

High traffic volume in an airline network is a significant concern, as an airline would like to avoid interruptions and delays in their operation network. Like in a road transportation network, in an airline network, heavy traffic on specific routes causes congestion, resulting in delays, high demands of gates at the airports, increased fuel consumption, and increased workload of the support staff including air traffic controllers [Wu, 2016]. Together, these factors cause degradation in airlines and airports' on-time performance measures, causing heavy losses in the operations [Zou and Hansen, 2012].

Over recent years, the use of historical data to study airline operations has been widely explored. The most extensively studied problems in the aviation domain include: flight delay analysis & prediction [Carvalho et al., 2020]; impact of delays on airline cost, airfare, and flight frequency [Ball et al., 2010; Peterson et al., 2013; Zou and Hansen, 2014]; delay propagation in national airspace [Wu and Wu, 2018]; aircraft maintenance routing [Liang et al., 2015]; and airline schedule generation [Zhou et al., 2020]. The focus in aviation has been on building descriptive, predictive and prescriptive models. Descriptive models are used to enhance the understanding of the data, build deeper insights into the targeted problem, and verify the causes of events in the data. On the contrary, predictive models are used to forecast an event, while prescriptive models aid in decision making.

Flight delay analysis is one of the routine problems studied extensively to build both descriptive and predictive models. In the past, several models for predicting flight delays have been proposed employing different machine learning algorithms such as random forest [Gui et al., 2019], support vector machines [Wu et al., 2019], reinforcement learning [Balakrishna et al., 2008], gradient boosting decision tree [Manna et al., 2017], and deep learning methods [Kim et al., 2016]. On the other hand, descriptive models have been proposed to learn patterns causing delays using methods such as frequent pattern mining [Sternberg et al., 2016], subgroup discovery [Proença et al., 2018], K-means clustering [Woodburn and Ryerson, 2014], and DBSCAN [Tian et al., 2019].

In the literature, different studies have been carried out to predict different categories of delays. The Bureau of Transportation Statistics, US [BTS, 2019] classifies delays into

5 major categories, which are:

- *Air Carrier delay*: delays in control of airliners such as aircraft maintenance, crew issues, baggage handling, or refuelling.
- *Extreme Weather*: delays due to extremely harsh weather conditions such as cyclone, tornado, and heavy snowfall.
- *Late Arriving Aircraft*: delays due to the aircraft used in a flight which has been delayed on it previous flight.
- *National Aviation System (NAS)*: delays in the airside operations due to airport or air traffic control operations, nonextreme weather, and heavy traffic.
- *Security*: delays due to boarding operations such as screening of passengers, boarding/reboarding, and high passenger flow.

To the best of the author's knowledge, NAS delays have been limitedly studied, with little or no dedicated work on investigating the factor of heavy traffic volumes. Wang et al. [2003] studied the impact of NAS delay propagation, Hao et al. [2014b] investigated the impact of the specific airport on NAS delay, and Sridhar and Swei [2006] explored the relationships between the factors causing NAS delays. While a short-term prediction model for NAS delays based on weather index is given by [Sridhar and Chen, 2009].

Similarly, another commonly studied problem in the aviation domain is airline scheduling. It is a complex problem composed of several sub-problems. One of the sub-problems is optimal Scheduled Block Time (SBT) selection. Based on historical data analysis, several descriptive models have been proposed to study the behavior of airliners towards SBT selection [Hao and Hansen, 2013, 2014]. In the most recent work, Kang and Hansen [2017] investigated airliners' behavior upon SBT adjustment. Another gap observed in the literature is the application of network analysis to study airliners' behavior in scenarios of heavy traffic volumes.

As mentioned earlier, with the research gaps, we argue that subjectively interesting subgraph patterns can provide potential insights about the network, such as regions that experience or may experience unexpectedly heavy traffic volumes. These insights can be used to understand the dynamics of a network, select the optimal SBT for a flight,

and possibly even predict delays in the network. Therefore, this chapter discusses the possible applications of subjective interestingness based algorithms in learning from an airline network. We represent the airports by vertices and flights between the airports by directed edges from an origin airport to a destination airport in an airline network. We consider various snapshots of the network for a given time duration, where edges are added only for those flights that are either departing from an origin airport or arriving at a destination airport in the given time duration.

The remaining part of this chapter is organized as follows. In Section 6.2, we analyze the airline network as a multigraph using the SIMP algorithm and study the discovered patterns. In Sections 6.3 and 6.4, we present a methodology to construct features using SIMPs to build a scheduled block time selection model and delay prediction models, respectively. In Section 6.5, we present an application of the DSSG algorithm demonstrating how an analyst can efficiently study the structural changes in the network in an online and incremental manner to render valuable insights. Finally, we conclude the chapter in Section 6.6.

6.2 Patterns in a Static Airline Network

In this section, we analyze subjective multigraph patterns in an airline network using the SIMP algorithm proposed in Chapter 3. As discussed in previous chapters, such an airline dataset can be best represented as a directed multigraph. We focus on finding regions in the network that are likely to experience a high delay due to heavy traffic, categorized in the data as the NAS (National Aviation System) delay. There could be various factors for NAS delay, but heavy traffic is one of the major factors accounting for NAS delays [BTS, 2019].

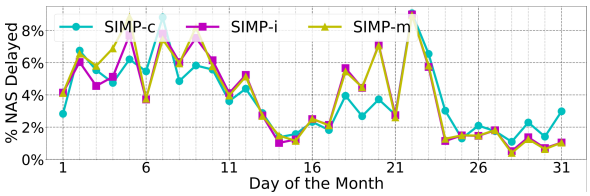
From the on-time performance data by Bureau of Transportation Statistics (BTS), US[6.2]—we consider 298 commercial airports with 450 017 flights that took place in January 2017. As a first case, we investigate the most interesting patterns for each day over the month of January 2017. For each day, we construct the background distribution based on prior beliefs taken from the flight *schedule data*; note that this is a very realistic

[6.2] `https://www.transtats.bts.gov/DataIndex.asp`

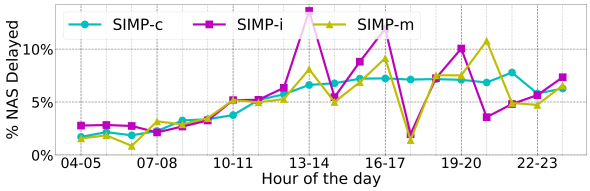
scenario, as the schedule informs our expectations and we look for deviations from these expectations in the actual flight data. As a second case, we build the background distribution from scheduled data for each hour of a specific day, i.e., 22nd of January 2017. We consider flights either arriving or departing from any airport in any time block on the day, we have 20 time-blocks of one hour each (from 0400 hours to 2400 hrs, all converted to Pacific Standard Time). We exclude cancelled flights from the data, as these would have an infinite delay.

The most interesting patterns per time frame found by SIMP are shown in Figure 6.1. Figures 6.1a and 6.1b show that the patterns found by SIMP have a reasonably large number of NAS delayed flights in the set of flights present in the found pattern. This shows that the first patterns found by SIMP-i and SIMP-m have a fairly large 'precision', indicating that a fair number of the NAS delays occurs in these patterns. This is corroborated by Figure 6.1c and 6.1d, which indicates that, among all delayed flights present in a pattern, a fair set of flights are categorized as NAS delayed. To verify that these patterns are the primary source of NAS delay, we computed the 'recall' of the patterns in Figure 6.1e and 6.1f, i.e., the number of NAS delayed flights present in the pattern among all NAS delayed flights in the current view of the network. It was found that SIMP-c has a reasonably large recall, where around 25% of NAS delayed flights were present in around 10% of the airports of the network (see Figures 6.1g and 6.1h). This is because of the large size of the patterns. Upon closely inspecting the patterns found by SIMP-i and SIMP-m, we found that these patterns all have a similar ratio of 'recall' to the percentage of airports in a pattern, but have high 'precision', which supports our hypothesis that NAS delay is most likely to occur in the regions identified by SIMP.

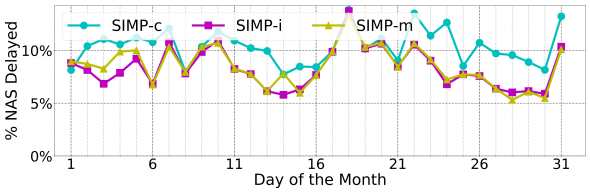
Following the observations on the most interesting pattern per time frame, we analyse the top-10 patterns shown in Figure 6.2. For this analysis, the union of all top-10 patterns is considered, i.e., all the airports and flights that were present in any found pattern are taken together. Analyzing the network over a period of a month, Figure 6.2e shows that each day the top-10 patterns found by SIMP-c, SIMP-i and SIMP-m have a very high presence of NAS delayed flights among all the NAS delayed flights in the network on that day. A similar observation was made in Figure 6.2f while analyzing the airline network,



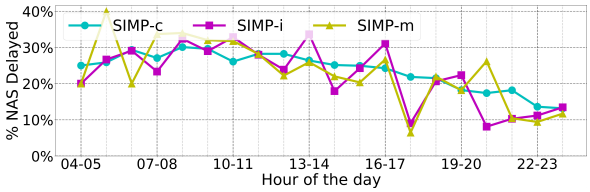
(a) NAS delayed flights among flights in a pattern



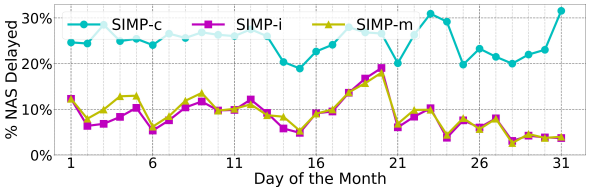
(b) NAS delayed flights among flights in a pattern



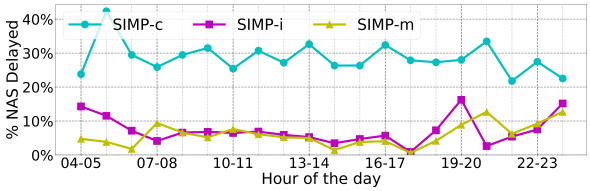
(c) NAS delayed flights among delayed flights in a pattern



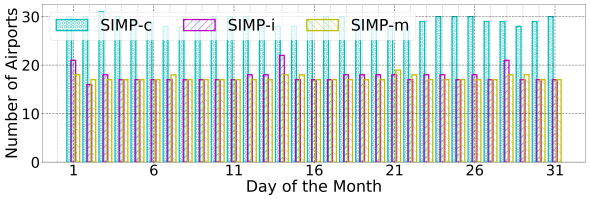
(d) NAS delayed flights among delayed flights in a pattern



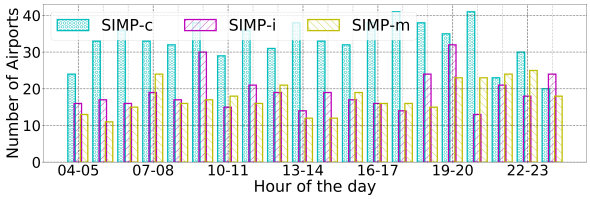
(e) NAS delayed flights in a pattern among all NAS delayed flights in the network



(f) NAS delayed flights in a pattern among all NAS delayed flights in the network

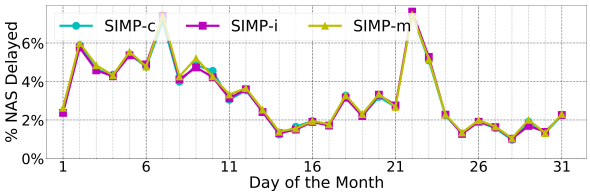


(g) Number of airports in a pattern

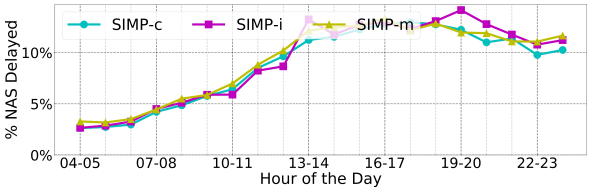


(h) Number of airports in a pattern

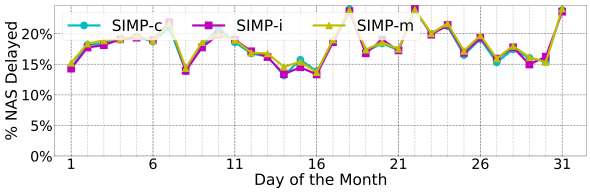
Fig. 6.1. Results of best pattern found by SIMP-c, SIMP-i and SIMP-m for two cases, i.e., (left) the entire month and (right) a single day.



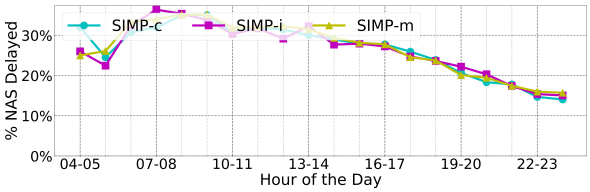
(a) NAS delayed flights among flights in 10 patterns



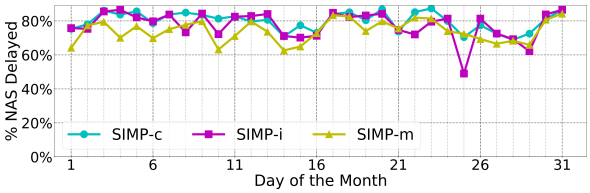
(b) NAS delayed flights among flights in 10 patterns



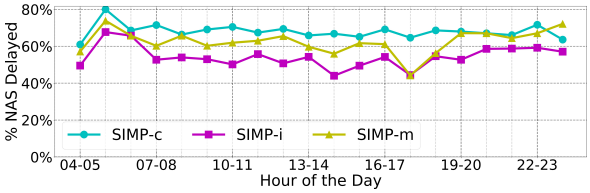
(c) NAS delayed flights among delayed flights in 10 patterns



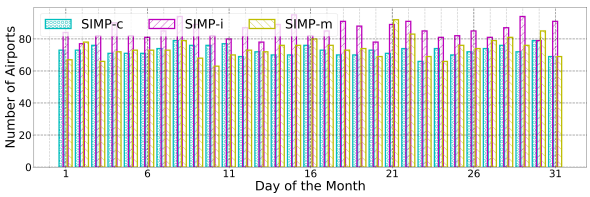
(d) NAS delayed flights among delayed flights in 10 patterns



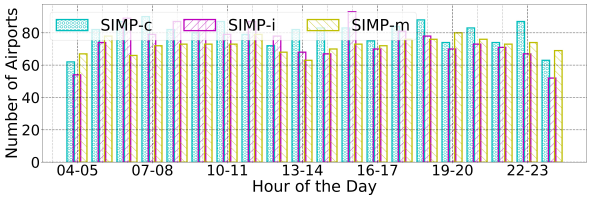
(e) NAS delayed flights in 10 patterns among all NAS delayed flights in the network



(f) NAS delayed flights in 10 patterns among all NAS delayed flights in the network



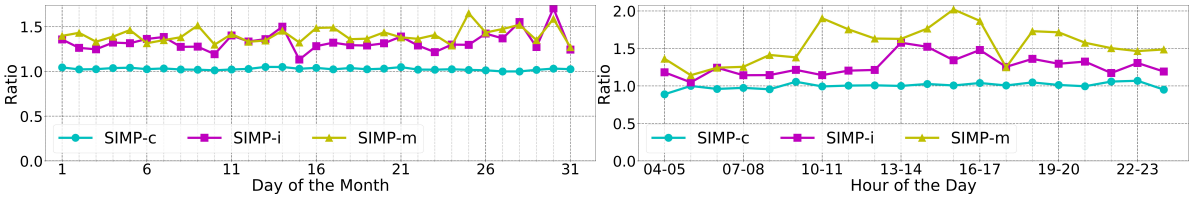
(g) Number of airports in all 10 patterns



(h) Number of airports in all 10 patterns

Fig. 6.2. Results of top 10 patterns found by SIMP-c, SIMP-i and SIMP-m for two cases, i.e., (left) the entire month and (right) a single day.

each hour for a single day. SIMP-c, SIMP-i and SIMP-m follow almost the same trend to account for NAS delayed flights in the top-10 patterns (Figures 6.2a-6.2d).



(a) Ratio vs day of the month

(b) Ratio vs hour of the day

Fig. 6.3. Plots showing the ratio of % of NAS delays present in top-10 SIMP patterns to the % of NAS delays present in a baseline pattern having the same number of edges.

To further investigate this, we compute baseline patterns having the top-r airports with the highest multigraph degree, such that each such pattern has a number of edges (approximately) equal to the number of edges covered by the top-10 patterns found by SIMP. We then compute the ratio of the number of NAS delayed flights covered by the top-10 SIMP patterns to the number of NAS delayed flights covered by their respective baseline patterns, as shown in Figure 6.3. This ratio is always close to one for SIMP-c, indicating that SIMP finds patterns with high densities, very similar to our constructed baseline patterns with this type of belief. SIMP-i and SIMP-m, on the other hand, have relatively high ratios, above 1 and sometimes close to 2, suggesting that these types of belief help discover patterns that correspond to NAS delays. These patterns may not always be structurally dense, i.e., their diameters may be high, but they encompass many air routes with a larger number of flights. This shows the potential of using prior beliefs—such as the ones that we propose in this thesis—for finding patterns that correspond to high traffic congestion, which may lead to NAS delays.

Overall, this exploratory analysis shows that NAS delay is likely to occur in regions of the network that are *subjectively* interesting, i.e., relative to Belief-i and Belief-m. These multigraph patterns might provide strategic information to airliners in the context of flight schedules.

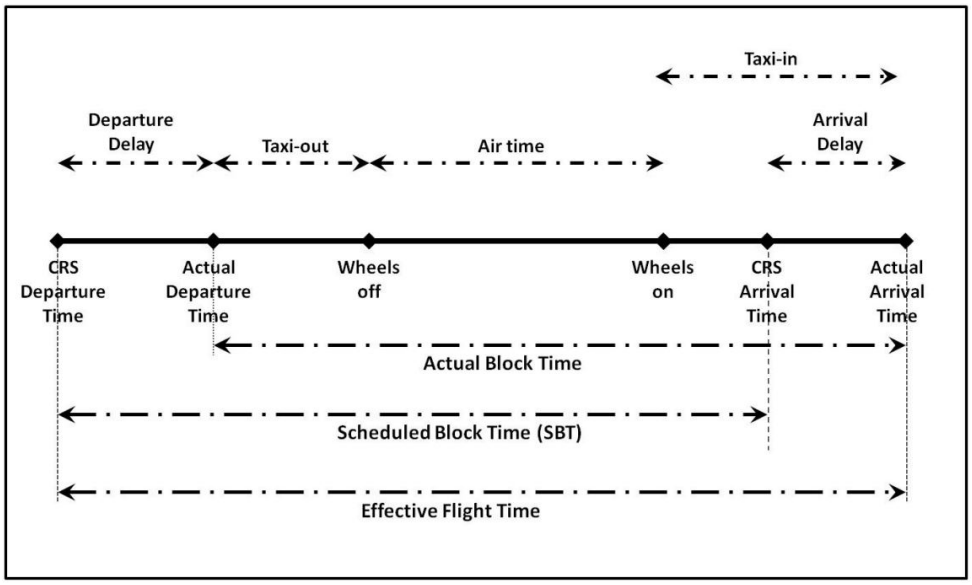


Fig. 6.4. Pictorial representation of Scheduled Block Time (SBT) [Hao et al., 2014a].

6.3 Analyzing Patterns and Scheduled Block Times

In this section, we explore associations between SIMP patterns and Scheduled Block Time (SBT) selection by airlines. The SBT of a flight is defined as the time during which the engines of an aircraft remain powered on. In other words, it is the time between the moment an aircraft leaves the gate of an origin airport and the moment it arrives at the gate of the destination airport, see Figure 6.4. Selecting an optimal SBT is a complex problem as a trade-off between cost-saving and fuel exhaustion is observed. Shorter SBT means less fuel loading and higher profit because of less fuel consumption but at the same time, a likelihood of experiencing fuel exhaustion. Larger SBT comes with high fuel reliability on-board but at the cost of high fuel consumption and low profits. Since fuel expenses occupy the top position in terms of the overall cost to a carrier, selecting an optimal SBT is critical to airliners.

Dataset. In this section, we use the BTS on-time performance data.1[6.6] along with air carrier statistics data from the T-100 domestic segment of U.S..1[6.6] for years 2016 and 2017.

6.3.1 Preprocessing and Feature Construction

In this study, we use the SIMP framework for discovering patterns in an airline network. For each day in the year 2016, we create 22 independent snapshots, each of one-hour duration from 0300 hours to 0100 hours of the following date, where the time is converted to Pacific Standard Time. For each snapshot, we discover (at most) the top 5 subjectively interesting patterns using the SIMP algorithm. The motivation is because of the previous section's observations, where the subjectively dense multigraph patterns in an airline network contain a relatively large number of delayed (specifically NAS) flights. It can be argued that SIMP patterns discover the set of connected airports having a surprisingly high volume of traffic, and in order to decongest the network an airline may alter the schedule as well as the SBT of several flights.

For each of the two types of belief, i.e., belief-c and belief-i (see Section 3.3), we create a binary feature, χ^c and χ^i, respectively. We assign a value 1 to the flights covered by any discovered pattern concerning the type of belief used, else we assign value 0. Further, we have two different formulations of interestingness, i.e., Aggregate Deviation (AD) based (see Chapter 3) & Self-Information (SI) based (see Chapter 5), and two types of airline networks, i.e., scheduled (S) & actual (A) flight network. Thus, we create in all eight different features, represented as $\chi^b_{X,Y}$, where the superscript b represents the type of belief, the subscript X represents the type of interestingness formulation used, and the subscript Y represents the type of network.

The data is aggregated for a quarter of a year. Without loss of generality, we remove all diverted and cancelled flights along with all weekend flights and only consider those flights which are flown at least 50 times in a given quarter, i.e., during aggregation if the count is less than 50. In the process, we finally get a total of 16 954 unique observations in the aggregated dataset. The distribution of each feature in the final dataset is shown in Figure 6.5, and the pairwise correlation between the features is given in Figure 6.6. It is of note that during aggregation, we choose to use the mean value for all the 8 newly created features. In other words, the value of each created feature for each observation is the ratio of the number of times a flight was a part of an interesting pattern to the

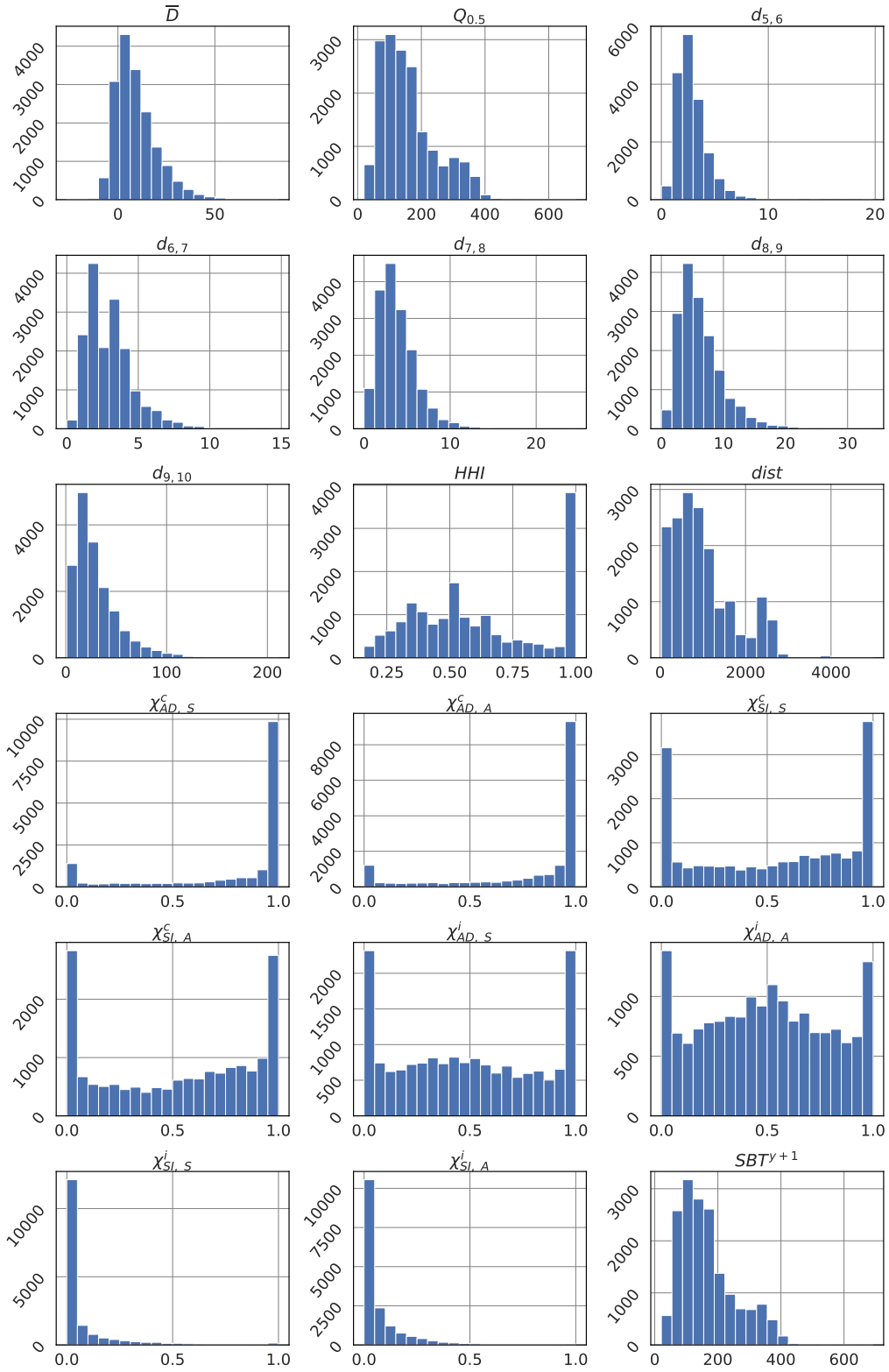


Fig. 6.5. Plots showing the distribution of each feature in the dataset. The $\chi^b_{X,Y}$ features are described in Subsection 6.3.1, the other features in Subsection 6.3.2

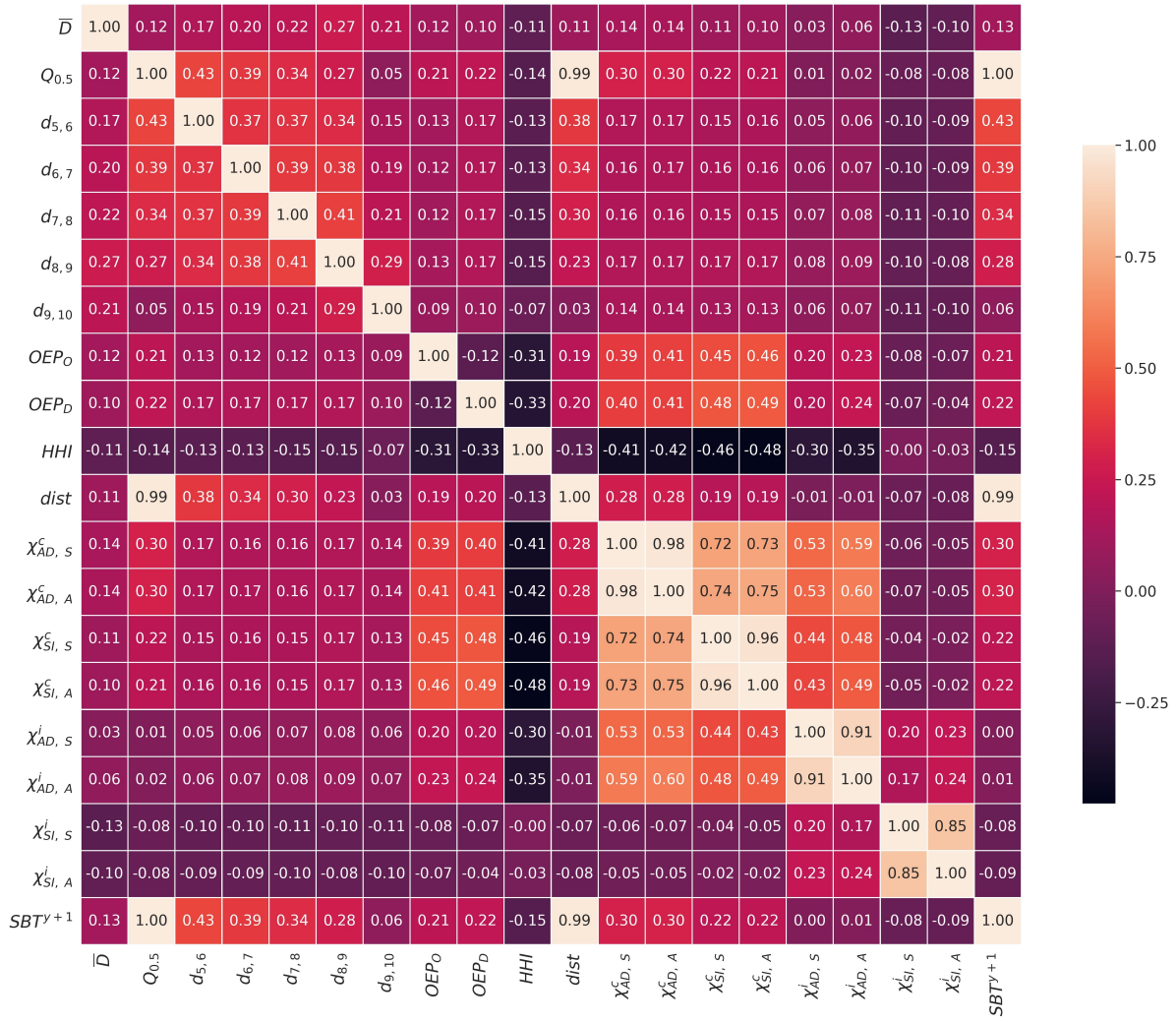


Fig. 6.6. Heatmap showing pairwise correlations between the features in the dataset.

number of times a flight is flown, in a quarter.

6.3.2 SBT Selection Model

The selection of SBT is a complex process, and an airline may consider various factors for decision making. However, to learn from the data and get insight into how an airline may select SBT for a year, Hao and Hansen [2013] gave a well known aggregate percentile model for SBT selection based on multiple linear regression. The authors modelled the SBT for a flight using flying time data by the same flight during the preceding year. The model is given as

$$\begin{aligned} SBT_f^{q,y+1} = \alpha_1 \times \overline{D_f^{q,y}} + \alpha_2 \times dist_{od} + \beta_1 \times Q_{0.5}^{f,q,y} + \sum_{i=5}^{9} \beta_{i-3} \times d_{i,i+1}^{f,q,y} \\ + \alpha_3 \times HHI_{od} + \sum_{q=2}^{4} \times Q_q^y + \gamma_1 \times OEP_O + \gamma_2 \times OEP_D + \kappa, \end{aligned} \tag{6.1}$$

where all α_i, β_i, γ_i and κ are coefficients learned from the data.

The dependent or explanatory variables, which are aggregated for a quarter q of a year y, include:

1. mean departure delay of a flight f, as $\overline{D_f^{q,y}}$,
2. distance between the origin-destination pair (od) of f, as $dist_{od}$,
3. median flying time taken by f, as $Q_{0.5}^{f,q,y}$,
4. difference between every 10^{th} percentile of flying time beyond median flying time, as $d_{i,i+1}^{f,q,y} = Q_{(i+1)/10}^{f,q,y} - Q_{i/10}^{f,q,y}$,
5. dummy variables to capture the seasonal effects of each quarter, as Q_q^y,
6. dummy variables OEP_O and OEP_D to indicate if origin or destination airport is an OEP 35 airport[6.3], respectively,

[6.3]The 35 Operational Evolution Partnership (OEP) airports are the major commercial U.S. airports with significant flight operations.

7. market share of a carrier or competition among carriers in an OD pair, as HHI_{od}. The HHI_{od} is the Herfindahl–Hirschman Index calculated as

$$HHI_{od} = \sum_{c \in \mathcal{AC}_{od}} \left(\frac{s_c}{s_{od}} \right)^2, \tag{6.2}$$

where s_c is the total number of seats offered a carrier c, and s_{od} is the total number of seats offered by all carriers $\mathcal{AC}_{od}$ flying between the given OD pair.

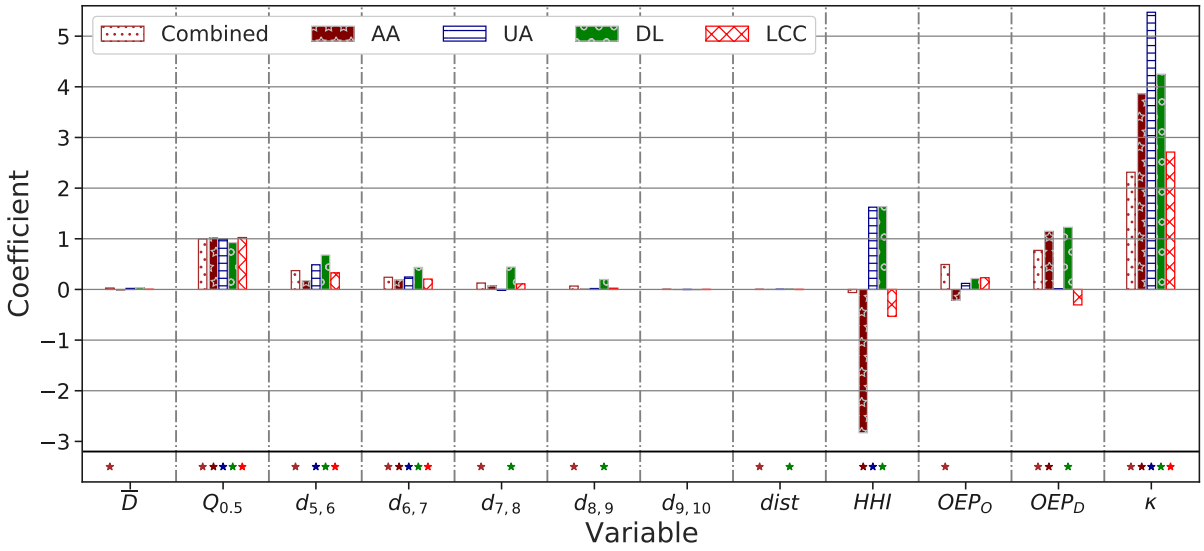


Fig. 6.7. Estimation results of the aggregate percentile model (Equation 6.1) on the different subset of the data depending on the airline carrier. The significant coefficients with p-values less that 0.001 are highlighted with a star symbol below. The observed R-squared value for all models is 0.997. Here, AA—American Airlines, UA—United Airlines, DL—Delta Airlines, LCC—Low Cost Carriers, Combined—All Flights.

In Figure 6.7, we show the estimated coefficients for the model in Equation 6.1, when the data of the year 2016 is used to fit the SBT selected by airlines for the year 2017. It is of note that for simplicity, we omit variables Q_q from the model because similar to [Hao and Hansen, 2013], we also found that the respective coefficients are insignificant. Here, it is observed that airlines tend to select the median flying time as the baseline for selecting SBT for the following year. This is because p-values less than 0.001 indicates that the coefficients for $Q_{0.5}$, in models for each subset of data, are significant and near

to 1. Similarly, the coefficients of $d_{5,6}$ and $d_{6,7}$ are also found to be significant in almost every case. Thus, it is correct to say that most of the airlines target 70^{th} percentile of flying time as a baseline, which is also targeted Block Time Reliability (BTR) [Hao and Hansen, 2014]. With similar observations, an exceptional case is Delta Airlines which is found to target 90^{th} percentile BTR.

Based on the significance value of each coefficient, we also observed the following points an airline is found to follow while selecting the SBT for the succeeding year.

1. *dist*: Only Delta Airlines has a positive and significant coefficient, which suggest that the airline add few minutes to the baseline corresponding to this variable.
2. OEP_O: No airline is found to make changes to the baseline based on this variable.
3. OEP_D: American and Delta airlines add more than a minute to the baseline if the destination is an OEP category airport for a flight.
4. HHI: American airlines subtracts from the baseline based on the market share on a flight route. At the same time, Delta and United airlines add to the baseline based on this factor.
5. κ: All airlines are found to add some minutes to the baseline, irrespective of any factor. United airline adds the most—more than 5 minutes, Delta and American add near to 4 minutes, whereas the Low-Cost Carriers add the least—which is 3 minutes.
6. $\overline{D}$: In the combined case, the coefficient is found to be significant, but for airline-specific case the coefficients are insignificant. It is somewhat expected as departure delay is a part of effective block time but not the actual block time, which is the only time span that matters to the airlines as the fuel would be consumed only in this duration (see Figure 6.4). Thus, any delay that occurred during the airborne, taxi-out and taxi-in phases may affect an airline's choices. Nevertheless, these factors are counted indirectly in terms of actual flying (block) time.

Thus, we can conclude from the observations that American and Delta airlines evaluates and learns from more decision-making factors than the other airlines, and thus, have

a more rigorous model. On the other hand, United airlines and the Low-Cost Carriers, generally make the decisions very straight-forward, where they target 70^{th} percentile BTR and add few minutes to select the SBT.

We found that the aggregate model effectively models the choices that an airline may have made to select the SBT for the year 2017, where the R-squared value of 0.997 is observed in each case. It is of note that an airline choice may not be limited to these factors as the industry practice is to consider a feedback loop where the SBT of specific flights are altered based on several other factors [Hao et al., 2014a]. Since this knowledge is not shared by the airliners, it is worth exploring if the SIMP pattern-based variables explain the SBT selection model used by airlines.

6.3.3 SBT Modelling using SIMP Patterns

In this section, we explore whether SIMP variables can potentially explain the SBT selection model of airlines. The motivation is to learn whether an airline integrates the knowledge of high traffic volumes (both anticipated and observed dense region) in the process of selection of an optimal SBT or not. For this, we adapt the model given by Hao and Hansen [2013] (Equation 6.1).

In the exploratory analysis, in Figure 6.5, we observe that the eight constructed features have different distributions compared to the decision variables considered in Equation 6.1. We also observe in Figure 6.6 that the new features have a maximum correlation with only three decision variables, including *HHI*, OEP_O and OEP_D. These decision variables capture the traffic densities between an OD-pair, at the origin airport and the destination airport, respectively. Similarly, the SIMP patterns also discover the regions of high traffic volume in the network. However, the SIMP patterns find time-based patterns or regions that are not best represented by any of the three variables mentioned above.

Since among the newly constructed features high correlation is observed in multiple cases (see Figure 6.6) and to better understand each feature's relevance, we consider eight different and independent multiple linear regression models. In each model, we only consider only one of the eight SIMP-based features as a dependent variable. Hence, a

model is represented by $M^{b}_{X,Y}$ and is given as

$$\begin{aligned} SBT_f^{q,y+1} = \alpha_1 \times \overline{D_f^{q,y}} + \alpha_2 \times dist_{od} + \beta_1 \times Q_{0.5}^{f,q,y} + \sum_{i=5}^{9} \beta_{i-3} \times d_{i,i+1}^{f,q,y} \\ + \alpha_3 \times HHI_{od} + \gamma_1 \times OEP_O + \gamma_2 \times OEP_D + \kappa \\ + \omega \times \chi_{X,Y}^{b,f,q,y}. \end{aligned} \tag{6.3}$$

Notably, also in this case, we do not consider the dummy variables to capture seasonal effects because of the same reason mentioned earlier.

Regression Analysis:

In the first scenario, we fit the eight multiple linear regression models considering one of the eight SIMP-based features at a time on the complete dataset of all airlines combined. We found that the adjusted R-squared value is close to 1 and can be considered a good fit. We found in Figure 6.8 that the coefficients of SIMP-based variable in 6 out of the 8 models are found to be significant, and the coefficient with maximum magnitude is -1.1. This suggests that airlines may have considered this knowledge in the process of SBT selection, where 1 minute is subtracted from the 70^{th} percentile BTR.

The significant SIMP-based features belong to the category of prior belief-i. For belief-i, we discovered patterns which are small in size and denser compared to the patterns found by belief-c. This indicates that for the flights which are covered by any of the discovered subjectively dense patterns relative to belief-i (most of the time in a quarter)—airliners tend to select around 1 minute shorter SBT. Also, we observed that when belief-c is used in conjunction with the $\mathcal{SI}$ based interestingness measure, then also the SIMP based variables have significant coefficients. This is because the size of the patterns found by this measure are smaller than the $\mathcal{AD}$ based measure. It is also intuitive that as feedback from schedulers, airliners may alter the SBT of an only limited set of flights based on some unknown factor. Among these unknown factors, it can be conjectured that SIMP based variables explain some of the factors that an airline may have considered.

We investigate the same models for specific airlines[6.4] (see Figures 6.9-6.12), including the American Airlines, United Airlines, Delta Airlines (the three major legacy carriers of US) and Low-Cost Carriers (includes Frontier, JetBlue, Southwest and Spirit Airlines).

[6.4]The results in tabular form are relegated to Appendix C.1.

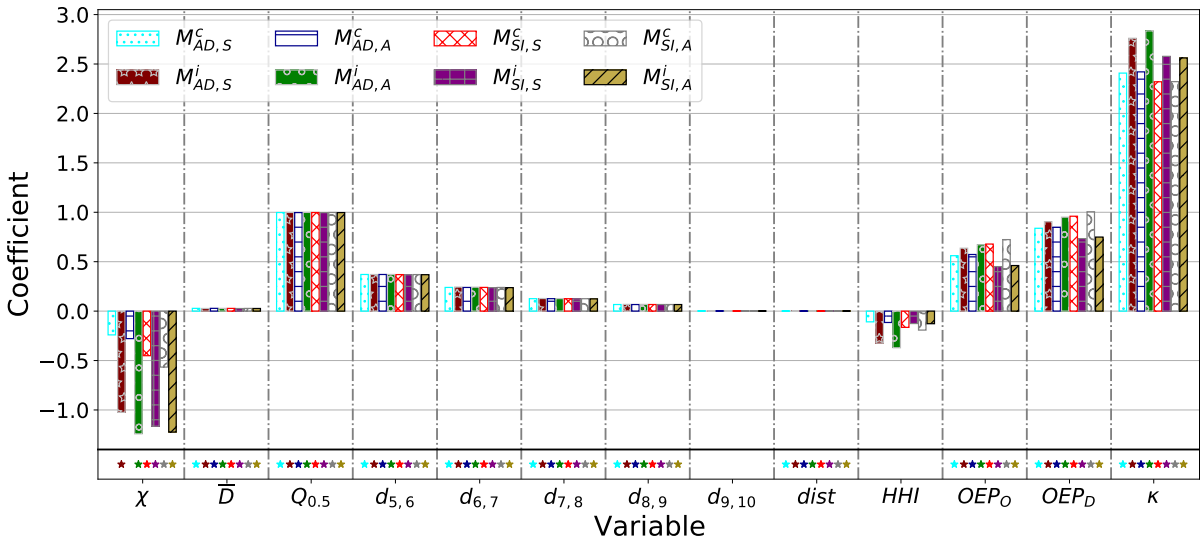


Fig. 6.8. Estimation results of 8 different multiple regression models for the dataset of all airlines combined. The significant coefficients with p-values less that 0.001 are highlighted with a star symbol below. The observed R-squared value for all models is 0.997.

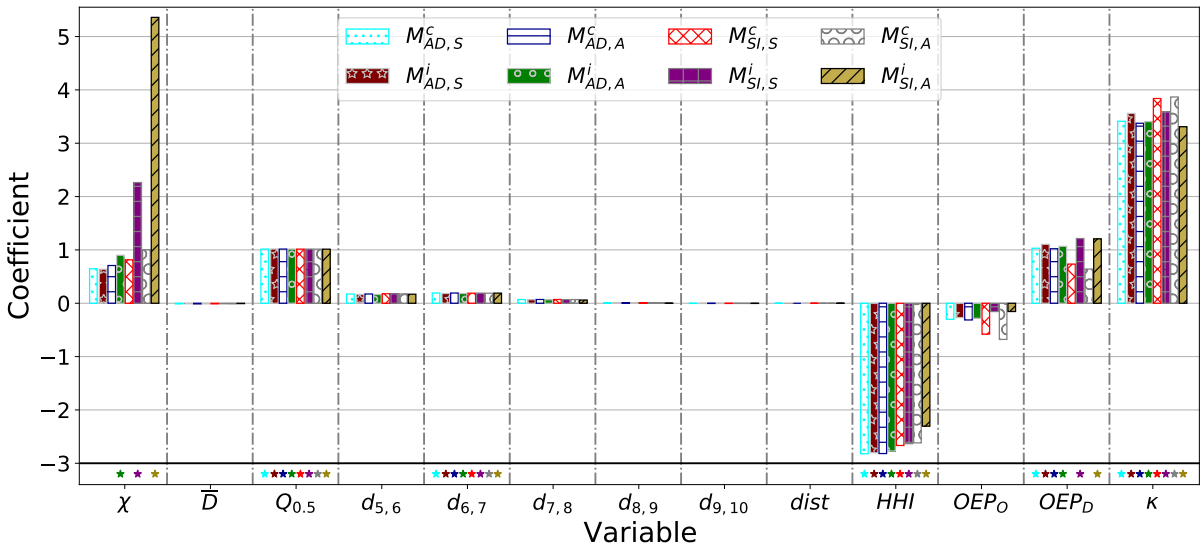


Fig. 6.9. Estimation results of 8 different multiple regression models for the dataset of only American airline's flights (number of observations is 4148). The significant coefficients with p-values less that 0.001 are highlighted with a star symbol below. The observed R-squared value for all models is 0.997.

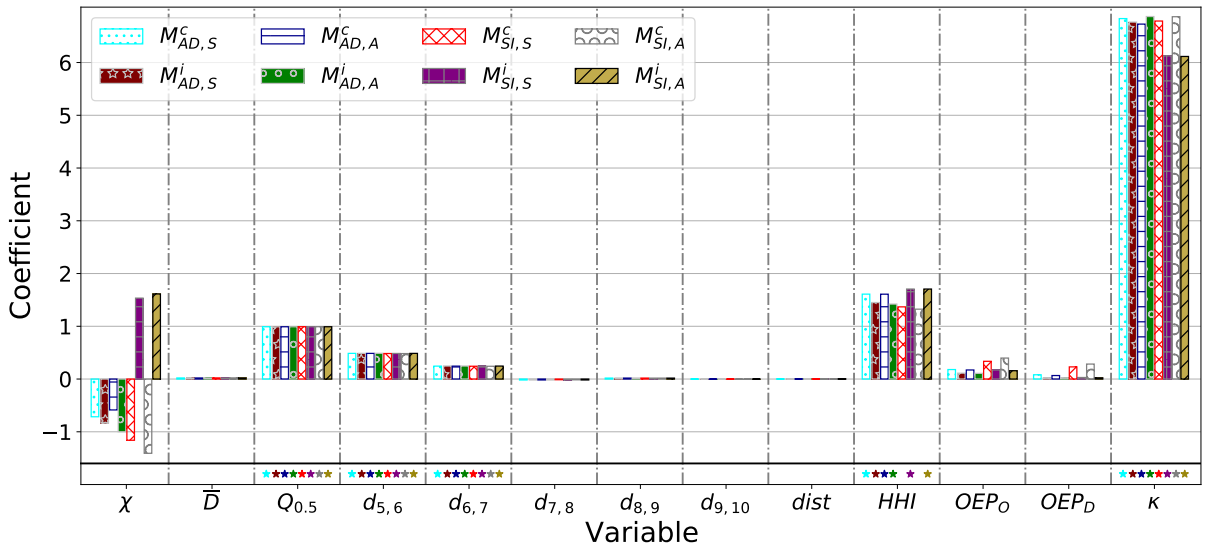


Fig. 6.10. Estimation results of 8 different multiple regression models for the dataset of only United airline's flights (number of observations is 2563). The significant coefficients with p-values less that 0.001 are highlighted with a star symbol below. The observed R-squared value for all models is 0.998.

We observe the largest coefficients for $\chi^i_{SI,S}$ and $\chi^i_{SI,A}$ (also significant, see Figure 6.9). This observation indicates that American Airlines, a legacy carrier, prefers high reliability, as it selects higher SBT for flights that are usually covered by patterns with unexpectedly high traffic volumes, realising both the scheduled and the actual flight operations. In other words, American Airlines adds 5.36 minutes and 2.26 minutes to the selected SBT for the flights which are always covered by corresponding subjectively dense patterns, as indicated by $\chi^i_{SI,A}$ and $\chi^i_{SI,S}$, respectively.

On the contrary, we observe a different behavior for United Airlines (see Figure 6.10), which prefers to only aim at targeted block time reliability without considering any other factor except market competition (through *HHI*). In the case of Delta Airlines (see Figure 6.11), among the new SIMP-based features only the coefficient of $\chi^i_{SI,S}$ was found to be significant. Hence, we can say that Delta Airlines only learns from the scheduled flight network and reduces the length of selected SBT during the feedback loop, for flights which are usually covered by the patterns found using belief-i.

Interestingly, in case of low-cost carriers (LCC) the coefficients of the features corre-

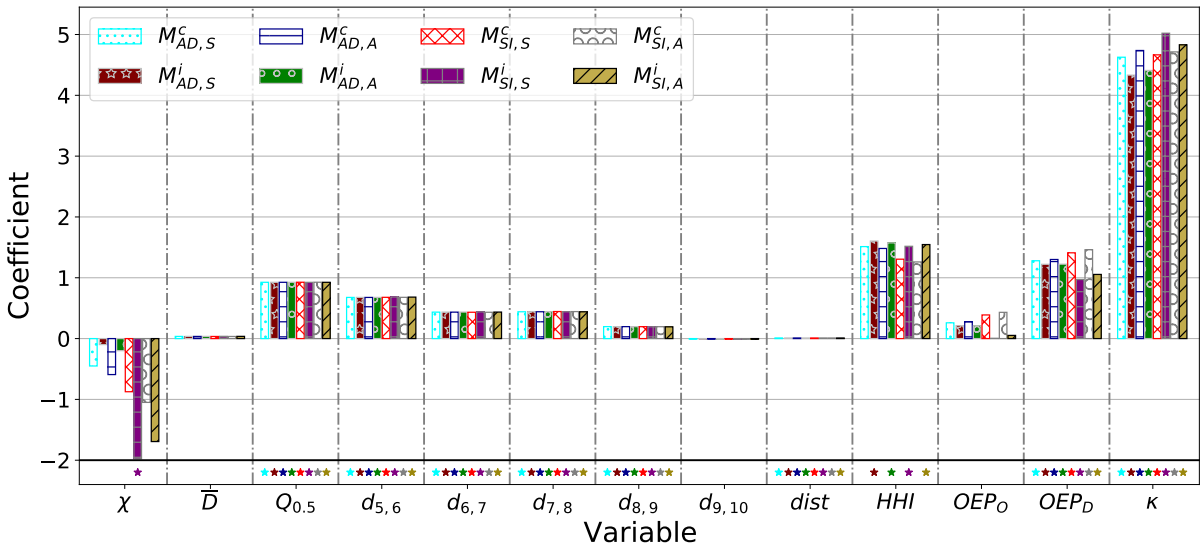


Fig. 6.11. Estimation results of 8 different multiple regression models for the dataset of only Delta airline's flights (number of observations is 2223). The significant coefficients with p-values less that 0.001 are highlighted with a star symbol below. The observed R-squared value for all models is 0.997.

sponding to patterns under belief-c, i.e., $\chi^c_{SI,S}$ and $\chi^c_{SI,A}$, are found to be significant (see Figure 6.12). This is opposite to the behavior of American Airlines, a legacy carrier. One of the reasons for this exception could be the mode of operation of both categories of airlines. That is, American Airlines follows a 'hub and spoke' model, while LCC operate on a 'point to point' connection model. This is supported by the fact that with prior belief-i, SIMP commonly finds star-shaped patterns, where one vertex acts as a hub and is connected to other vertices acting as spokes. In contrast, with belief-c, SIMP usually finds a highly connected set of vertices, where a vertex is directly connected to a high number of other vertices representing a 'point to point' connection model.

Finally, we conclude from the above that the process of selecting the SBT appears to be associated with the SIMP-based variables. Since we are not affirmative whether airliners take into account the knowledge of SIMPs, it is worth exploring the '*what if*' scenarios. For example, investigating what are the benefits associated, if an airline may have altered the selected SBT based on the insight provided by SIMPs. The benefits can be anything, even subjected to an airline's preference. We know that the Federal

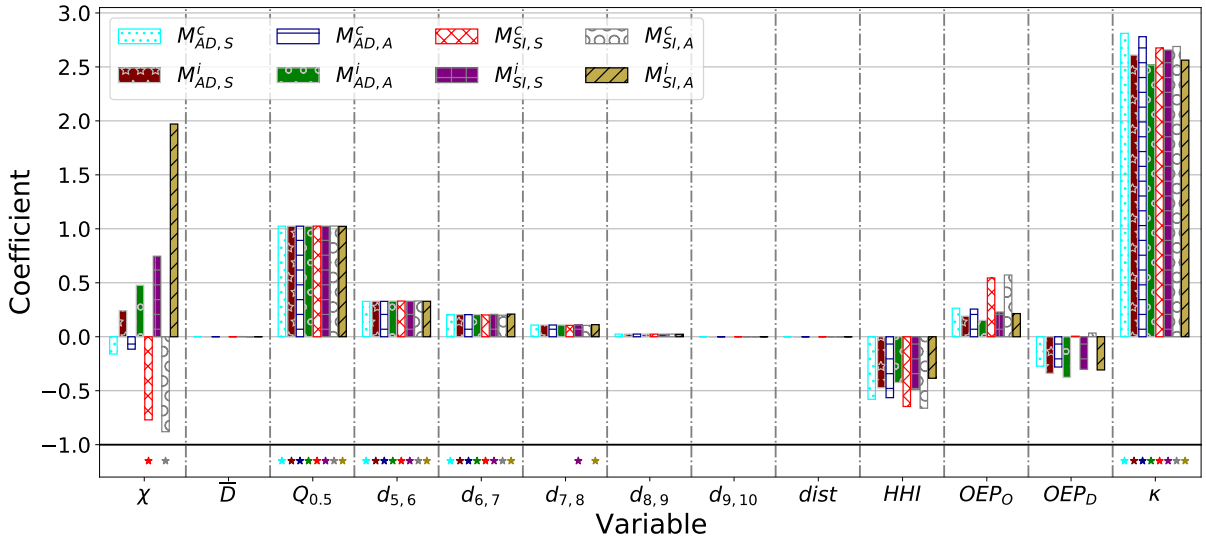


Fig. 6.12. Estimation results of 8 different multiple regression models for the dataset of only Low-Cost Carrier's flights (number of observations is 3817). The significant coefficients with p-values less that 0.001 are highlighted with a star symbol below. The observed R-squared value for all models is 0.998.

Aviation Administration's (FAA) aspiration for Destination 2025 plan is to improve flight predictability in an airline network [Hao et al., 2014a]. Therefore, it will be interesting to explore this aspect using SIMPs. Due to unavailability of domain knowledge about the exact process of SBT selection, evaluating the benefits of using SIMP-based knowledge is beyond the scope of this thesis.

6.3.4 Discussion on SBT selection

In the study of SBT selection, we investigated different airliners' behavior for selecting the SBT for a flight. However, it may be noted that selecting an optimal SBT is a complex process as airliners face a trade-off, where:

1. selecting a larger SBT implies
   - higher fuel loading & consumption,
   - higher travelling time for a passenger,
   - higher negative delays with high airport charges on arriving earlier than scheduled,

+ better reliability, on-time performance, less positive delays, lower missed connections by a passenger.

2. selecting a smaller SBT implies

- high likelihood of positive delays,
- lower reliability & on-time performance,
+ lower fuel loading & consumption,
- high possibility of fuel exhaustion,
+ lower travelling time for a passenger.

Based on the above trade-off of selecting an optimal SBT, an airliner may target more than one objectives to optimize. In a future scope of this work, it will be interesting to learn how the SIMPs can better address all or most of the objectives targeted by an airliner during the process of SBT selection. However, a critical challenge in this further study is the availability of airline-specific data and domain knowledge for the process followed for SBT selection. Nevertheless, the SIMPs may prove to be a tool for the airliners towards efficient SBT selection.

6.4 Predicting Delays in an Airline Network

In this section, we investigate the potential of SIMP-based features for the prediction of delays in an airline network. In the previous sections, we found that the SIMPs can be used to analyze scheduled and actual flight networks to find regions of surprisingly high traffic volume than expected. We observed that in an actual flight networks, SIMPs have a relatively high recall and precision in finding NAS delayed flights. We also found that SIMPs are associated with SBTs. From these discoveries, we now explore the possibility of using SIMPs to classify and predict if a flight will be delayed or not.

6.4.1 Preprocessing and Feature Construction

In this study, we use the BTS on-time performance data of US Airlines for three different years, 2016-2018, and convert the date-time features such as arrival time and departure time (both scheduled and actual) into Pacific timezone. Although the classification of delayed flights may depend on a large number of features, we only consider features nec-

essary to describe a flight. The 'base' set of seven necessary features which may describe a flight include: the day of the week, reporting airline, origin, destination, departure block hour, arrival block hour and distance group. Although, month is also a feature which is necessary to describe a flight, we exclude it from the 'base' set as we will build independent models for each month. The reason to using these features is that we only aim to investigate the SIMP-based features' potential for delay classification. It may be noted that building a robust and efficient prediction model is outside the scope of this study. Without loss of generality, we remove all the instances of diverted and cancelled flights.

To discover SIMP patterns in an airline network, we create the scheduled flight network using the scheduled departure and arrival times of flights, and similar to previous section create 22 independent snapshots of each day. Each snapshot is of one-hour duration from 0300 hours to 0100 hours of the following day. In each snapshot, we discover (at most) the top 5 interesting patterns using the SIMP algorithm. Since we cannot use the actual flight network details to classify flights in a real-world scenario, we only consider the scheduled flight network. Thus, for each of the two types of belief, i.e., belief-c and belief-i, and two different formulations of interestingness, i.e., AD and SI, we construct 6 features. The features are:

1. $isSubInteresting$: Binary, where a value 1 is assigned if a flight is covered by any of the top 5 interesting patterns, otherwise 0,
2. $\#SubInteresting$: Non-negative integer, specifies the number of patterns that covers a flight,
3. R_m: Non-negative integer, indicates the minimum rank of all patterns covering a flight, where rank is the iteration at which the pattern is discovered by the SIMP algorithm,
4. $\overline{\#inNodes}$: Non-negative real number, indicates the average number of nodes with non-zero in-degree in the patterns where a flight is covered,
5. $\overline{\#outNodes}$: Non-negative real number, indicates the average number of nodes with non-zero out-degree in the patterns where a flight is covered,

6. $\overline{\rho}$: Non-negative real number, specifies the average of average densities of patterns covering a flight, where the average density of a pattern is the ratio of the number of edges in a multigraph pattern to the maximum possible number of edges in a simple directed graph equivalent of the same set of vertices.

The first three features quantify the importance of discovering a flight in a region with surprisingly high traffic volumes, while the latter three features quantify the observed traffic volume in the patterns. As discussed earlier, heavy traffic is one of the major cause of NAS delays. Thus, using SIMP-based features, we quantify the unforeseen high traffic volumes in an airline network and investigate these features' relevance in predicting flight delays.

Given that we have four different configurations to run the SIMP algorithm, we name the set of six features constructed using the patterns in each case as follows. For configuration using AD based interestingness, we call the set of features as set 'AD-c' for prior belief-c and set 'AD-i' for prior belief-i. Similarly, for SI based interestingness, we call the set of features 'SI-c' and 'SI-i' for prior belief-c and belief-i, respectively. We perform one-hot encoding for 'base' set features as all features are categorical. Since we consider 298 airports for all three years of data with 12 unique airlines and distances between airports grouped into 11 categories, we get a total of 670 features after one-hot encoding of 'base' set features.

6.4.2 Experiments

After constructing the features, we now build the classifiers. We build 12 independent classifiers for each month, where we train the classifiers using the data of the year 2016, validate on data of the year 2017 to select the best hyperparameters, and test the classifiers on the data of the year 2018. Although such models may not excel at predicting delayed flights, we here investigate whether SIMPs are relevant for learning the delayed flights in the network.

In the experiments, we will find the answer to the following questions:

1. Can the number of misclassifications (false positives and false negatives) be reduced using SIMP-based features?

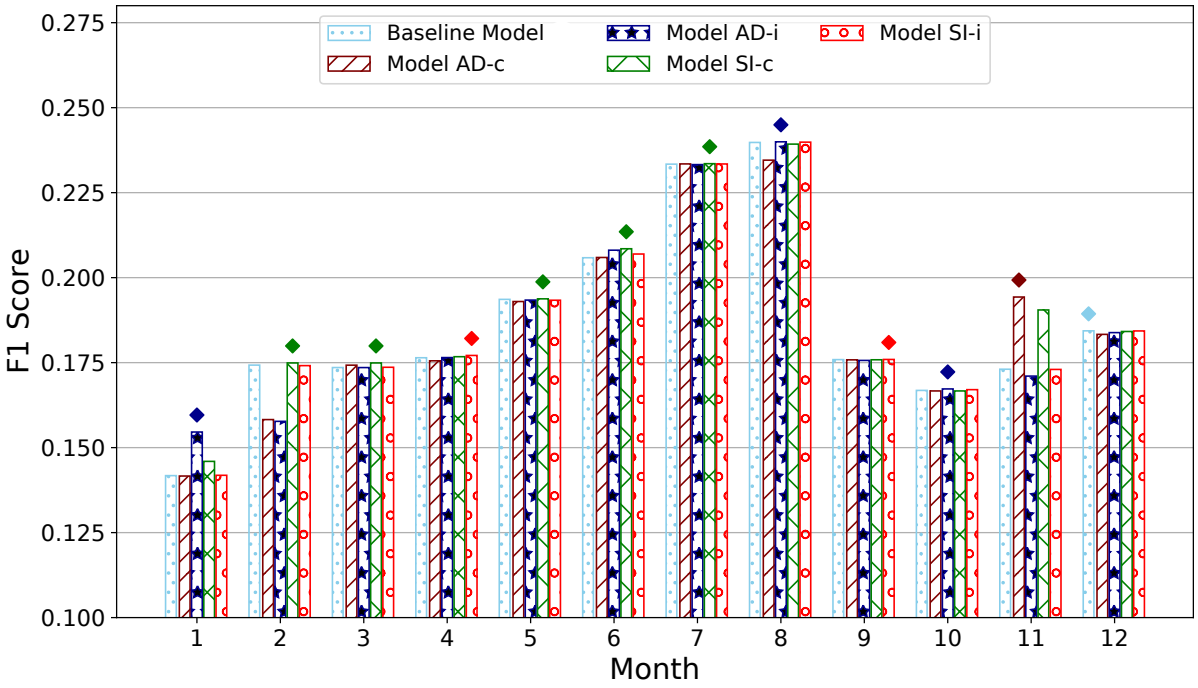


Fig. 6.13. F1 score of each classifier built for different months when the target feature is the overall delayed flights, the best model for each month is highlighted with a diamond symbol on the top.

2. How relevant are the SIMP-based features for the prediction of delayed flights?
3. Which configuration of SIMP, among the four contributes the most in learning flight delays?

For this study, we choose decision tree-based classifiers as they are easy to interpret and provides the importance of features in prediction. We now build five independent models for each month using a different set of features, and hence the nomenclature used corresponds to each set of features used. However, in models using sets 'AD-c', 'AD-i', 'SI-c' and 'SI-i', we also include the features of the 'base' set. Thus, the model learnt using only the 'base' set can be considered a baseline model.

6.4.2.1 First Case: Overall delayed flights

In the first case, we consider overall delay, which is binary—as target for prediction—where the positive class includes the flights which are delayed by at least 15 minutes and arrival delay is larger than the departure delay. Thus, we aim to learn about the flights

Table 6.1. Shown are the median feature importance (FI) rank in a decision tree classifier and the median Chi^2 rank of the SIMP-based features for the different models, considering the target feature of overall delayed flights. Total number of features considered in each model are 676.

Model	$isSubInteresting$		$\#SubInteresting$		R_m		$\overline{\#inNodes}$		$\overline{\#outNodes}$		$\overline{\rho}$	
	FI	χ^2	FI	χ^2	FI	χ^2	FI	χ^2	FI	χ^2	FI	χ^2
AD-c	39	75	25	24	34	18	25	2	22	3	12	56
AD-i	44	57	35	21	30	8	28	2	26	3	25	37
SI-c	48	61	36	38	32	12	21	3	20	3	25	33
SI-i	102	364	103	445	56	102	63	76	52	66	60	254

which got further delayed during the airborne phase. This class may include flights which are directly or indirectly delayed due to high traffic congestion. Both types of errors, type-I (non-delayed classified as delayed) and type-II (delayed flight classified as not delayed) are critical and need to be avoided as much as possible. We evaluate the models based on the F1-score of the positive class on the test data. In Figure 6.13, we observe that the models built using the SIMP-based features are better than the baseline model 11 out of 12 times. It is also observed that the difference in the performance measure is not significant in most of the cases, but for January and November, a clear improvement can be observed for models 'AD-c' and 'AD-i' compared to the baseline, respectively.

Further, in Table 6.1, we investigate the relevance of SIMP-based features by observing the rank at which the features lie based on feature importance (FI) in a decision tree based classifier. At the same time, we also evaluate the dependency of the proposed features on the target feature. For this, we perform the Chi^2 test and show the rank of the variables based on Chi^2 statistics. Here, a lower rank feature is better than the features with higher rank and have a higher dependency on the target feature.

The FI ranks of the SIMP-based features in Table 6.1 are less than 50 for each model except 'SI-i'. While in model 'SI-i', the FI ranks are between 50 to 103. Since the total features in each model are 676, the ranks below 50 can be considered as good features. Similarly, the SIMP-based features in model 'SI-i' on the basis on FI ranks

can be considered as mediocre features. It is of note that the FI ranks are dependent on the final model, i.e., the one which performs best on the validation data. Thus, for completeness, the ranks based on the Chi^2 statistics are also shown which indicates the relevance of the feature in explaining the target feature based on the training data. The Chi^2 ranks of SIMP-based features in each model, except 'SI-i', are less than 75 (with maximum possible rank of 676). This indicates that the features can be considered as good in explaining the target feature. In these three models, the rank of features R_m, $\overline{\#inNodes}$ and $\overline{\#outNodes}$ are significantly better with maximum observed rank of 18. This suggests that SIMP-based features explain the target much better than most of the 'base' set features.

It shall also be noted that the features $isSubInteresting$, $\#SubInteresting$ and R_m have dependencies on each other, as when R_m is 0 for any flight then the other two features value would also be 0 as none of the SIMPs cover those flights. Thus, one feature is found to be relatively more relevant than the other two features. In model 'AD-c', 'AD-i' and 'SI-c', the SIMP-based features' ranks show that the SIMPs are relevant in learning about delayed flights in an airline network. While on the other hand, in the case of Model 'SI-i' the SIMP-based patterns are little relevant due to the observed high ranks. This could be because of the fact that for SI-based interestingness with prior belief-i, we discover patterns of smaller size (as shown in Chapter 5), where fewer flights are covered. In other words, for most flights, the value of these features is 0. This limitation can be overcome by increasing the number of patterns discovered.

6.4.2.2 Second Case: NAS delayed flights

In this case, we consider NAS delay, which is binary—as target for prediction—where the positive class implies the flights that are NAS delayed by at least 15 minutes. Thus, we aim to learn about flights where high traffic congestion is a direct factor of delay. Similarly, in this case, we also evaluate a classifier's performance using the F1-score of the positive class on the test data. In Figure 6.14, we observe that the models built using the SIMP-based features are better than the baseline model 10 out of 12 times. We also observe that Model 'AD-c' performed the best for 7 out of 12 months. This indicates that

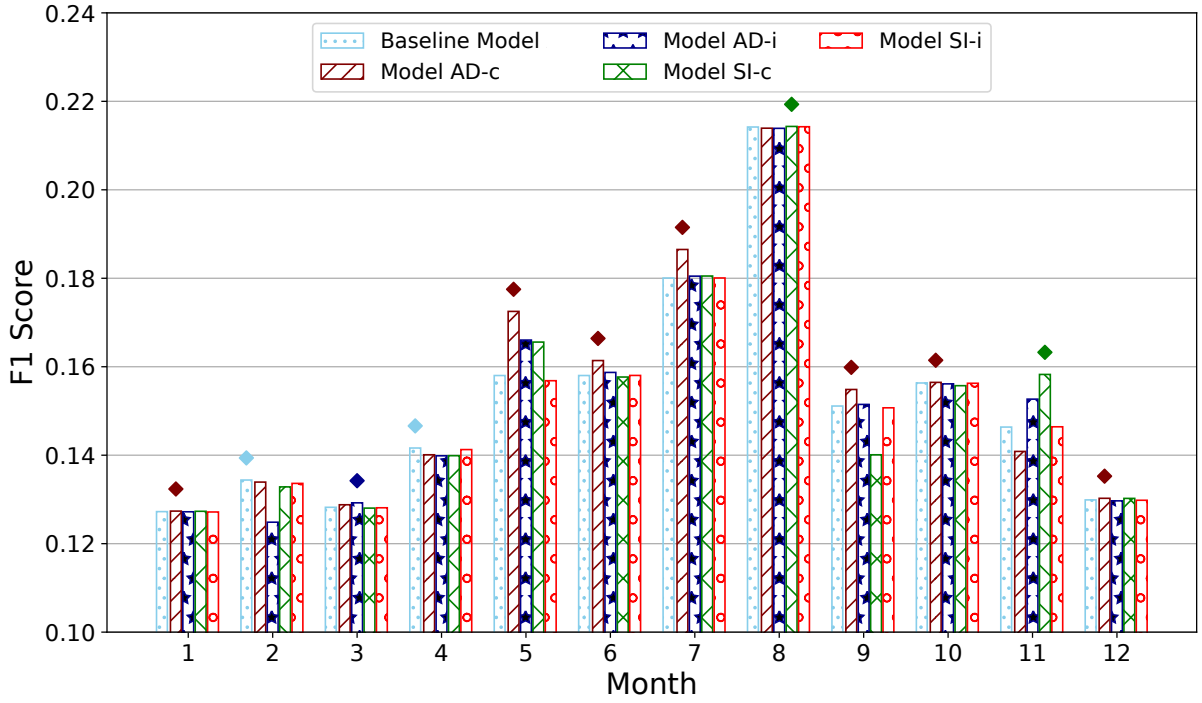


Fig. 6.14. F1 score of each classifier built for different months when the target feature is the NAS delay. The best model for each month is highlighted with a diamond symbol.

Table 6.2. Shown are the median feature importance (FI) rank in a decision tree classifier and the median Chi^2 rank of the SIMP-based features for the different models, considering the target feature of NAS delay. Total number of features considered in each model are 676.

Model	$isSubInteresting$		$\#SubInteresting$		R_m		$\overline{\#inNodes}$		$\overline{\#outNodes}$		$\overline{\rho}$	
	FI	χ^2	FI	χ^2	FI	χ^2	FI	χ^2	FI	χ^2	FI	χ^2
AD-c	60	50	43	17	48	9	33	2	43	3	36	41
AD-i	51	55	51	28	47	11	38	7	39	5	33	45
SI-c	67	22	56	9	45	1	38	2	33	3	37	11
SI-i	116	72	127	48	97	8	89	16	68	24	76	48

with belief-c and AD based interestingness, the SIMPs discovered better aids in learning factors causing NAS delays in the network. We also observed that Model 'SI-c' performed the best in 2 out of 12 months, which suggests that prior belief-c effectively captures the NAS delay instances in the network.

Interestingly, in Table 6.2, we observed that now density of a pattern $\overline{\rho}$ is among the top 3 relevant features out of the 6 proposed, in terms of FI ranks. The other two SIMP-based features are $\overline{\#inNodes}$ and $\overline{\#outNodes}$. This indicates that NAS delays are dependent on the density of traffic volumes in a region identified by SIMPs. Notably, SIMPs are patterns with surprisingly high density, i.e., relatively large number of flights are observed than expected. This confirms our motivation that NAS delays are relatively more likely in regions with unexpectedly high traffic volumes. The rest of the observations, in this case, are similar to the observations made in the previous case.

6.4.3 Discussion

The problem of delay prediction in an airline network is tricky, as there are many factors, including human, machine and nature-related, which can cause delays in the network. On top of it, the airliners also make several changes in the schedule, both strategic and tactical, to overcome delays. The strategic decisions are made for long term goals (made 3-6 months in advance), including changes in the schedule of a flight (change in SBT included), whereas tactical decisions are for short term goals (made on the day or 1 week in advance), including aircraft swapping, changing the priority of a flight by air traffic controllers to depart & land, as so on. These are some factors which make learning delay patterns in the network excessively difficult. Hence, the prediction of delays is an overwhelmingly tricky problem.

In this study, we proposed one way to quantify SIMPs in terms of features to predict delays. Although we cannot significantly improve the performance measure of a classifier, we did realise that SIMPs–with different prior beliefs–can be a potential tool to learn the delay patterns in a network for prediction as we believe that there can be other ways to efficiently construct explainable features, such as a way proposed by Bringmann et al. [2011]. The authors proposed that each pattern can be a feature for a pattern-based

classification, followed by either model-independent feature selection or guided by the model.

We also argue that other data, such as weather and aircraft, can be used together with the on-time performance data to build a robust prediction model. However, this may require to build a scalable prediction model to handle a large volume of data. We consider this as one of the future directions of our study.

6.5 Exploring an Evolving Airline Network

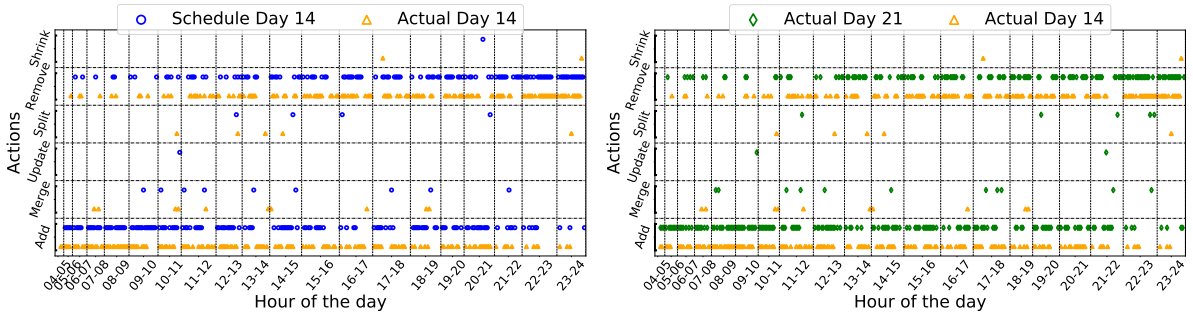
In this section, we present a case study on the US flight network[6.5] to demonstrate an application of DSSG and DSIMP algorithm presented in Chapter 4 and 5. Flight networks are typical examples of dynamic graphs that one would like to analyze on the fly, e.g., to detect and monitor delays as early as possible. Since DSIMP and DSSG are similar algorithms, we only show the application of DSSG in this case study.

6.5.1 Dataset

We use the scheduled and actual flight operating data for the month of January 2017, with 298 airports (considered as vertices) and 450 017 flights operated in that month. The dataset has features such as scheduled departure & arrival time and actual departure & arrival time, for each flight. Using these features we create two types of networks: 1) in a scheduled flight network, a directed edge for a given time interval is included from origin to destination airport if at least one flight was scheduled to depart or arrive in that interval; 2) in an actual flight network, a directed edge is included between two airports if at least one flight actually departed or arrived in that interval.

For either type of network, we create 31 independent instances, one for each day of January 2017. Each network is segmented into 20 sequential snapshots, or states, of one hour each (from 0400 hours to 2400 hrs, all converted to UTC -7). The motivation behind choosing one hour as the length of a snapshot is that airliners manage their operations in blocks of one-hour duration each. For simplicity, we do not consider cancelled flights in this case study.

[6.5] source: https://www.transtats.bts.gov/



(a) Scheduled vs actual operations of day 14.

(b) Actual operations of day 14 vs day 21.

Fig. 6.15. Actions (and their types) throughout the day as found by DSSG on the scheduled and actual flight networks of two days.

6.5.2 Approach

We use DSSG to summarize both the scheduled and actual flight network independently. In both the cases, we assume the analyst to have a prior belief on the number of routes scheduled to be operated from each airport in the initial snapshot (i.e., the total number of airports from where at least one flight is arriving and the total number of airports to where at least one fight is departing). We then inspect the resulting summaries.

6.5.3 Summaries and their comparisons

As the data is large and dynamic, visualizing all patterns or the complete summary at once is not practical. Instead, Figure 6.15 visualizes the sequence of actions identified by our method on a given flight network, to provide a high-level overview—or *fingerprint*—of the summary. Such fingerprints can then be compared to spot deviations between the scheduled and actual dynamic networks.

An analyst could investigate the discovered patterns (as shown in Section 4.5.5), but here we first investigate the differences between the obtained summaries, to learn about unexpected events (here: delays) causing the observed network to differ from the expected network. For illustrative purposes, we use the scheduled network of day 14, and actual networks of days 14 and 21.

Inspecting the fingerprints in Figure 6.15 shows that the actual flight network of day 14 behaves differently from both the scheduled flight network of day 14 (Figure 6.15a)

and the actual flight network of the same day one week later (Figure 6.15b). For example, in the initial snapshot (0400–0500hrs) in Figure 6.15a, the prior distribution sufficiently described the scheduled flight network of day 14, and hence no new patterns are discovered. In the actual flight network of that day, however, two patterns are discovered for that snapshot. A closer look at the data reveals that this is caused by flights that operated either ahead of time or delayed. In Figure 6.15b, similar observations can be made for the actual flight networks for two days precisely one week apart. To further investigate the causes of deviations, an analyst could inspect the patterns and actions. DSSG provides a sequence of actions (descending by $\mathcal{IG}$) that an analyst could learn from, mainly when supported by an environment for interactive data and pattern exploration.

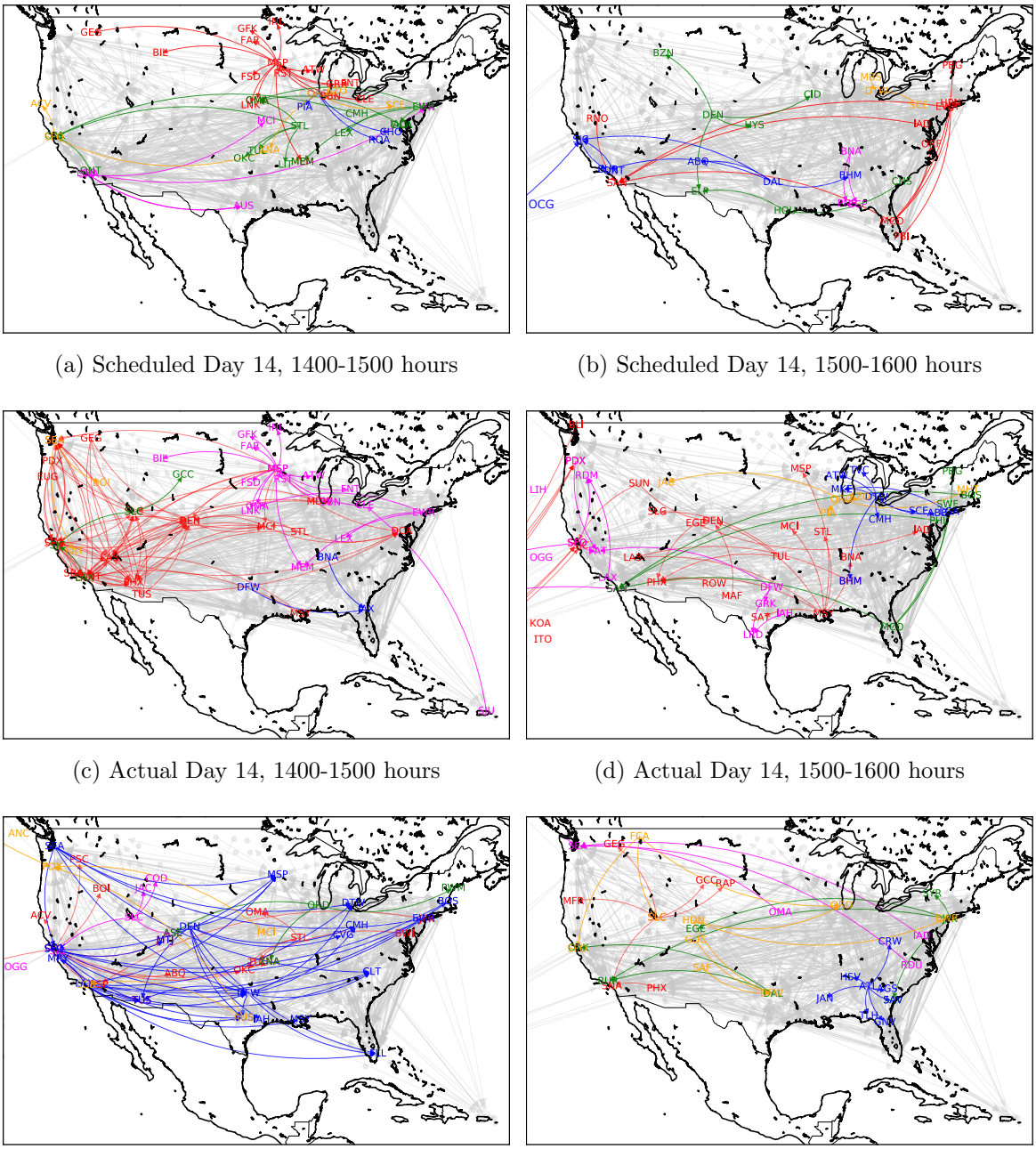
6.5.4 Inspecting patterns

To further understand the differences between the flight networks, we consider two typical block hours, i.e., 1400-1500 and 1500-1600 hours. Figure 6.16 shows the top 5 patterns[6.6] concerning information content ($\mathcal{IC}$) and for the same three different networks as above. Note that this means that we only show patterns that are newly discovered or revised in the current state.

From Figure 6.16a we observe that, for the *scheduled flight network* of day 14 during 1400-1500 hours, four out of the five patterns are star-shaped, with hub airports. In the first pattern (shown in red) MSP is the hub, with flights departing for airports such as ATW, LNK and MEM. Similarly, patterns 2 (magenta) and 4 (blue) have SNA and ORD as hubs, respectively. Pattern 3 (green) has STL and EWR as hubs, where flights are departing, and two other airports, OMA and OAK, where flights are arriving. These patterns indicate that a large number of flights are scheduled to depart from hubs like MSP, SNA, ORD and STL, while flights are expected to arrive at OAK and OMA. Finally, pattern 5 is a connected set of airports including SFO, XNA, ORD and SCE.

In the *actual flight network* for the same timeslot in Figure 6.16c, the most informative pattern is a set of densely connected airports including SLC, DEN, LAS, and SEA (shown

[6.6] An analyst could, of course, visualize all actions or patterns in the summaries, but we only show the top 5 patterns for reasons of space and clarity.



(a) Scheduled Day 14, 1400-1500 hours

(b) Scheduled Day 14, 1500-1600 hours

(c) Actual Day 14, 1400-1500 hours

(d) Actual Day 14, 1500-1600 hours

(e) Actual Day 21, 1400-1500 hours

(f) Actual Day 21, 1500-1600 hours

Fig. 6.16. The top 5 patterns with regard to information content discovered from each respective flight network. Color coding: the pattern with highest information content is shown in red, followed in order by magenta, green, blue, and orange. Labels indicate airport codes.

in red). The second pattern (magenta) is similar to the most informative pattern found in the scheduled network (red in Figure 6.16a), with MSP as a hub. Patterns 3, 4, and 5 are also star-shaped, with hubs SLC, JAX, and SEA, respectively. Upon investigating the underlying data, we find that patterns 1 and 3 comprise flights having a combined positive delay (flights departing and/or arriving late) of 1083 minutes and 23 minutes, respectively. This is a relevant discovery, as 1083 minutes is a considerable combined delay, and pattern 1 was not found in the scheduled data. For pattern 2, which we did find in the scheduled network, no positive delay is observed (instead we find a combined negative 'delay' of roughly 9 minutes, which is very moderate). For patterns 4 and 5 negative delays are observed. Similar observations can be made for the block hour in Figures 6.16c–6.16d.

The fingerprints of Figure 6.15b already suggested that the actual flight networks of days 14 and 21 differ, and this is confirmed by the different top 5 patterns shown in Figures 6.16e-6.16f. Interestingly, none of these patterns is present in either the scheduled or actual flight network of day 14, and these patterns are also found to correspond to substantial positive and/or negative delays.

Together, these observations indicate that by comparing the summaries and patterns discovered by DSSG, an analyst can learn about sets of connected airports where structural operational deviations from the schedule occurred, which often resulted in delays. As such, this case study served to illustrate how our approach could be used in a real-world scenario where an online and incremental analysis of structural changes in dynamic graphs can render valuable insights.

6.6 Summary

In this chapter, we presented potential application of the proposed notion of subjectively interesting patterns for networks and the algorithms presented in this thesis, on a real-world example of Airline Operations. Within airline operations, the learnings from the subjectively interesting patterns can be used to analyse and understand the network's behavior, both in case of static views and in case of an evolving instance of a graph. We demonstrated that subjectively interesting patterns discovered using SIMP algorithm can

be used by an airliner to select an SBT for a flight effectively and can also be used in building delay prediction models. Finally, we also presented an example of how DSSG can be used to analyse the structural changes in a dynamic airline network. The application of SIMPs may not be limited to the domains discussed in this chapter, but could also be used in other domains such as crew scheduling, baggage handling, etc.

Chapter 7

Conclusions and Future Scope

7.1 Conclusion

In this thesis, we addressed the concept of subjective interestingness for different graph types, including static multigraphs, evolving simple graphs, and evolving multigraphs. This thesis is built on the premise that subjective interestingness measures are instrumental and critical in data mining to quantify the actual usefulness of a pattern to an analyst. Studying a real-world network by employing subjective interestingness measures in graph mining methods was found to reveal informative and surprising patterns.

It was identified that many real-world scenarios can be best represented as multigraph or dynamic graph, apart from simple, attributed & weighted graphs. However, in the literature of graph mining methods, it was observed that most graph mining methods focused on developing a measure with a fixed belief about the expected structure of a subgraph pattern. These measures do not consider the analyst's prior belief and are limited in quantifying a critical aspect of interestingness, i.e., usefulness to an analyst. This led to the introduction of subjective interestingness measures. In the past, the notion of subjectively interesting patterns in a network has been limitedly studied, i.e., in static simple graphs [van Leeuwen et al., 2016] and attributed graphs [Bendimerad et al., 2020]. With the above observations, two main research gaps were highlighted in this thesis. These gaps point to the fact that the concept of subjective interestingness has not been developed for static multigraphs and dynamic graphs. Given these research gaps, the thesis' goal was to develop graph mining methods using the notion of subjective interestingness to discover patterns in static multigraphs, evolving simple graphs, and evolving multigraphs.

In Chapter 2, a comprehensive literature review was presented, and the research gaps have been highlighted. In Chapters 3-5, we built on the notion of subjective interestingness as proposed in the FORSIED framework [De Bie, 2011b] to address the identified research gaps. In these chapters, an analyst's prior beliefs are modelled as probability distributions

following the FORSIED framework.

In Chapter 3, an approach defining subjective interestingness of multigraph patterns has been proposed. Subsequently, to iteratively discover multigraph patterns, a greedy based heuristic algorithm named SIMP has been proposed. In this algorithm, the proposed interestingness measure emphasizes the *novelty* and *surprisingness* of a pattern by considering the analyst's prior beliefs through the proposed Aggregate Deviation measure. Simultaneously, the *conciseness* of a pattern is ensured by a parameter used to compute Description Length of a pattern. To emphasize *pecularity* and *diversity* of the set of patterns discovered, it was suggested to update the background probability distributions after each pattern is discovered. This aids the proposed subjective interestingness measure to avoid discovering the same patterns again. Finally, the efficacy of the proposed algorithm, over the above-discussed features, has been demonstrated with extensive experiments performed on several synthetic and real-world data, where it is natural to use a multigraph for modelling.

In Chapter 4 and 5, a novel framework to summarize dynamic data was proposed, and an instantiation of the framework was presented for evolving simple graphs and evolving multigraphs. The proposed framework is built on the concept of subjective interestingness [De Bie, 2011b], but also draws its inspiration from the minimum description length (MDL) principle [Grünwald, 2007]. This inspiration led to the introduction of the novel *information gain* measure, with which the proposed approach discovers *concise* and *compact* summaries without having to decide on the number of patterns in advance. Further, we capture the informative changes in an evolving network employing the proposed *atomic changes* which can also be *easily interpreted by an analyst.* This marked the first attempt to combine data mining approaches based on subjective interestingness using the MaxEnt principle with pattern-based summarization using the MDL principle. An instantiation of the proposed framework was discussed in Chapter 4 for evolving simple graphs (algorithm was termed DSSG) and in Chapter 5 for evolving multigraphs (algorithm was termed DSIMP). The experiments in both chapters demonstrated the efficacy of the proposed respective algorithms on discovering meaningful evolutions while summarizing the data.

Finally, Chapter 6 presented potential real-world applications of the proposed methods in airline operations. First, a case study was presented on a static airline network where using the SIMP algorithm, patterns with unexpectedly high traffic volumes were discovered. These patterns were found to constitute a large number of NAS delayed flights. In the second and third part of this chapter, the application of SIMPs in the process of SBT selection and delay prediction was discussed, respectively. Finally, a dedicated case study was presented to demonstrate the potential usefulness of DSSG in analyzing evolving network *on-the-fly*.

This thesis highlighted the importance of an analyst's intervention in a data mining process with the above contributions. In the proposed methods, the analyst provides the beliefs about the data before starting the mining process and communicating with the method through the process in a dynamic setting. Thus, this thesis addressed the limitation of most data mining methods, where both analyst and the data itself drives the proposed methods. However, it is still challenging to model every type of prior belief that an analyst may have. Here, only a few limited types of prior beliefs were discussed.

One of the limitation of the proposed self-information based measure for multigraphs is that it is only applicable when prior beliefs result in the background distribution that is represented in the natural form of geometric distributions. We observed this in case of belief-m for multigraphs. However, aggregate deviation based measure can be used in such cases.

In DSSG and DSIMP, during the process of learning and forgetting the patterns at different time snapshots, re-discovery of a pattern was not evaluated. This is useful to know when a pattern appears sporadically and is discovered by the algorithm multiple times; here communicating re-discovery of a pattern might require less description length than communicating it as a new pattern. Hence, this information is useful to an analyst when periodicity of a pattern is important.

A common significant shortcoming of the proposed algorithms is the observed high runtimes compared to other state-of-the-art methods. Although several steps are involved in each iteration of the algorithm's execution, a considerable factor responsible is the evaluation of all possible seed graphs based on the neighborhood subgraph's interest-

ingness value. This evaluation is required each time to determine top-k seeds and execute k independent hill-climber searches for discovery of a pattern. Notably, in a later stage, the computational complexity of each algorithm was reduced significantly by eliminating redundant computations, and parallelizing the top-k seed evaluations and independent hill-climber search executions for each seed. This led to the runtimes for an experiment even reduced by upto ten times compared to the runtimes reported in this thesis. The source code of the optimized implementation of the three proposed algorithms along with the SSG algorithm [van Leeuwen et al., 2016] in Python3 has been published on Zenodo [Kapoor and van Leeuwen, 2021].

7.2 Future Scope

The research presented in this thesis offers several exciting avenues. We identified a few potential future opportunities, for extending the research work carried out in this thesis, which are as follows.

First, from the theoretical point of view, this thesis can be considered a basis towards subjective interestingness-based methods for types of networks that are not considered in this thesis. Potential directions include: 1) subjective interestingness measures for subjectively interesting patterns in multilayer graphs, 2) subjective interestingness based learning methods for evolving attributed, and evolving multilayer graphs. In the context of dynamic graphs, the proposed methods can be further extended to discover periodic and frequent subjectively interesting subgraph patterns.

Second, the proposed novel framework of subjective summarization of sequential data in Chapter 4 can be instantiated for other types of data as well. Some of the cases include evolving attributed graphs, evolving multilayer graphs, and evolving weighted graphs. Apart from the graph data, the framework can naturally be adapted for any rectangular dataset represented in different snapshots, such as tile data which evolves with time.

Third, a possible future direction is to study different types and forms of prior beliefs. As seen in this thesis, the prior beliefs are mostly application-specific and require special attention to model for computation of MaxEnt distribution. Also, depending on the definition of a pattern, the formulation of a subjective interestingness measure changes. This

opens the opportunity to explore different types of patterns according to the requirements in different real-world applications.

Fourth, the applications of the proposed methods in airline domain presented in this thesis can be further studied. This includes studying airline networks using DSSG and DSIMP for SBT selection and delay prediction. Further, it would be intriguing to investigate feature construction and engineering aspects—using the proposed methods—for machine learning applications. It would also be interesting to investigate the advantages of the proposed methods by incorporating domain knowledge and utilizing airline-specific data.

Fifth, from the perspective of an application in airline operations, several other problems can be studied using the methods proposed in this thesis. An example is the crew pairing optimization problem, where the challenge is to assign crews to different flights in an airline's operational network, with minimum associated costs and expenses [Andersson et al., 1998]. Crew pairings are generally modelled using graphs; thus, subjectively interesting patterns can be studied in the context of this problem. A few other similar problems in airline operations include fleet assignment, baggage handling, and maintenance scheduling of aircrafts.

Finally, applications of the proposed methods are not restrictive to the airline domain. It is certainly of interest to study other real-world problems dedicatedly. It would also be valuable to develop a tool based on the proposed methods for interactive visualization and exploration of the knowledge discovered in a future opportunity. This tool would provide a user-friendly platform to model an analyst's prior beliefs and learn informative patterns in the data provided.

Appendix **A**

Proofs from Chapter 3

A.1 Proof of Probability Distribution for Belief-m

The problem of maximizing entropy under the user's belief about the number of edges per vertex and the number of neighbors per vertex is given as

$$\underset{P(\mathbf{A})}{\operatorname{argmax}} - \sum_{\mathbf{A}\in\mathbb{N}_0^{n\times n}} P(\mathbf{A})\log(P(\mathbf{A})) \tag{A.1.1}$$

$$\text{s.t.} \sum_{\mathbf{A}\in\mathbb{N}_0^{n\times n}} P(\mathbf{A})\sum_v a_{u,v} = d_u^r; \sum_{\mathbf{A}\in\mathbb{N}_0^{n\times n}} P(\mathbf{A})\sum_u a_{u,v} = d_v^c, \tag{A.1.2}$$

$$\sum_{\mathbf{A}\in\mathbb{N}_0^{n\times n}} P(\mathbf{A})\sum_v 1_{a_{u,v}\neq 0} = m_u^r; \sum_{\mathbf{A}\in\mathbb{N}_0^{n\times n}} P(\mathbf{A})\sum_u 1_{a_{u,v}\neq 0} = m_v^c \tag{A.1.3}$$

$$\sum_{\mathbf{A}\in\mathbb{N}_0^{n\times n}} P(\mathbf{A}) = 1 \tag{A.1.4}$$

Since this optimization problem is convex, we solve it using convex optimization methods [Boyd and Vandenberghe, 2004]. Let us introduce the Lagrangian multipliers λ_i^r & λ_i^c for constraints in Eq A.1.2, μ_i^r & μ_i^c for constraints in Eq. A.1.3; and ψ for constraint A.1.4. The Lagrangian of the Problem A.1.1-A.1.4 is now given by

$$\begin{aligned}
&\mathcal{L}(P(\mathbf{A}), \boldsymbol{\lambda^r}, \boldsymbol{\lambda^c}, \boldsymbol{\mu^r}, \boldsymbol{\mu^c}, \psi) = \\
&-\sum_{\mathbf{A}} P(\mathbf{A})\log P(\mathbf{A}) + \sum_u \lambda_u^r \left(\sum_{\mathbf{A}\in\mathbb{N}_0^{n\times n}} P(\mathbf{A})\sum_v a_{u,v} - d_u^r\right) \\
&+\sum_v \lambda_v^c \left(\sum_{\mathbf{A}\in\mathbb{N}_0^{n\times n}} P(\mathbf{A})\sum_u a_{u,v} - d_v^c\right) + \sum_u \mu_u^r \left(\sum_{\mathbf{A}\in\mathbb{N}_0^{n\times n}} P(\mathbf{A})\sum_v 1_{a_{u,v}} - m_u^r\right) \\
&+\sum_v \mu_v^c \left(\sum_{\mathbf{A}\in\mathbb{N}_0^{n\times n}} P(\mathbf{A})\sum_u 1_{a_{u,v}} - m_v^c\right) + \psi\left(\sum_{\mathbf{A}\in\mathbb{N}_0^{n\times n}} P(\mathbf{A}) - 1\right)
\end{aligned} \tag{A.1.5}$$

The optimality conditions are achieved by equating the derivative of Equation A.1 w.r.t. $P(\mathbf{A})$ to 0. Hence, we get

$$P(\mathbf{A}) = \frac{1}{Z(\boldsymbol{\lambda^r}, \boldsymbol{\lambda^c}, \boldsymbol{\mu^r}, \boldsymbol{\mu^c})} \exp\left(\sum_{u,v} a_{u,v}(\lambda_u^r + \lambda_v^c) + \sum_{u,v} 1_{a_{u,v}}(\mu_u^r + \mu_v^c)\right), \tag{A.1.6}$$

where $Z(\boldsymbol{\lambda^r}, \boldsymbol{\lambda^c}, \boldsymbol{\mu^r}, \boldsymbol{\mu^c}) = \exp(1-\psi)$ is a partition function. De Bie [2011b] suggested that the choice of partition function is such to ensure the normalization constraint A.1.4. Similarly, here the partition function is also found to be the product of individual partition function represented by unique pair u and v, i.e., $Z(\boldsymbol{\lambda^r}, \boldsymbol{\lambda^c}, \boldsymbol{\mu^r}, \boldsymbol{\mu^c}) = \prod_{u,v} Z(\lambda_u^r, \lambda_v^c, \mu_u^r, \mu_v^c)$. Therefore, Equation A.1.6 now becomes

$$P(\mathbf{A}) = \prod_{u,v} \frac{1}{Z(\lambda_u^r, \lambda_v^c, \mu_u^r, \mu_v^c)} \exp\left(\lambda_u^r + \lambda_v^c\right)^{a_{u,v}} \cdot \exp\left(\mu_u^r + \mu_v^c\right))^{1_{a_{u,v}}} . \tag{A.1.7}$$

This perfectly aligns with the proposition made by De Bie [2011b], as here also $P(\mathbf{A})$ comes out to be the product of an exponential family distribution. Given the domain of $a_{u,v}$, the partition function is calculated as $Z(\lambda_u^r, \lambda_v^c, \mu_u^r, \mu_v^c) = \sum_{a_{u,v}\in\mathbb{N}_0} \exp(a_{u,v}(\lambda_u^r + \lambda_v^c) + 1_{a_{u,v}}(\mu_u^r + \mu_v^c))$ which results in $Z(\lambda_u^r, \lambda_v^c, \mu_u^r, \mu_v^c) = \frac{1-\exp(\lambda_u^r+\lambda_v^c)(1-\exp(\mu_u^r+\mu_v^c))}{1-\exp(\lambda_u^r+\lambda_v^c)}$ such that $\lambda_u^r + \lambda_v^c < 0$. Finally, from Equations A.1.7 we get

$$P_{u,v}(a_{u,v}) = \frac{\left[1-\exp\left(\lambda_u^r+\lambda_v^c\right)\right]}{\left[1-\exp\left(\lambda_u^r+\lambda_v^c\right)\left(1-\exp\left(\mu_u^r+\mu_v^c\right)\right)\right]} \cdot \exp\left(\lambda_u^r+\lambda_v^c\right)^{a_{u,v}} \cdot \exp\left(\mu_u^r+\mu_v^c\right)^{1_{a_{u,v}}}$$

■

A.2 Proof of Theorem 3.1 (Claim 2)

The Lagrangian of Equations 3.10-3.12 is given as

$$L = \sum_{\mathbf{A}} Q\left(\mathbf{A}\right) \log\left(\frac{Q(\mathbf{A})}{P(\mathbf{A})}\right) + \lambda_H \left(|E'| - \sum_{\mathbf{A}} Q(\mathbf{A}) \sum_{u,v\in W} a_{u,v}\right) + \mu_H \left(1 - \sum_{\mathbf{A}} Q(\mathbf{A})\right) \tag{A.2.8}$$

Thus, upon taking the derivative of L w.r.t. Q, such that $P'(\mathbf{A}) = Q(\mathbf{A})$ at $\frac{\partial L}{\partial Q} = 0$. Then, we get

$$\Rightarrow P'(\mathbf{A}) = \frac{P(\mathbf{A})}{Z'} \prod_{u,v\in W} \exp\left(\lambda_H\right)^{a_{u,v}} \tag{A.2.9}$$

where $Z' = \exp\left(1-\mu_H\right)$ where Z' is a new partition function. Now, using $P(\mathbf{A})$ as given in second part of Theorem 3.1, Equation A.2.9 becomes

$$P'(\mathbf{A}) = \frac{1}{Z'} \prod_{u,v\in W} \exp\left(\lambda_H\right)^{a_{u,v}} \cdot \prod_{u,v} \frac{1-R_{u,v}}{1-R_{u,v}(1-S_{u,v})} R_{u,v}^{a_{u,v}} S_{u,v}^{1_{a_{u,v}}} \tag{A.2.10}$$

Let, $R'_{u,v} = R \cdot \exp(\lambda_H)$, hence Equation A.2.10 is further bifurcated as

$$P'(\mathbf{A}) = \prod_{u,v \in W} \frac{1}{Z'} \frac{1 - R_{u,v}}{1 - R_{u,v}(1 - S_{u,v})} [R'_{u,v}]^{a_{u,v}} S_{u,v}^{1_{a_{u,v}}} \cdot \prod_{\neg u,v \in W} P_{u,v}(a_{u,v}) \tag{A.2.11}$$

Now, for partition function Z', we know that $\sum_{\mathbf{A}} P'(\mathbf{A}) = 1$ and also $a_{u,v} \in \mathbb{N}_0$. Thus, $Z' = \sum_{a_{u,v} \in \mathbb{N}_0} \frac{1-R_{u,v}}{1-R_{u,v}(1-S_{u,v})} [R'_{u,v}]^{a_{u,v}} S_{u,v}^{1_{a_{u,v}}} = \frac{1-R_{u,v}}{1-R_{u,v}(1-S_{u,v})} \cdot \frac{1-R'_{u,v}(1-S_{u,v})}{1-R'_{u,v}}$

Now, putting in Equation A.2.11, we get

$$P'(\mathbf{A}) = \prod_{u,v \in W} \frac{1 - R'_{u,v}}{1 - R'_{u,v}(1 - S_{u,v})} [R'_{u,v}]^{a_{u,v}} S_{u,v}^{1_{a_{u,v}}} \cdot \prod_{\neg u,v \in W} \frac{1 - R_{u,v}}{1 - R_{u,v}(1 - S_{u,v})} R_{u,v}^{a_{u,v}} S_{u,v}^{1_{a_{u,v}}}$$

$$\text{Hence, we can say} \quad P'(\mathbf{A}) = \prod_{u,v \in V} \frac{1 - R'_{u,v}}{1 - R'_{u,v}(1 - S_{u,v})} \cdot (R'_{u,v})^{a_{u,v}} \cdot S_{u,v}^{1_{a_{u,v}}}$$

$$\text{where} \quad R'_{u,v} = \begin{cases} R_{u,v} \cdot \exp(\lambda_H), & \text{if } (u,v) \in W \\ R_{u,v}, & otherwise \end{cases}$$

Note: The λ_H can be found using the bi-section method. [Boyd and Vandenberghe, 2004]. ■

Appendix **B**

Proofs from Chapter 5

B.1 Proof of Theorem 5.1

The proof follows the theorem given in Janson [2018]. However, the author considered the geometric distribution of type $\mathbb{P}(X_i = k) = p_i(1-p_i)^{k-1}$, where $k = 1, 2, \ldots$. Whereas, we require the distribution of type $\mathbb{P}(Y_i = k) = p_i(1-p_i)^{k}$, where $k = 0, 1, 2, \ldots$. Hence, on the same principles the Theorem 5.1 is derived.

Let $X_1, X_2, \ldots X_n$ be n independent geometric random variables with possibly different distributions: $X_i \sim \mathbf{Ge}(p_i)$ with $0 < p_i \leq 1$ (probability of success), i.e.,

$$\mathbb{P}(X_i = k) = p_i(1-p_i)^k, \ \ where k = 0, 1, 2, \ldots \tag{B.1.1}$$

Let $X = \sum_{i=1}^{n} X_i$, then to estimate the upper bound of tail probability $\mathbb{P}(X \geq x)$, we define

$$\mu = \mathbb{E}[X] = \sum_{i=1}^{n} \mathbb{E}[X_i] = \sum_{i=1}^{n} \frac{1-p_i}{p_i} \text{ or } \left[\sum_{i=1}^{n} \frac{1}{p_i}\right] - n, \tag{B.1.2}$$

$$\Rightarrow \qquad \mu + n = \sum_{i=1}^{n} \frac{1}{p_i}, \tag{B.1.3}$$

and,

$$p_* = \min_i p_i \tag{B.1.4}$$

We also use the moment generating function for geometric distribution given as

$$\mathbb{E}\left[e^{tX_i}\right] = \sum_{k=0}^{\infty} e^{tk}\mathbb{P}(X_i = k) = \frac{p_i}{1 - e^t(1-p_i)}, \qquad \forall t < -\ln(1-p_i). \tag{B.1.5}$$

Now, using the Markov's Inequality we get the upper bound as

$$\mathbb{P}(X \geq \lambda\mu) \leq e^{-t\lambda\mu}\mathbb{E}\left[e^{tX}\right], \qquad \text{for} t \geq 0 \text{ and} \lambda \geq 1. \tag{B.1.6}$$

Following Equation B.1.5

$$\mathbb{E}\left[e^{tX_i}\right] = \frac{e^{-t}p_i}{e^{-t}-1+p_i} \tag{B.1.7}$$

Now, if $0 \leq t < p_i$, then $0 < p_i - t \leq e^{-t} - 1 + p_i$. Thus,

$$\mathbb{E}\left[e^{tX_i}\right] = \frac{e^{-t}p_i}{e^{-t}-1+p_i} \leq \frac{e^{-t}p_i}{p_i - t} = e^{-t}\left(1-\frac{t}{p_i}\right)^{-1}. \tag{B.1.8}$$

now, extending Equation B.1.8, if $0 \leq t < p_* = \min_i p_i$, then

$$\mathbb{E}\left[e^{tX}\right] = \prod_{i=1}^{n}\mathbb{E}\left[e^{tX_i}\right] \leq \prod_{i=1}^{n} e^{-t}\left(1-\frac{t}{p_i}\right)^{-1}. \tag{B.1.9}$$

Now, substituting Equation B.1.9 in Equation B.1.6, we get

$$\mathbb{P}(X \geq \lambda\mu) \leq e^{-t\lambda\mu}\prod_{i=1}^{n} e^{-t}\left(1-\frac{t}{p_i}\right)^{-1} \tag{B.1.10}$$

$$\Rightarrow \qquad \mathbb{P}(X \geq \lambda\mu) \leq \exp\left(-t\lambda\mu - nt + \sum_{i=1}^{n} -\ln\left(1-\frac{t}{p_i}\right)\right). \tag{B.1.11}$$

We note that $x \mapsto -\ln(1-x)$ is convex on $(0,1)$ and 0 for $x = 0$, then

$$-\ln(1-x) \leq -\frac{x}{y}\ln(1-y), \quad 0 < x \leq y < 1.$$

We know that $0 < \frac{p_*}{p_i} \leq 1$ and $0 \leq t < p_*$, thus $0 < \frac{t}{p_i} \leq \frac{t}{p_*} < 1$, which suggests

$$-\ln\left(1-\frac{t}{p_i}\right) \leq -\frac{p_*}{p_i}\ln\left(1-\frac{t}{p_*}\right). \tag{B.1.12}$$

Now, substituting Equation B.1.12 in Equation B.1.11, we get

$$\mathbb{P}(X \geq \lambda\mu) \leq \exp\left(-t\lambda\mu - nt - \ln\left(1-\frac{t}{p_*}\right)\sum_{i=1}^{n}\frac{p_*}{p_i}\right). \tag{B.1.13}$$

Further, using Equation B.1.3, we get

$$\mathbb{P}(X \geq \lambda\mu) \leq \exp\left(-t\lambda\mu - nt - p_*(\mu+n)\ln\left(1-\frac{t}{p_*}\right)\right). \tag{B.1.14}$$

Finally, using $t = \mu p_*\left(\frac{\lambda-1}{\lambda\mu+n}\right)$, which is optimal for Equation B.1.14. Thus, we get

$$\mathbb{P}(X \geq \lambda\mu) \leq \exp\left(-\mu p_*(\lambda-1) - p_*(\mu+n)\ln\left(\frac{\mu+n}{\lambda\mu+n}\right)\right). \tag{B.1.15}$$

■

Appendix **C**

Results from Chapter 6

C.1 Results from Section 6.3.3

Table C.1. Estimation results of 8 different multiple regression models for the dataset of all airlines combined (Figure 6.8). The p-values of each coefficient are shown in the parenthesis. The values are highlighted in bold if p-value is significant, i.e., < 0.001.

Var	$M^c_{AD,S}$	$M^i_{AD,S}$	$M^c_{AD,A}$	$M^i_{AD,A}$	$M^c_{SI,S}$	$M^i_{SI,S}$	$M^c_{SI,A}$	$M^i_{SI,A}$
χ	-0.2405	**-1.0239**	-0.2781	**-1.2417**	-0.4509	**-1.1642**	**-0.5661**	**-1.2235**
	(0.0869)	**(0.0000)**	(0.0578)	**(0.0000)**	(0.0010)	**(0.0000)**	**(0.0001)**	**(0.0000)**
$\overline{D}$	**0.0279**	**0.0267**	**0.0280**	**0.0276**	**0.0275**	**0.0262**	**0.0273**	**0.0267**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
$Q_{0.5}$	**0.9973**	**0.9982**	**0.9973**	**0.9985**	**0.9976**	**0.9969**	**0.9978**	**0.9970**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
$d_{5,6}$	**0.3707**	**0.3702**	**0.3706**	**0.3706**	**0.3695**	**0.3676**	**0.3697**	**0.3685**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
$d_{6,7}$	**0.2396**	**0.2432**	**0.2395**	**0.2419**	**0.2399**	**0.2375**	**0.2395**	**0.2378**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
$d_{7,8}$	**0.1260**	**0.1274**	**0.1259**	**0.1267**	**0.1257**	**0.1234**	**0.1254**	**0.1239**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
$d_{8,9}$	**0.0667**	**0.0679**	**0.0666**	**0.0677**	**0.0668**	**0.0666**	**0.0667**	**0.0665**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
$d_{9,10}$	0.0029	0.0028	0.0029	0.0028	0.0029	0.0021	0.0030	0.0022
	(0.0922)	(0.0967)	(0.0898)	(0.0982)	(0.0854)	(0.2084)	(0.0794)	(0.2063)
$dist$	**0.0017**	**0.0015**	**0.0017**	**0.0015**	**0.0016**	**0.0017**	**0.0016**	**0.0017**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
HHI	-0.1085	-0.3242	-0.1150	-0.3708	-0.1644	-0.1257	-0.1919	-0.1274
	(0.4980)	(0.0429)	(0.4731)	(0.0211)	(0.3067)	(0.4265)	(0.2343)	(0.4208)
OEP_O	**0.5607**	**0.6365**	**0.5728**	**0.6713**	**0.6785**	**0.4504**	**0.7236**	**0.4611**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
OEP_D	**0.8381**	**0.9044**	**0.8487**	**0.9503**	**0.9600**	**0.7335**	**1.0053**	**0.7492**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
κ	**2.4080**	**2.7605**	**2.4204**	**2.8347**	**2.3203**	**2.5775**	**2.3214**	**2.5613**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
R-squared	0.997							

Table C.2. Estimation results of 8 different multiple regression models (Figure 6.9) for the dataset of only American airline's flights (number of observations is 4148). The p-values of each coefficient are shown in the parenthesis. The values are highlighted in bold if p-value is significant, i.e., < 0.001.

Var	$M^c_{AD,S}$	$M^i_{AD,S}$	$M^c_{AD,A}$	$M^i_{AD,A}$	$M^c_{SI,S}$	$M^i_{SI,S}$	$M^c_{SI,A}$	$M^i_{SI,A}$
χ	0.6465	0.6368	0.7075	**0.8995**	0.8145	**2.2629**	1.0033	**5.3594**
	(0.1516)	(0.0022)	(0.1518)	**(0.0004)**	(0.0037)	**(0.0001)**	(0.0018)	**(0.0000)**
$\overline{D}$	-0.0123	-0.0096	-0.0124	-0.0107	-0.0116	-0.0092	-0.0110	-0.0067
	(0.1053)	(0.2098)	(0.1034)	(0.1613)	(0.1280)	(0.2304)	(0.1494)	(0.3776)
$Q_{0.5}$	**1.0160**	**1.0159**	**1.0159**	**1.0156**	**1.0151**	**1.0156**	**1.0147**	**1.0139**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
$d_{5,6}$	0.1731	0.1642	0.1734	0.1638	0.1782	0.1749	0.1753	0.1698
	(0.0020)	(0.0033)	(0.0019)	(0.0034)	(0.0014)	(0.0017)	(0.0017)	(0.0022)
$d_{6,7}$	**0.1920**	**0.1830**	**0.1915**	**0.1830**	**0.1887**	**0.1901**	**0.1882**	**0.1903**
	(0.0001)	**(0.0001)**	**(0.0001)**	**(0.0001)**	**(0.0001)**	**(0.0001)**	**(0.0001)**	**(0.0001)**
$d_{7,8}$	0.0705	0.0668	0.0707	0.0662	0.0708	0.0683	0.0692	0.0606
	(0.0531)	(0.0668)	(0.0526)	(0.0690)	(0.0520)	(0.0606)	(0.0574)	(0.0939)
$d_{8,9}$	0.0107	0.0058	0.0111	0.0049	0.0111	0.0076	0.0116	0.0007
	(0.6226)	(0.7905)	(0.6094)	(0.8230)	(0.6096)	(0.7257)	(0.5956)	(0.9727)
$d_{9,10}$	-0.0038	-0.0040	-0.0037	-0.0038	-0.0040	-0.0036	-0.0041	-0.0037
	(0.2264)	(0.2022)	(0.2289)	(0.2235)	(0.1978)	(0.2487)	(0.1884)	(0.2335)
$dist$	-0.0001	0.0000	0.0000	0.0001	0.0001	-0.0001	0.0001	0.0001
	(0.9179)	(0.9540)	(0.9434)	(0.8689)	(0.9056)	(0.9340)	(0.8327)	(0.8205)
HHI	**-2.8215**	**-2.8024**	**-2.8176**	**-2.7701**	**-2.6664**	**-2.6320**	**-2.6174**	**-2.3039**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
OEP_O	-0.3015	-0.2686	-0.3132	-0.2818	-0.5788	-0.1558	-0.6785	-0.1549
	(0.2338)	(0.2746)	(0.2201)	(0.2517)	(0.0359)	(0.5257)	(0.0181)	(0.5249)
OEP_D	**1.0312**	**1.1042**	**1.0220**	**1.0671**	**0.7346**	**1.2138**	**0.6407**	**1.2070**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0041)**	**(0.0000)**	**(0.0166)**	**(0.0000)**
κ	**3.4118**	**3.5584**	**3.3746**	**3.4058**	**3.8383**	**3.5855**	**3.8673**	**3.3091**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
R-squared	0.997							

Table C.3. Estimation results of 8 different multiple regression models (Figure 6.10) for the dataset of only United airline's flights (number of observations is 2563). The p-values of each coefficient are shown in the parenthesis. The values are highlighted in bold if p-value is significant, i.e., < 0.001.

Var	$M^c_{AD,S}$	$M^i_{AD,S}$	$M^c_{AD,A}$	$M^i_{AD,A}$	$M^c_{SI,S}$	$M^i_{SI,S}$	$M^c_{SI,A}$	$M^i_{SI,A}$
χ	-0.7151	-0.8339	-0.5860	-0.9995	-1.1606	1.5340	-1.4077	1.6129
	(0.3552)	(0.0349)	(0.4670)	(0.0277)	(0.0100)	(0.1358)	(0.0037)	(0.1983)
$\overline{D}$	0.0205	0.0205	0.0205	0.0210	0.0217	0.0210	0.0219	0.0208
	(0.0111)	(0.0111)	(0.01110)	(0.0094)	(0.0071)	(0.0093)	(0.0068)	(0.0100)
$Q_{0.5}$	**0.9910**	**0.9903**	**0.9910**	**0.9906**	**0.9916**	**0.9910**	**0.9921**	**0.9911**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
$d_{5,6}$	**0.4864**	**0.4868**	**0.4861**	**0.4877**	**0.4839**	**0.4848**	**0.4827**	**0.4863**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
$d_{6,7}$	**0.2448**	**0.2469**	**0.2446**	**0.2466**	**0.2428**	**0.2480**	**0.2422**	**0.2460**
	(0.0000)	**(0.0000)**	**(0.0001)**	**(0.0000)**	**(0.0001)**	**(0.0000)**	**(0.0001)**	**(0.0000)**
$d_{7,8}$	-0.0182	-0.0177	-0.0177	-0.0178	-0.0134	-0.0185	-0.0150	-0.0180
	(0.6848)	(0.6922)	(0.6923)	(0.6901)	(0.7645)	(0.6793)	(0.7376)	(0.6867)
$d_{8,9}$	0.0182	0.0194	0.0183	0.0193	0.0183	0.0175	0.0177	0.0179
	(0.4799)	(0.4490)	(0.4759)	(0.4524)	(0.4769)	(0.4956)	(0.4894)	(0.4869)
$d_{9,10}$	-0.0018	-0.0021	-0.0018	-0.0020	-0.0018	-0.0019	-0.0018	-0.0018
	(0.6302)	(0.5812)	(0.6315)	(0.5977)	(0.6301)	(0.6179)	(0.6245)	(0.6393)
$dist$	0.0017	0.0018	0.0017	0.0017	0.0016	0.0017	0.0015	0.0017
	(0.0114)	(0.0082)	(0.0119)	(0.0097)	(0.0154)	(0.0126)	(0.0204)	(0.0130)
HHI	**1.6094**	**1.4502**	**1.6085**	**1.4208**	**1.3682**	**1.7061**	**1.3244**	**1.7062**
	(0.0001)	**(0.0007)**	**(0.0001)**	**(0.0009)**	**(0.0014)**	**(0.0001)**	**(0.0021)**	**(0.0001)**
OEP_O	0.1800	0.1112	0.1718	0.1075	0.3361	0.1719	0.3977	0.1577
	(0.5407)	(0.6979)	(0.5613)	(0.7076)	(0.2603)	(0.5517)	(0.1879)	(0.5843)
OEP_D	0.0809	0.0252	0.0664	0.0253	0.2304	0.0293	0.2837	0.0227
	(0.7807)	(0.9284)	(0.8193)	(0.9282)	(0.4316)	(0.9168)	(0.3371)	(0.9356)
κ	**6.8330**	**6.7659**	**6.7282**	**6.8717**	**6.7865**	**6.1265**	**6.8682**	**6.1151**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
R-squared	0.998							

Table C.4. Estimation results of 8 different multiple regression models (Figure 6.11) for the dataset of only Delta airline's flights (number of observations is 2223). The p-values of each coefficient are shown in the parenthesis. The values are highlighted in bold if p-value is significant, i.e., < 0.001.

Var	$M^c_{AD,S}$	$M^i_{AD,S}$	$M^c_{AD,A}$	$M^i_{AD,A}$	$M^c_{SI,S}$	$M^i_{SI,S}$	$M^c_{SI,A}$	$M^i_{SI,A}$
χ	-0.4507	-0.0986	-0.5918	-0.1986	-0.8739	**-1.9772**	-1.0494	-1.6932
	(0.3486)	(0.7934)	(0.2423)	(0.6495)	(0.0424)	**(0.0004)**	(0.0224)	(0.0155)
$\overline{D}$	0.0355	0.0360	0.0354	0.0359	0.0349	0.0367	0.0348	0.0364
	(0.0021)	(0.0018)	(0.0021)	(0.0018)	(0.0024)	(0.0014)	(0.0024)	(0.0015)
$Q_{0.5}$	**0.9255**	**0.9254**	**0.9253**	**0.9254**	**0.9265**	**0.9247**	**0.9266**	**0.9254**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
$d_{5,6}$	**0.6768**	**0.6772**	**0.6767**	**0.6771**	**0.6785**	**0.6838**	**0.6783**	**0.6820**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
$d_{6,7}$	**0.4347**	**0.4353**	**0.4347**	**0.4355**	**0.4329**	**0.4391**	**0.4326**	**0.4360**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
$d_{7,8}$	**0.4408**	**0.4417**	**0.4412**	**0.4417**	**0.4448**	**0.4407**	**0.4441**	**0.4403**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
$d_{8,9}$	**0.1966**	**0.1963**	**0.1963**	**0.1961**	**0.1957**	**0.1933**	**0.1953**	**0.1932**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
$d_{9,10}$	-0.0081	-0.0082	-0.0081	-0.0082	-0.0081	-0.0085	-0.0081	-0.0085
	(0.0376)	(0.0349)	(0.0368)	(0.0347)	(0.0365)	(0.0271)	(0.0373)	(0.0289)
$dist$	**0.0093**	**0.0093**	**0.0093**	**0.0093**	**0.0092**	**0.0093**	**0.0091**	**0.0093**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
HHI	1.5126	**1.6037**	1.4818	**1.5735**	1.3054	**1.5174**	1.2603	**1.5461**
	(0.0013)	**(0.0006)**	(0.0016)	**(0.0008)**	(0.0064)	**(0.0008)**	(0.0085)	**(0.0006)**
OEP_O	0.2593	0.2119	0.2765	0.2112	0.3886	0.0041	0.4315	0.0522
	(0.2967)	(0.3868)	(0.2679)	(0.3877)	(0.1327)	(0.9871)	(0.0990)	(0.8366)
OEP_D	**1.2798**	**1.2214**	**1.3006**	**1.2232**	**1.4102**	**0.9712**	**1.4609**	**1.0548**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0002)**	**(0.0000)**	**(0.0001)**
κ	**4.6264**	**4.3317**	**4.7371**	**4.4089**	**4.6654**	**5.0210**	**4.7140**	**4.8304**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
R-squared	0.997							

Table C.5. Estimation results of 8 different multiple regression models (Figure 6.12) for the dataset of only Low-Cost Carrier's flights (number of observations is 3817). The p-values of each coefficient are shown in the parenthesis. The values are highlighted in bold if p-value is significant, i.e., < 0.001.

Var	$M^c_{AD,S}$	$M^i_{AD,S}$	$M^c_{AD,A}$	$M^i_{AD,A}$	$M^c_{SI,S}$	$M^i_{SI,S}$	$M^c_{SI,A}$	$M^i_{SI,A}$
χ	-0.1601	0.2411	-0.1153	0.4767	**-0.7702**	0.7462	**-0.8798**	1.9709
	(0.5129)	(0.2573)	(0.6542)	(0.0602)	**(0.0009)**	(0.1230)	**(0.0003)**	(0.0011)
$\overline{D}$	0.0010	0.0013	0.0011	0.0012	-0.0001	0.0018	-0.0008	0.0011
	(0.8493)	(0.8050)	(0.8419)	(0.8251)	(0.9868)	(0.7328)	(0.8811)	(0.8324)
$Q_{0.5}$	**1.0246**	**1.0241**	**1.0246**	**1.0237**	**1.0254**	**1.0240**	**1.0256**	**1.0230**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
$d_{5,6}$	**0.3277**	**0.3291**	**0.3279**	**0.3301**	**0.3306**	**0.3285**	**0.3316**	**0.3287**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
$d_{6,7}$	**0.2047**	**0.2047**	**0.2048**	**0.2051**	**0.2019**	**0.2076**	**0.1997**	**0.2100**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
$d_{7,8}$	0.1087	0.1087	0.1088	0.1086	0.1056	**0.1099**	0.1052	**0.1122**
	(0.0011)	(0.0011)	(0.0011)	(0.0011)	(0.0015)	**(0.0010)**	(0.0016)	**(0.0008)**
$d_{8,9}$	0.0245	0.0245	0.0246	0.0245	0.0244	0.0233	0.0246	0.0233
	(0.2180)	(0.2183)	(0.2177)	(0.2176)	(0.2189)	(0.2431)	(0.2168)	(0.2421)
$d_{9,10}$	-0.0006	-0.0009	-0.0006	-0.0010	-0.0003	-0.0007	-0.0003	-0.0006
	(0.8538)	(0.7802)	(0.8396)	(0.7505)	(0.9303)	(0.8183)	(0.9323)	(0.8382)
$dist$	-0.0008	-0.0007	-0.0008	-0.0007	-0.0009	-0.0007	-0.0009	-0.0006
	(0.2114)	(0.2568)	(0.2103)	(0.2964)	(0.1583)	(0.2457)	(0.1452)	(0.3117)
HHI	-0.5795	-0.4726	-0.5645	-0.4245	-0.6462	-0.4886	-0.6616	-0.3845
	(0.0797)	(0.1467)	(0.0882)	(0.1935)	(0.0456)	(0.1301)	(0.0408)	(0.2359)
OEP_O	0.2644	0.1907	0.2563	0.1517	0.5448	0.2288	0.5721	0.2148
	(0.1409)	(0.2785)	(0.1570)	(0.3922)	(0.0055)	(0.1840)	(0.0036)	(0.2120)
OEP_D	-0.2723	-0.3387	-0.2806	-0.3763	0.0049	-0.3022	0.0347	-0.3075
	(0.1246)	(0.0497)	(0.1165)	(0.0307)	(0.9796)	(0.0755)	(0.8584)	(0.0701)
κ	**2.8112**	**2.6096**	**2.7801**	**2.5180**	**2.6758**	**2.6562**	**2.6893**	**2.5629**
	(0.0000)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**
R-squared	0.998							

Bibliography

Abello, J., Resende, M. G., and Sudarsky, S. (2002). Massive quasi-clique detection. In *Latin American symposium on theoretical informatics*, pages 598–612. Springer.

Adhikari, B., Zhang, Y., Bharadwaj, A., and Prakash, B. A. (2017). Condensing temporal networks using propagation. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 417–425. SIAM.

Adriaens, F., Lijffijt, J., and De Bie, T. (2017). Subjectively interesting connecting trees. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 53–69. Springer.

Aggarwal, C. C., Wang, H., et al. (2010). *Managing and mining graph data*, volume 40. Springer.

Ahmed, R. and Karypis, G. (2012). Algorithms for mining the evolution of conserved relational states in dynamic networks. *Knowledge and Information Systems*, 33(3):603–630.

Ahmed, R. and Karypis, G. (2015). Algorithms for mining the coevolving relational motifs in dynamic networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(1):1–31.

Alpert, C. J., Kahng, A. B., and Yao, S.-Z. (1999). Spectral partitioning with multiple eigenvectors. *Discrete Applied Mathematics*, 90(1-3):3–26.

Andersen, R. and Chellapilla, K. (2009). Finding dense subgraphs with size bounds. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 25–37. Springer.

Andersson, E., Housos, E., Kohl, N., and Wedelin, D. (1998). Crew pairing optimization. In *Operations research in the airline industry*, pages 228–258. Springer.

Angles, R. and Gutierrez, C. (2008). Survey of graph database models. *ACM Comput. Surv.*, 40(1).

Apostolico, A., Barbares, M., and Pizzi, C. (2011). Speedup for a periodic subgraph miner. *Information Processing Letters*, 111(11):521–523.

Araujo, M., Papadimitriou, S., Günnemann, S., Faloutsos, C., Basu, P., Swami, A., Papalexakis, E. E., and Koutra, D. (2014). Com2: fast automatic discovery of temporal ('comet') communities. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 271–283. Springer.

Balakrishna, P., Ganesan, R., Sherry, L., and Levy, B. S. (2008). Estimating taxi-out times with a reinforcement learning algorithm. In *2008 IEEE/AIAA 27th Digital Avionics Systems Conference*, pages 3–D. IEEE.

Ball, M., Barnhart, C., Dresner, M., Hansen, M., Neels, K., Odoni, A., Peterson, E., Sherry, L., Trani, A., Zou, B., et al. (2010). Total delay impact study. In *NEXTOR Research Symposium, Washington DC*.

Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *science*, 286(5439):509–512.

Batagelj, V. and Zaversnik, M. (2003). An o (m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049*.

Bazargan, M. (2016). *Airline operations and scheduling*. Routledge.

Bendimerad, A., Mel, A., Lijffijt, J., Plantevit, M., Robardet, C., and De Bie, T. (2020). Sias-miner: mining subjectively interesting attributed subgraphs. *Data Mining and Knowledge Discovery*, 34(2):355–393.

Borgwardt, K. M., Kriegel, H.-P., and Wackersreuther, P. (2006). Pattern mining in frequent dynamic subgraphs. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 818–822. IEEE.

Bouvel, M., Grebinski, V., and Kucherov, G. (2005). Combinatorial search on graphs motivated by bioinformatics applications: A brief survey. In Kratsch, D., editor, *Graph-Theoretic Concepts in Computer Science*, pages 16–27, Berlin, Heidelberg. Springer Berlin Heidelberg.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Bringmann, B. and Nijssen, S. (2008). What is frequent in a single graph? In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 858–863. Springer.

Bringmann, B., Nijssen, S., and Zimmermann, A. (2011). Pattern-based classification: A unifying perspective. *arXiv preprint arXiv:1111.6191*.

BTS (2019). The bureau of transportation statistics databases. Retrieved from https://www.bts.gov/.

Carvalho, L., Sternberg, A., Maia Gonçalves, L., Beatriz Cruz, A., Soares, J. A., Brandão, D., Carvalho, D., and Ogasawara, E. (2020). On the relevance of data science for flight delay research: a systematic review. *Transport Reviews*, pages 1–30.

Chan, R., Yang, Q., and Shen, Y. (2003). Mining high utility itemsets. In *2013 IEEE 13th International Conference on Data Mining*, page 19, Los Alamitos, CA, USA. IEEE Computer Society.

Charikar, M. (2000). Greedy approximation algorithms for finding dense components in a graph. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 84–95. Springer.

Charikar, M., Naamad, Y., and Wu, J. (2018). On finding dense common subgraphs. *arXiv preprint arXiv:1802.06361*.

Chatterjee, N., Jain, G., and Bajwa, G. S. (2018). Single document extractive text summarization using neural networks and genetic algorithm. In *Science and Information Conference*, pages 338–358. Springer.

Clauset, A., Newman, M. E., and Moore, C. (2004). Finding community structure in very large networks. *Physical review E*, 70(6):066111.

Cook, D. J. and Holder, L. B. (1994). Substructure discovery using minimum description length and background knowledge. *J. Artif. Int. Res.*, 1(1):231–255.

Cover, T. and Thomas, J. (1991). Elements of information theory,(pp 33-36) john wiley and sons. *Inc, NY*.

Dakiche, N., Tayeb, F. B.-S., Slimani, Y., and Benatchba, K. (2019). Tracking community evolution in social networks: A survey. *Information Processing & Management*, 56(3):1084–1102.

De Bie, T. (2011a). An information theoretic framework for data mining. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, page 564–572, New York, NY, USA. Association for Computing Machinery.

De Bie, T. (2011b). Maximum entropy models and subjective interestingness: an application to tiles in binary databases. *Data Mining and Knowledge Discovery*, 23(3):407–446.

Deng, J., Lijffijt, J., Kang, B., and De Bie, T. (2019). Simit: Subjectively interesting motifs in time series. *Entropy*, 21(6):566.

Ding, C. H., He, X., Zha, H., Gu, M., and Simon, H. D. (2001). A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 107–114. IEEE.

Dong, X., Frossard, P., Vandergheynst, P., and Nefedov, N. (2012). Clustering with multi-layer graphs: A spectral perspective. *IEEE Transactions on Signal Processing*, 60(11):5820–5831.

Duflot, R., Avon, C., Roche, P., and Bergès, L. (2018). Combining habitat suitability models and spatial graphs for more effective landscape conservation planning: An applied

methodological framework and a species case study. *Journal for Nature Conservation*, 46:38–47.

Dutta, P., Saha, S., and Gulati, S. (2019). Graph-based hub gene selection technique using protein interaction information: Application to sample classification. *IEEE journal of biomedical and health informatics*, 23(6):2670–2676.

Elseidy, M., Abdelhamid, E., Skiadopoulos, S., and Kalnis, P. (2014). Grami: Frequent subgraph and pattern mining in a single large graph. *Proceedings of the VLDB Endowment*, 7(7):517–528.

Estrada, E., Patlewicz, G., and Uriarte, E. (2003). From molecular graphs to drugs. a review on the use of topological indices in drug design and discovery. *Indian journal of chemistry. Sect. A: Inorganic, physical, theoretical & analytical*, 42:1315–1329.

Fang, Y., Wang, Z., Cheng, R., Wang, H., and Hu, J. (2018). Effective and efficient community search over large directed graphs. *IEEE Transactions on Knowledge and Data Engineering*, 31(11):2093–2107.

Flake, G. W., Tarjan, R. E., and Tsioutsiouliklis, K. (2004). Graph clustering and minimum cut trees. *Internet Mathematics*, 1(4):385–408.

Fournier-Viger, P., He, G., Cheng, C., Li, J., Zhou, M., Lin, J. C.-W., and Yun, U. (2020). A survey of pattern mining in dynamic graphs. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, page e1372.

Galimberti, E., Barrat, A., Bonchi, F., Cattuto, C., and Gullo, F. (2018). Mining (maximal) span-cores from temporal networks. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 107–116. ACM.

Galimberti, E., Bonchi, F., and Gullo, F. (2017). Core decomposition and densest subgraph in multilayer networks. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1807–1816.

Geng, L. and Hamilton, H. J. (2006). Interestingness measures for data mining: A survey. *ACM Comput. Surv.*, 38(3):9–es.

Gibson, D., Kumar, R., and Tomkins, A. (2005). Discovering large dense subgraphs in massive graphs. In *Proceedings of the 31st international conference on Very large data bases*, pages 721–732.

Goebl, S., Tonch, A., Böhm, C., and Plant, C. (2016). Megs: Partitioning meaningful subgraph structures using minimum description length. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 889–894. IEEE.

Goldberg, M. K. (1984). Edge-coloring of multigraphs: Recoloring technique. *Journal of Graph Theory*, 8(1):123–137.

Grochow, J. A. and Kellis, M. (2007). Network motif discovery using subgraph enumeration and symmetry-breaking. In *Annual International Conference on Research in Computational Molecular Biology*, pages 92–106. Springer.

Grünwald, P. D. (2007). *The minimum description length principle*. MIT press.

Gui, G., Liu, F., Sun, J., Yang, J., Zhou, Z., and Zhao, D. (2019). Flight delay prediction based on aviation big data and machine learning. *IEEE Transactions on Vehicular Technology*, 69(1):140–150.

Gupta, S. K. and Jana, P. K. (2015). Energy efficient clustering and routing algorithms for wireless sensor networks: Ga based approach. *Wireless Personal Communications*, 83(3):2403–2423.

Győri, E. and Palmer, C. (2009). A new type of edge-derived vertex coloring. *Discrete mathematics*, 309(22):6344–6352.

Halder, S., Samiullah, M., and Lee, Y.-K. (2017). Supergraph based periodic pattern mining in dynamic social networks. *Expert Systems with Applications*, 72:430–442.

Han, J., Pei, J., and Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

Hao, L. and Hansen, M. (2013). How airlines set scheduled block times. In *10th USA/Europe Air Traffic Management Research and Development Seminar, Chicago IL*, pages 1–9.

Hao, L. and Hansen, M. (2014). Block time reliability and scheduled block time setting. *Transportation Research Part B: Methodological*, 69:98–111.

Hao, L., Hansen, M., Lovell, D., Vlachou, K., and Ball, M. (2014a). Flight predictability: Concepts, metrics and impacts. *The National Center of Excellence for Airline Operations Research (NEXTOR)*, pages 1–68.

Hao, L., Hansen, M., Zhang, Y., and Post, J. (2014b). New york, new york: Two ways of estimating the delay impact of new york airports. *Transportation Research Part E: Logistics and Transportation Review*, 70:245–260.

Hilderman, R. J. and Hamilton, H. J. (2013). *Knowledge discovery and measures of interest*, volume 638. Springer Science & Business Media.

Ingalalli, V., Ienco, D., and Poncelet, P. (2018). Mining frequent subgraphs in multigraphs. *Information Sciences*, 451:50–66.

Jain, P., Singh, P. K., and Abraham, A. (2011). Intrusion detection and self healing model for network security. In *2011 7th International Conference on Next Generation Web Services Practices*, pages 320–325.

Janson, S. (2018). Tail bounds for sums of geometric and exponential variables. *Statistics & Probability Letters*, 135:1–6.

Jaroszewicz, S. and Simovici, D. A. (2004). Interestingness of frequent itemsets using bayesian networks as background knowledge. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 178–186.

Jaynes, E. T. (1982). On the rationale of maximum-entropy methods. *Proceedings of the IEEE*, 70(9):939–952.

Jethava, V. and Beerenwinkel, N. (2015). Finding dense subgraphs in relational graphs. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 641–654. Springer.

Kang, L. and Hansen, M. (2017). Behavioral analysis of airline scheduled block time adjustment. *Transportation Research Part E: Logistics and Transportation Review*, 103:56–68.

Kapoor, S., Saxena, D. K., and van Leeuwen, M. (2020). Discovering subjectively interesting multigraph patterns. *Machine Learning*, 109(8):1669–1696.

Kapoor, S., Saxena, D. K., and van Leeuwen, M. (2021). Online summarization of dynamic graphs using subjective interestingness for sequential data. *Data Mining and Knowledge Discovery*, 35(1):88–126.

Kapoor, S. and van Leeuwen, M. (2021). Mining subjective subgraph patterns. doi: 10.5281/zenodo.4483825. *Zenodo*.

Khan, A. and Aggarwal, C. (2016). Query-friendly compression of graph streams. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 130–137.

Khuller, S. and Saha, B. (2009). On finding dense subgraphs. In *International Colloquium on Automata, Languages, and Programming*, pages 597–608. Springer.

Kim, Y. J., Choi, S., Briceno, S., and Mavris, D. (2016). A deep learning approach to flight delay prediction. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pages 1–6. IEEE.

Kontonasios, K.-N. and De Bie, T. (2015). Subjectively interesting alternative clusterings. *Machine Learning*, 98(1-2):31–56.

Koutra, D., Kang, U., Vreeken, J., and Faloutsos, C. (2014). Vog: Summarizing and understanding large graphs. In *Proceedings of the 2014 SIAM international conference on data mining*, pages 91–99. SIAM.

Lee, T. C. (2001). An introduction to coding theory and the two-part minimum description length principle. *International statistical review*, 69(2):169–183.

LeFevre, K. and Terzi, E. (2010). Grass: Graph structure summarization. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 454–465. SIAM.

Liang, Z., Feng, Y., Zhang, X., Wu, T., and Chaovalitwongse, W. A. (2015). Robust weekly aircraft maintenance routing problem and the extension to the tail assignment problem. *Transportation Research Part B: Methodological*, 78:238–259.

Lijffijt, J., Spyropoulou, E., Kang, B., and De Bie, T. (2016). Pn-rminer: A generic framework for mining interesting structured relational patterns. *International Journal of Data Science and Analytics*, 1(1):61–76.

Lin, Y.-R., Sun, J., Sundaram, H., Kelliher, A., Castro, P., and Konuru, R. (2011). Community discovery via metagraph factorization. *ACM Trans. Knowl. Discov. Data*, 5(3).

Ling, C. X., Chen, T., Yang, Q., and Cheng, J. (2002). Mining optimal actions for profitable crm. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 767–770. IEEE.

Liu, B., Hsu, W., Mun, L.-F., and Lee, H.-Y. (1999). Finding interesting patterns using user expectations. *IEEE Transactions on Knowledge and Data Engineering*, 11(6):817–832.

Luce, R. D. (1950). Connectivity and generalized cliques in sociometric group structure. *Psychometrika*, 15(2):169–190.

Manna, S., Biswas, S., Kundu, R., Rakshit, S., Gupta, P., and Barman, S. (2017). A statistical approach to predict flight delay using gradient boosted decision tree. In *2017 International Conference on Computational Intelligence in Data Science (ICCIDS)*, pages 1–5. IEEE.

Marshall, S., Gil, J., Kropf, K., Tomko, M., and Figueiredo, L. (2018). Street network studies: from networks to models and their representations. *Networks and Spatial Economics*, 18(3):735–749.

Matsuda, H., Ishihara, T., and Hashimoto, A. (1999). Classifying molecular sequences using a linkage graph with their pairwise similarities. *Theoretical Computer Science*, 210(2):305–325.

McClosky, B. and Hicks, I. V. (2012). Combinatorial algorithms for the maximum k-plex problem. *Journal of combinatorial optimization*, 23(1):29–49.

Meng, J. and Tu, Y.-c. (2017). Flexible and feasible support measures for mining frequent patterns in large labeled graphs. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 391–402.

Mokken, R. J. (1979). Cliques, clubs and clans. *Quality & Quantity*, 13(2):161–173.

Navlakha, S., Rastogi, R., and Shrivastava, N. (2008). Graph summarization with bounded error. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 419–432. ACM.

Newman, M. E. (2006). Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582.

Newman, M. E. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113.

Padmanabhan, B. and Tuzhilin, A. (2000). Small is beautiful: Discovering the minimal set of unexpected patterns. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, page 54–63, New York, NY, USA. Association for Computing Machinery.

Palla, G., Derényi, I., Farkas, I., and Vicsek, T. (2005). Uncovering the overlapping community structure of complex networks in nature and society. *nature*, 435(7043):814.

Papalexakis, E. E., Akoglu, L., and Ience, D. (2013). Do more views of a graph help? community detection and clustering in multi-graphs. In *Information fusion (FUSION), 2013 16th international conference on*, pages 899–905. IEEE.

Paul, M. and Anand, A. (2018). A new family of similarity measures for scoring confidence of protein interactions using gene ontology. *bioRxiv*.

Peterson, E. B., Neels, K., Barczi, N., and Graham, T. (2013). The economic cost of airline flight delay. *Journal of Transport Economics and Policy (JTEP)*, 47(1):107–121.

Piatetsky-Shapiro, G. (1990). Knowledge discovery in real databases: A report on the ijcai-89 workshop. *AI magazine*, 11(4):68–68.

Proença, H. M., Klijn, R., Bäck, T., and van Leeuwen, M. (2018). Identifying flight delay patterns using diverse subgroup discovery. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 60–67. IEEE.

Qi, G.-J., Aggarwal, C. C., and Huang, T. (2012). Community detection with edge content in social media networks. In *2012 IEEE 28th International Conference on Data Engineering*, pages 534–545. IEEE.

Qu, Q., Liu, S., Zhu, F., and Jensen, C. S. (2016). Efficient online summarization of large-scale dynamic networks. *IEEE Transactions on Knowledge and Data Engineering*, 28(12):3231–3245.

Rissanen, J. (1983). A universal prior for integers and estimation by minimum description length. *The Annals of statistics*, pages 416–431.

Robardet, C. (2009). Constraint-based pattern mining in dynamic graphs. In *2009 Ninth IEEE International Conference on Data Mining*, pages 950–955.

Rozenshtein, P., Bonchi, F., Gionis, A., Sozio, M., and Tatti, N. (2018). Finding events in temporal networks: Segmentation meets densest-subgraph discovery. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 397–406. IEEE.

Rozenshtein, P., Tatti, N., and Gionis, A. (2017). Finding dynamic dense subgraphs. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 11(3):27.

Ruan, Y., Fuhry, D., and Parthasarathy, S. (2013). Efficient community detection in large networks using content and links. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1089–1098. ACM.

Saha, S. and Bandyopadhyay, S. (2013). A generalized automatic clustering algorithm in a multiobjective framework. *Applied Soft Computing*, 13(1):89–108.

Sahar, S. (1999). Interestingness via what is not interesting. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, page 332–336, New York, NY, USA. Association for Computing Machinery.

Sahu, S. K. and Anand, A. (2018). Drug-drug interaction extraction from biomedical texts using long short-term memory network. *Journal of biomedical informatics*, 86:15–24.

Saran, D. and Vreeken, J. (2019). Summarizing dynamic graphs using mdl. In *Proceedings of the ECMLPKDD Workshop on Graph Embedding and Mining (GEM)*.

Scharwächter, E., Müller, E., Donges, J., Hassani, M., and Seidl, T. (2016). Detecting change processes in dynamic networks by frequent graph evolution rule mining. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 1191–1196. IEEE.

Seidman, S. B. (1983). Network structure and minimum degree. *Social networks*, 5(3):269–287.

Seidman, S. B. and Foster, B. L. (1978). A graph-theoretic generalization of the clique concept. *Journal of Mathematical sociology*, 6(1):139–154.

Shah, N., Koutra, D., Zou, T., Gallagher, B., and Faloutsos, C. (2015). Timecrunch: Interpretable dynamic graph summarization. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1055–1064. ACM.

Sharma, T., Charls, A., and Singh, P. (2009). Community mining in signed social networks-an automated approach. *ICCEA09, Manila*, 9:163–168.

Silberschatz, A. and Tuzhilin, A. (1996). What makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge and data engineering*, 8(6):970–974.

Silva, A., Meira Jr, W., and Zaki, M. J. (2012). Mining attribute-structure correlated patterns in large attributed graphs. *Proceedings of the VLDB Endowment*, 5(5):466–477.

Sridhar, B. and Chen, N. Y. (2009). Short-term national airspace system delay prediction using weather impacted traffic index. *Journal of guidance, control, and dynamics*, 32(2):657–662.

Sridhar, B. and Swei, S. (2006). Relationship between weather, traffic and delay based on empirical methods. In *6th AIAA aviation technology, integration and operations conference (ATIO)*, page 7760.

Sternberg, A., Carvalho, D., Murta, L., Soares, J., and Ogasawara, E. (2016). An analysis of brazilian flight delays based on frequent patterns. *Transportation Research Part E: Logistics and Transportation Review*, 95:282–298.

Sun, J., Faloutsos, C., Faloutsos, C., Papadimitriou, S., and Yu, P. S. (2007). Graphscope: parameter-free mining of large time-evolving graphs. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 687–696. ACM.

Tan, P.-N., Kumar, V., and Srivastava, J. (2002). Selecting the right interestingness measure for association patterns. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, page 32–41, New York, NY, USA. Association for Computing Machinery.

Tang, N., Chen, Q., and Mitra, P. (2016). Graph stream summarization: From big bang to big crunch. In *SIGMOD 2016 - Proceedings of the 2016 International Conference on*

Management of Data, Proceedings of the ACM SIGMOD International Conference on Management of Data, pages 1481–1496. Association for Computing Machinery. 2016 ACM SIGMOD International Conference on Management of Data, SIGMOD 2016 ; Conference date: 26-06-2016 Through 01-07-2016.

Tian, Y., Ye, B., Wan, L., Yang, M., and Xing, D. (2019). Restricted airspace unit identification using density-based spatial clustering of applications with noise. *Sustainability*, 11(21):5962.

Toivonen, H., Zhou, F., Hartikainen, A., and Hinkka, A. (2011). Compression of weighted graphs. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 965–973. ACM.

Tsalouchidou, I., Bonchi, F., Morales, G. D. F., and Baeza-Yates, R. (2020). Scalable dynamic graph summarization. *IEEE Transactions on Knowledge and Data Engineering*, 32(2):360–373.

Tsourakakis, C., Bonchi, F., Gionis, A., Gullo, F., and Tsiarli, M. (2013). Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 104–112. ACM.

Tukey, J. W. (1977). *Exploratory data analysis*, volume 2. Reading, MA.

Uno, T. (2010). An efficient algorithm for solving pseudo clique enumeration problem. *Algorithmica*, 56(1):3–16.

van Leeuwen, M., De Bie, T., Spyropoulou, E., and Mesnage, C. (2016). Subjective interestingness of subgraph patterns. *Machine Learning*, 105(1):41–75.

Veremyev, A., Prokopyev, O. A., Butenko, S., and Pasiliao, E. L. (2016). Exact mip-based approaches for finding maximum quasi-cliques and dense subgraphs. *Computational Optimization and Applications*, 64(1):177–214.

Wackersreuther, B., Wackersreuther, P., Oswald, A., Böhm, C., and Borgwardt, K. M. (2010). Frequent subgraph discovery in dynamic networks. In *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, pages 155–162.

Wang, K., Zhou, S., and Han, J. (2002). Profit mining: From patterns to actions. In *International Conference on Extending Database Technology*, pages 70–87. Springer.

Wang, P. T., Schaefer, L. A., and Wojcik, L. A. (2003). Flight connections and their impacts on delay propagation. In *Digital Avionics Systems Conference, 2003. DASC'03. The 22nd*, volume 1, pages 5–B. IEEE.

Wernicke, S. (2006). Efficient detection of network motifs. *IEEE/ACM transactions on computational biology and bioinformatics*, 3(4):347–359.

Woodburn, A. and Ryerson, M. (2014). Airport capacity enhancement and flight predictability. *Transportation Research Record*, 2400(1):87–97.

Wu, C.-L. (2016). *Airline operations and delay management: insights from airline economics, networks and strategic schedule planning*. Routledge.

Wu, Q. and Hao, J.-K. (2015). A review on algorithms for maximum clique problems. *European Journal of Operational Research*, 242(3):693–709.

Wu, W., Cai, K., Yan, Y., and Li, Y. (2019). An improved svm model for flight delay prediction. In *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, pages 1–6. IEEE.

Wu, W. and Wu, C.-L. (2018). Enhanced delay propagation tree model with bayesian network for modelling flight delay propagation. *Transportation Planning and Technology*, 41(3):319–335.

Xia, F., Wei, H., Yu, S., Zhang, D., and Xu, B. (2019). A survey of measures for network motifs. *IEEE Access*, 7:106576–106587.

Xu, Z., Ke, Y., Wang, Y., Cheng, H., and Cheng, J. (2012). A model-based approach to attributed graph clustering. In *Proceedings of the 2012 ACM SIGMOD international conference on management of data*, pages 505–516. ACM.

Yao, H. and Hamilton, H. J. (2006). Mining itemset utilities from transaction databases. *Data & Knowledge Engineering*, 59(3):603–626.

You, C. h., Holder, L. B., and Cook, D. J. (2009). Learning patterns in the dynamics of biological networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 977–986, New York, NY, USA. Association for Computing Machinery.

Zhong, N., Yao, Y. Y., and Ohishima, M. (2003). Peculiarity oriented multidatabase mining. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):952–960.

Zhou, L., Liang, Z., Chou, C.-A., and Chaovalitwongse, W. A. (2020). Airline planning and scheduling: Models and solution methodologies. *Frontiers of Engineering Management*, pages 1–26.

Zhou, Y., Cheng, H., and Yu, J. X. (2009). Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment*, 2(1):718–729.

Zou, B. and Hansen, M. (2012). Impact of operational performance on air carrier cost structure: evidence from us airlines. *Transportation Research Part E: Logistics and Transportation Review*, 48(5):1032–1048.

Zou, B. and Hansen, M. (2014). Flight delay impact on airfare and flight frequency: A comprehensive assessment. *Transportation research part E: logistics and transportation review*, 69:54–74.

List of Publications

Journal publications

- **Kapoor, S.**, Saxena, D.K. & van Leeuwen, M. Online summarization of dynamic graphs using subjective interestingness for sequential data. *Data Min Knowl Disc* **35**, 88–126 (2021).
  https://doi.org/10.1007/s10618-020-00714-8

- **Kapoor, S.**, Saxena, D.K. & van Leeuwen, M. Discovering subjectively interesting multigraph patterns. *Mach Learn* **109**, 1669–1696 (2020).
  https://doi.org/10.1007/s10994-020-05873-9

Conference Presentations:

- Accepted for presentation in ECML-PKDD 2021:
  Kapoor, S., Saxena, D.K. & van Leeuwen, M. Online summarization of dynamic graphs using subjective interestingness for sequential data. *Data Min Knowl Disc* **35**, 88–126 (2021). https://doi.org/10.1007/s10618-020-00714-8

Source-code published

- **Kapoor, S.** & van Leeuwen, M. Mining Subjective Subgraph Patterns. *Zenodo*. January 2021. doi:10.5281/zenodo.4483825.
  https://doi.org/10.5281/zenodo.4483825