11-8-2016

Large Scale Data Mining for IT Service Management

Chunqiu Zeng
Florida International University, czeng001@cs.fiu.edu

DOI: 10.25148/etd.FIDC001183

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

LARGE SCALE DATA MINING FOR IT SERVICE MANAGEMENT

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Chunqiu Zeng

2016

To: Interim Dean Ranu Jung
College of Engineering and Computing

This dissertation, written by Chunqiu Zeng, and entitled Large Scale Data Mining for IT Service Management, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Sundaraja Sitharama Iyengar

Jaime Leonardo Bobadilla

Wensong Wu

Shu-Ching Chen, Co-Major Professor

Tao Li, Co-Major Professor

Date of Defense: November 08, 2016

The dissertation of Chunqiu Zeng is approved.

Interim Dean Ranu Jung
College of Engineering and Computing

Andrés G. Gil
Vice President for Research and Economic Development
and Dean of University Graduate School

Florida International University, 2016

DEDICATION

I dedicate this dissertation work to my beloved family, especially my parents. Without their patience, understanding, support, or love, the completion of this work would not have been possible.

ACKNOWLEDGMENTS

It is the support from many people that brings me with the possibility to complete the dissertation for concluding my Ph.D. study.

First and foremost, I would like to express my sincerest thanks and appreciation to my advisor Dr. Tao Li, for his inimitable support, meticulous guidance, insightful advice, and getting me immersed with an excellent research atmosphere. Professor Tao Li, an eminent researcher in the community of data mining, is recognized as a knowledgeable and dedicated mentor. It is his great passion and endless patience that help me to locate my research interest and develop the related research skills in data mining area.

Second, I want to extend my gratitude to my co-advisor, Dr. Shu-Ching Chen, who has not only provided valuable guidance for me doing BCIN research project, but also given me constructive suggestion in developing my Ph.D. career.

Third, my thanks goes to all my dissertation committee members: Dr. S.S. Iyengar, Dr. Jaime Leonardo Bobadilla, and Dr. Wensong Wu, for their helpful advices, insightful comments on my dissertation research and future research career plans.

Fourth, I'd like to thank all my mentors including Dr. Larisa Shwartz, Dr. Genady Ya. Grabarnik, Mr. Jianfeng Yang, Mr. Jin Yi for my three internships: once at IBM and twice at Google. The patient guidance and help from them let me accumulate valuable experience and benefit me a lot in my Ph.D. study.

Moreover, I would like extend my thanks to our department staff for assisting me with the administrative tasks necessary during my doctoral study: Olga Carbonell, Carlos Cabrera, Steven Luis, Luis Rivera, etc.

Additionally, it's extremely fortunate for me to join Knowledge Discovery and Research Group (KDRG) where I have built up my research experience and enhanced my knowledge. I am very grateful to all my colleagues of KDRG including Dr. Liang Tang, Dr. Yexi Jiang, Dr. Li Zheng, Dr. Lei Li, Dr. Jingxuan Li, Dr. Chao Shen, Dr.

Longhui Zhang, Yahya Benhadda, Hongtai Li, Wubai Zhou, Wei Xue, Qing Wang, Shekoofeh Mokhtari, Wentao Wang, Ramesh Baral, Xiaolong Zhu and Boyuan Guan. Both valuable discussion and helpful suggestion from them continuously enlighten me with new insights into my research problems.

Finally, I would like to express my utmost gratitude to my parents and family, whose endless love and understanding are with me in whatever I pursue. Without the unlimited support from them, I would never go through any tough times in my life.

ABSTRACT OF THE DISSERTATION

LARGE SCALE DATA MINING FOR IT SERVICE MANAGEMENT

by

Chunqiu Zeng

Florida International University, 2016

Miami, Florida

Professor Tao Li, Co-Major Professor

Professor Shu-Ching Chen, Co-Major Professor

More than ever, businesses heavily rely on IT service delivery to meet their current and frequently changing business requirements. Optimizing the quality of service delivery improves customer satisfaction and continues to be a critical driver for business growth. The routine maintenance procedure plays a key function in IT service management, which typically involves problem detection, determination and resolution for the service infrastructure.

Many IT Service Providers adopt partial automation for incident diagnosis and resolution where the operation of the system administrators and automation operation are intertwined. Often the system administrators' roles are limited to helping triage tickets to the processing teams for problem resolving. The processing teams are responsible to perform a complex root cause analysis, providing the system statistics, event and ticket data. A large scale of system statistics, event and ticket data aggravate the burden of problem diagnosis on both the system administrators and the processing teams during routine maintenance procedures.

Alleviating human efforts involved in IT service management dictates intelligent and efficient solutions to maximize the automation of routine maintenance procedures. Three research directions are identified and considered to be helpful for IT service management optimization: (1) Automatically determine problem categories according to the symptom description in a ticket; (2) Intelligently discover interesting temporal patterns from system events; (3) Instantly identify temporal dependencies among system performance statistics data. Provided with ticket, event, and system performance statistics data, the three directions can be effectively addressed with a data-driven solution. The quality of IT service delivery can be improved in an efficient and effective way.

The dissertation addresses the research topics outlined above. Concretely, we design and develop data-driven solutions to help system administrators better manage the system and alleviate the human efforts involved in IT Service management, including (1) a knowledge guided hierarchical multi-label classification method for IT problem category determination based on both the symptom description in a ticket and the domain knowledge from the system administrators; (2) an efficient expectation maximization approach for temporal event pattern discovery based on a parametric model; (3) an online inference on time-varying temporal dependency discovery from large-scale time series data.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

More than ever, businesses heavily rely on IT service delivery to meet their current and frequently changing business requirements. Optimizing the quality of service delivery improves customer satisfaction and continues to be a critical driver for business growth. In order to optimize service quality, Service Providers seek to employ business intelligent solutions that provide deep analytical and automation capabilities for large scale IT service management [Log16b]. An efficient routine maintenance procedure plays a key function in service management, which typically involves problem detection, determination and resolution for the service infrastructure [MSGL09] [ZLSG14a] [ABD+07] [ZTL+14]. One of the ultimate goals in IT service management is to maximize the automation of its routine IT maintenance procedure.

The routine IT maintenance procedure for IT service providers, defining the IT activities such as problem detection, determination, diagnosis, and resolution, are prescribed by the Information Technology Infrastructure Library (ITIL) specification [Log16b]. A typical workflow of the IT routine maintenance is illustrated in Figure 1.1, where four stages are involved.

At the first stage, problem detection in the IT environment is realized by system monitoring. System monitoring, one important component in IT service management, is capable of tracking the states of a system by collecting system statistics information such as the CPU utilization, the memory usage, the number of data bytes written and read on the disk, the amount of data received and sent through the network, the sequence of requests and responses processed on an application server,

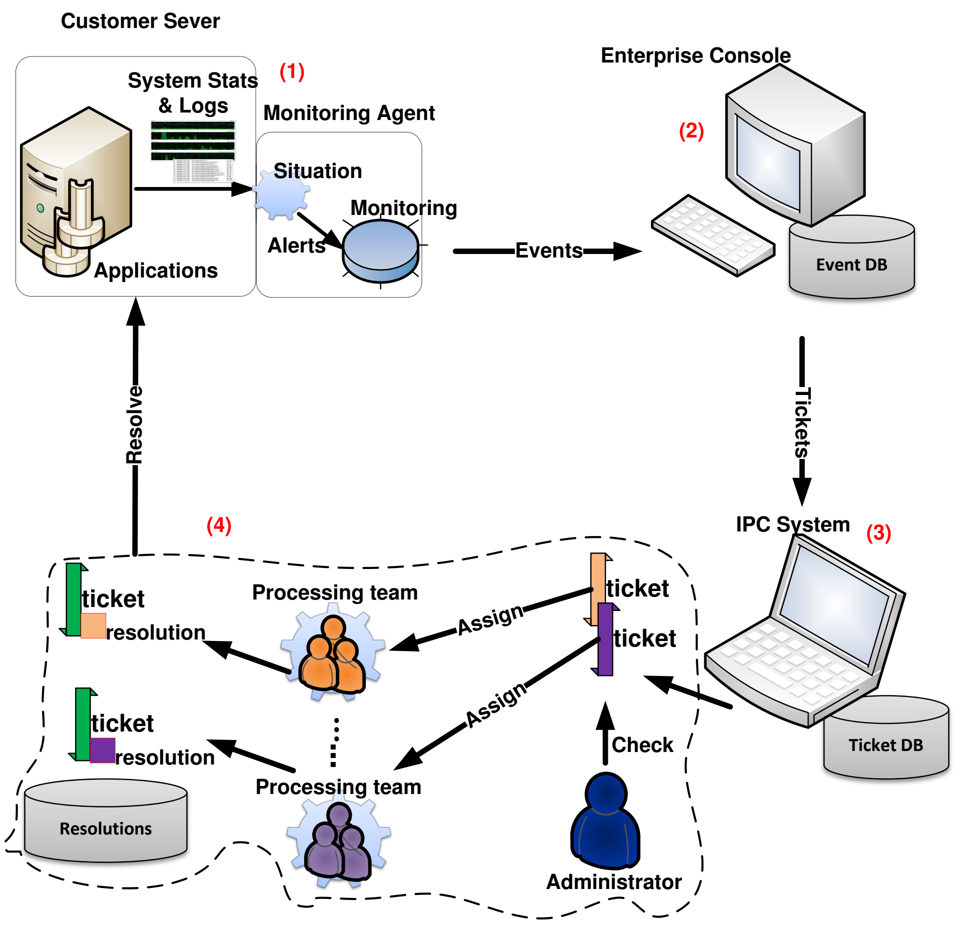
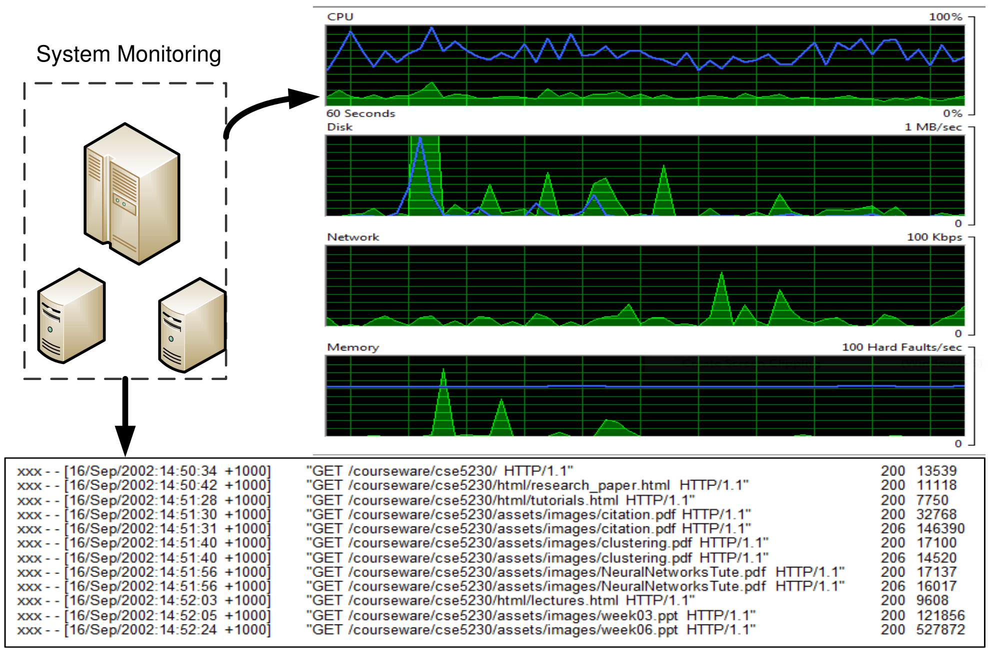


Figure 1.1: A typical IT routine maintenance procedure involves four stages.

etc. An example for the monitored data is illustrated in Figure 1.2, where performance information about both hardware and software is presented. Some popular system monitoring softwares are available on the market, encompassing IBM Tivoli Monitoring [IBM16], HP Open View [HPO], LogicMonitor [log16a], Zenoss [zen16], ManageEngine [Man16], and so on. The system monitoring computes metrics based on the regularly gathered system data and compares those metrics with some predefined acceptable thresholds, referred to as monitoring situations as well. Any violation after comparison raises an alert. If the alert persists beyond a certain duration specified in the situation, the monitoring emits an event.



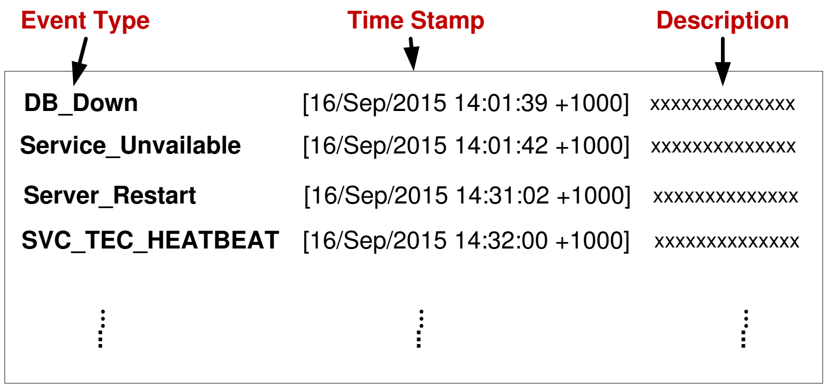
```
xxx - - [16/Sep/2002:14:50:34 +1000]    "GET /courseware/cse5230/ HTTP/1.1"                                          200  13539
xxx - - [16/Sep/2002:14:50:42 +1000]    "GET /courseware/cse5230/html/research_paper.html  HTTP/1.1"                 200  11118
xxx - - [16/Sep/2002:14:51:28 +1000]    "GET /courseware/cse5230/html/tutorials.html  HTTP/1.1"                      200  7750
xxx - - [16/Sep/2002:14:51:30 +1000]    "GET /courseware/cse5230/assets/images/citation.pdf HTTP/1.1"                200  32768
xxx - - [16/Sep/2002:14:51:31 +1000]    "GET /courseware/cse5230/assets/images/citation.pdf HTTP/1.1"                206  146390
xxx - - [16/Sep/2002:14:51:40 +1000]    "GET /courseware/cse5230/assets/images/clustering.pdf  HTTP/1.1"             200  17100
xxx - - [16/Sep/2002:14:51:40 +1000]    "GET /courseware/cse5230/assets/images/clustering.pdf  HTTP/1.1"             206  14520
xxx - - [16/Sep/2002:14:51:56 +1000]    "GET /courseware/cse5230/assets/images/NeuralNetworksTute.pdf  HTTP/1.1"     200  17137
xxx - - [16/Sep/2002:14:51:56 +1000]    "GET /courseware/cse5230/assets/images/NeuralNetworksTute.pdf  HTTP/1.1"     206  16017
xxx - - [16/Sep/2002:14:52:03 +1000]    "GET /courseware/cse5230/html/lectures.html  HTTP/1.1"                       200  9608
xxx - - [16/Sep/2002:14:52:05 +1000]    "GET /courseware/cse5230/assets/images/week03.ppt  HTTP/1.1"                 200  121856
xxx - - [16/Sep/2002:14:52:24 +1000]    "GET /courseware/cse5230/assets/images/week06.ppt  HTTP/1.1"                 200  527872
```

Figure 1.2: The system monitoring tracks the states of system. It presents performance information about both hardware (e.g., CPU, memory) and software (e.g., web server).

At the second stage, the generated events from the entire IT environment are consolidated in an enterprise console and archived in an event database. A snippet of event data set is shown in Figure 1.3, where each event is represented with its event type, occurring time stamp and description. The console employs rule, case or knowledge based engine to analyze the events and decide whether to report problems with a service ticket in the Incident, Problem, Change (IPC) system.

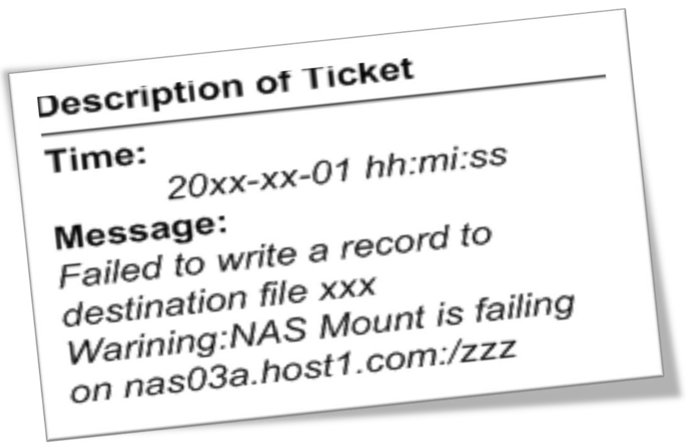
At the third stage, the reported tickets are stored in the ticket database of IPC system. A ticket example is illustrated in Figure 1.4. The information accumulated in the ticket describes the symptoms of the underlying problem and provides evidence for problem diagnosis, determination and resolution.

At the fourth stage, as a new ticket arrives, the system administrators inspect the ticket description, and infer the possible categories of the underlying IT problem based on their domain knowledge. The problem category inference further directs the ticket


Event Type
Time Stamp
Description

```
DB_Down              [16/Sep/2015 14:01:39 +1000]  xxxxxxxxxxxxxx
Service_Unvailable   [16/Sep/2015 14:01:42 +1000]  xxxxxxxxxxxxxx
Server_Restart       [16/Sep/2015 14:31:02 +1000]  xxxxxxxxxxxxxx
SVC_TEC_HEATBEAT     [16/Sep/2015 14:32:00 +1000]  xxxxxxxxxxxxxx
        ⋮                       ⋮                        ⋮
```

Figure 1.3: The event examples are presented. Each event is described with its event type, time stamp and description.



Description of Ticket

Time: *20xx-xx-01 hh:mi:ss*

Message:
Failed to write a record to destination file xxx Warining:NAS Mount is failing on nas03a.host1.com:/zzz

Figure 1.4: An ticket is taken as an example. The description of a ticket indicates the symptom of the underlying IT problem.

being assigned to proper processing teams for problem resolution, where different processing teams typically specialize in diverse IT problem categories. In general, the system administrators' role is limited to help triage tickets to the processing teams for problem resolving, while the processing teams are responsible to perform complex root cause analysis with respect to the related system performance statistics, event and ticket data. Finally, the service returns to be normal after problem resolving.

To sum up, the IT service management relies on partial automation of the IT routine maintenance procedure, where the operation of the system administrators and automation operation are intertwined. Among all the stages of the entire maintenance procedure, most of human efforts are invested during the fourth stage, where a labor-intensive and error-prone process for problem determination, diagnosis and resolution is conducted by both system administrators and processing teams.

1.2 Motivation and Problem Statement

Maximal automation of the IT routine maintenance procedure can alleviate the human effort investment, and thereby reduce the risk of human mistakes. The goal of IT service management optimization, effective and efficient delivery of IT service, can be achieved by maximizing the automation of the IT routine maintenance procedure.

IT service management optimization is urgently dictated in practice. Large and complex systems often aggravate the difficulty of service management, and increase the human labor involvement in the routine maintenance procedures. This procedure turns out to be extremely expensive, especially for those complex systems with changing environment. It has been reported that, in medium and large companies , anywhere from 30% to 70% of their information technology resources are used as maintenance cost [LPP+10]. High maintenance cost is attributed to several challenges summarized as below.

- The heterogeneous nature of the computing system requires more experts specializing in diverse domains. As a result, it makes the management task more complex and complicated. A typical computing system contains different devices (e.g., routers, CPU, GPU, and disks) with different software components (e.g., operating system, file system, databases, and user applications), possibly from different providers (e.g., Cisco, IBM, Google, and Microsoft). The heterogeneity increases the likelihood of unexpected interactions and poorly understood dependencies, and incurs more coordination cost for multiple experts with different domain knowledge.

- The scale and complexity of these systems probably causes a large number of unexpected behaviors during failures, system perturbations and even normal operations. A big number of complicated system behaviors greatly surpass what can be understood as to the system at the level of detail necessary for management, and are far beyond the processing capability for both the administrators and the processing teams.

- Current computing systems are undergoing a dynamic and rapid change with a growing number of software and hardware components. The fast rate of change worsens system dependability and exacerbates the difficulty of understanding system behaviors. It is extremely expensive, if not impossible, to instantly keep up with the current state of systems.

Driven by the challenges above, automatic and efficient solutions for complex system monitoring and management are pressingly demanded. Alleviating human efforts involved in IT service management dictates more intelligent and efficient solutions to maximize the automation of the routine maintenance procedures.

In recent years, data mining and machine learning techniques have acquired great interest to address the issues in system and service management [MSGL09, ABD+07, DJL09, LPG02, ABCM09, KWI+11, BO07, HMP02, LLMP05, MH01, TLS12]. These techniques are employed for efficiently extracting valuable knowledge from historical data produced during the entire IT maintenance procedure. In service management, the historical data includes the gathered system performance statistics, monitoring events and reported incident tickets, shown in Figure 1.5. Our work focuses on designing and implementing an integrated solution to extract valuable knowledge from

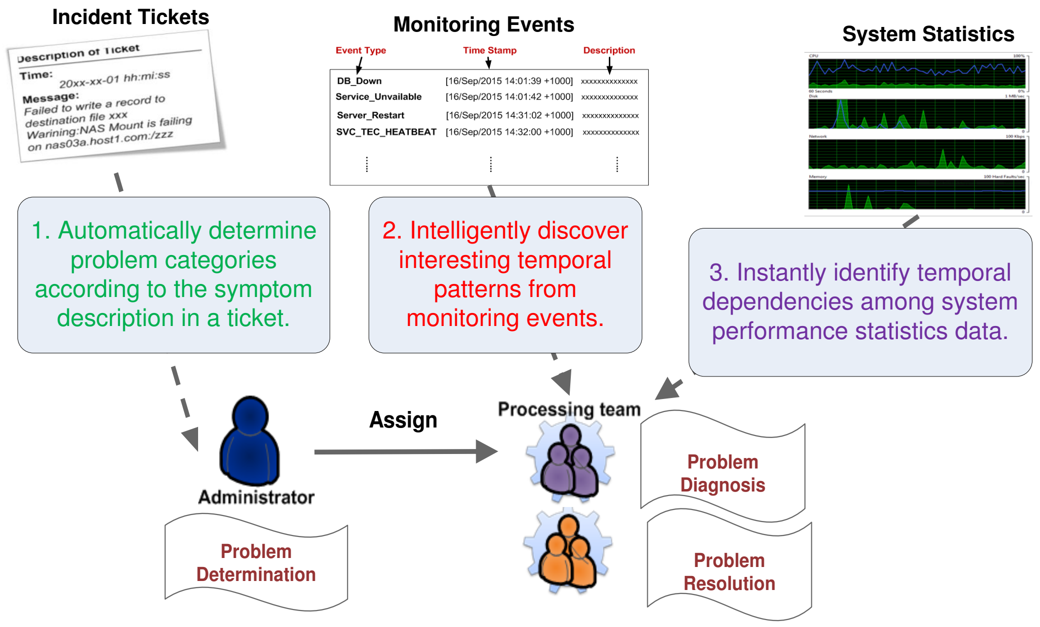

Incident Tickets
Description of Ticket
Time: 20xx-xx-01 hh:mi:ss
Message: Failed to write a record to destination file xxx Warining:NAS Mount is failing on nas03a.host1.com:/zzz
Monitoring Events
Event Type
Time Stamp
Description
DB_Down [16/Sep/2015 14:01:39 +1000]
Service_Unvailable [16/Sep/2015 14:01:42 +1000]
Server_Restart [16/Sep/2015 14:31:02 +1000]
SVC_TEC_HEATBEAT [16/Sep/2015 14:32:00 +1000]
System Statistics
1. Automatically determine problem categories according to the symptom description in a ticket.
2. Intelligently discover interesting temporal patterns from monitoring events.
3. Instantly identify temporal dependencies among system performance statistics data.
Administrator
Assign
Processing team
Problem Determination
Problem Diagnosis
Problem Resolution

Figure 1.5: Valuable knowledge, extracted from the historical data including incident tickets, monitoring events and system statistics data, facilitates problem determination, diagnosis and resolution.

the historical data and leverage the knowledge to facilitate the problem determination, diagnosis and resolution.

From the perspective of data mining, three research directions are identified and considered to be helpful for IT service management optimization.

1. ***Automatically determine problem categories according to the symptom description in a ticket.*** The symptom description of an IT problem is typically accumulated as a short text message, which is a combination of human and machine generated text with a very domain-specific vocabulary. In traditional IT maintenance procedure, the system administrators utilize their domain knowledge to identify the problem categories according to the short message in a ticket. This task is often recognized as addressing a text classification problem. Some existing work has been proposed to utilize data mining methods for automatic problem category determination, including support vec-

tor machine, classification and regression tree, k-nearest neighbors, rule-based classification, logistic regression [LPG02, DJL09, MBW14, KWI$^+$11]. However, an IT problem described in a ticket typically gets involved with multiple categories, where the categories are not independent from each other, but organized in a hierarchical relationship. Furthermore, the domain knowledge from the system administrators is valuable. Efficiently integrating the domain knowledge becomes increasingly important for problem category determination. All the issues pose new challenges on automatic ticket classification.

2. ***Intelligently discover interesting temporal patterns from system events.*** The generated events are consolidated in an enterprise console. An IT problem is reported with a service ticket in the IPC system after analyzing its related events. Arriving with a service ticket, One critical task of the processing team is to identify the root cause of the potential IT problem. There has been a great deal of effort spent on developing methodologies for root cause analysis in IT Service Management. One fruitful line of research has involved the development of techniques for traversing graphs dependencies of application configuration [ABCM09]. Although these methods have been successful in understanding the system's failures, they have had a limited impact due to overhead associated with constructing such graphs and keeping them up-to-date. Another approach has focused on the mining temporal event patterns [BO07, HMP02, LLMP05, LM04, MH01, TLS12]. The temporal event patterns are characterized with time lag, plays an important role in discovering the evolving trends of the upcoming events and helping with root cause analysis. However, mining temporal event patterns with fluctuating time lag among interleaving events is still a difficult task.

3. ***Instantly identify temporal dependencies among system performance statistics data.*** In order to instantly track the state of system, the system monitoring collects system statistics information such as CPU utilization, the memory usage, the number of data bytes written to and read from the disk, etc. The temporal dependencies among those system performance statistics are useful for problem diagnosis and root cause analysis. The discovered temporal dependencies are strong indicators that a fault of a particular component is highly correlated with the failure of its dependent components. The system performance statistics are time series data. To identify the latent temporal dependencies among the system performance statistics, existing methods on discovering Granger Causality [Gra80, ALA07] among time series can be applied. Most of existing work discovers Granger Causality from off-line data and assumes that the hidden Granger Casuality is stationary. However, in system management, instantly tracking the latent dependency among system performance statistics is critical and the stationary assumption rarely holds in practice. Therefore, online inference for time varying temporal dependency among system performance statistics is still a challenging problem.

Figure 1.5 summarizes the three research directions based on different types of historical data during the IT routine maintenance procedure, aiming at IT service management optimization. In the next section, the contributions of my dissertation along these research directions are briefly presented.

1.3 Contributions

My dissertation addresses the challenges relevant to the research topics outlined above, by designing and developing data-driven approaches, with the purpose of helping system administrators better manage the system and alleviate the human efforts involved in IT service management. Especially, the main outcomes of my dissertation are highlighted as follows: (1) a general knowledge guided hierarchical multi-label classification method for IT problem category determination, utilizing both the symptom description in a ticket and the domain knowledge from the system administrators; (2) a parametric model proposed for modeling temporal event patterns and an efficient expectation-maximization-based approach developed for model inference; (3) an online inference method for discovering the time varying temporal dependencies from the large-scale system statistics data. The detailed contributions for three research directions are provided in the reminder of Section 1.3.

1.3.1 Automatic IT Problem Category Determination

In light of the ticket description, system administrators determine the categories of the IT problem and triage the ticket to the corresponding processing teams for problem resolving. Automatic IT problem category determination acts as a critical part during the routine IT maintenance procedures. Our contributions related to this research direction are summarized as below.

1. In the real IT environment, IT problem categories are naturally organized in a hierarchy by specialization. Utilizing the category hierarchy, we come up with a hierarchical multi-label classification method to classify the monitoring tickets.
2. In order to find the most effective classification and minimize the cost caused by mistaken assignment, a novel contextual hierarchy (CH) loss is introduced in accordance with the problem hierarchy.

3. Consequently, an arising optimization problem is solved by a new greedy algorithm named GLabel. An extensive empirical study over ticket data was conducted to validate the effectiveness and efficiency of our method.

4. In practice, as well as the ticket instance itself, the knowledge from the domain experts, which partially indicates some categories the given ticket may or may not belong to, can also be leveraged to guide the hierarchical multi-label classification. Accordingly, in our dissertation, a multi-label inference with the domain expert knowledge is conducted on the basis of the given label hierarchy.

5. The experiment over the real ticket data demonstrates the great performance improvement, after incorporating the domain knowledge during the hierarchical multi-label classification.

1.3.2 Temporal Pattern Mining from Fluctuating Events

The significance of mining hidden temporal patterns from sequential event data is highlighted in many domains including system management, stock market analysis, climate monitoring, and more. Time lags, important features of temporal dependent patterns (i.e., temporal dependencies), characterize the temporal order among event occurrences. Mining time lags of temporal dependencies provides useful insights into the understanding of sequential data and predicting its evolving trend. Traditional methods mainly utilize the predefined time window to analyze the sequential items, or employ statistical techniques to identify the temporal dependencies from the sequential data. However, it is still a challenging task for existing methods to find the time lag of temporal dependencies in the real world, where time lags are fluctuating, noisy, and interleaved with each other. Our contributions along this research direction are summarized in the following.

1. In order to identify temporal dependencies with time lags in this setting, we come up with an integrated framework from both system and algorithm perspectives.

2. Specifically, a novel parametric model is introduced to model the noisy time lags for temporal dependencies discovery between events.

3. Based on the parametric model, an efficient expectation-maximization-based approach is proposed for time lag discovery with maximum likelihood.

4. Furthermore, we also contributes an approximation method for learning time lag to improve the scalability in terms of the number of events, without incurring significant loss of accuracy.

5. We have conducted extensive experiments on both synthetic and real data to illustrate the efficiency and effectiveness of the proposed approaches. The temporal dependency graph among events are demonstrated in the integrated system.

1.3.3 Temporal Dependency Discovery among Time Series

Large-scale time series data are prevalent across various application domains such as system management, biomedical informatics, social networks, finance. Temporal dependency discovery among time series performs an essential role in revealing the hidden interactions among components and provides a better understanding of complex systems. It has been explored in many applications such as neuroscience [SKX09], economics [ALA07], climate science [LLNM$^+$09], and microbiology [LALR09]. The inference of temporal dependencies among time series are typically categorized into two different frameworks: dynamic Bayesian network [Jen96, Mur02, SKX09] and

Granger causality [Gra69, Gra80, Gew82, ALA07]. My research is based on the Granger causality framework and the related contributions are listed as below.

1. Considering the sparsity of the temporal dependency structure among large-scale time series in real practice, we first formulate the problem from a Bayesian perspective.
2. Further, in order to capture dynamical temporal dependency typically occurring with real-world problems, we explicitly model the dynamical change as a random walk, resulting in time varying temporal dependency model.
3. Taking advantage of the Bayesian modeling, we develop an effective online inference algorithm using particle learning.
4. Extensive empirical studies on both the synthetic and real application time series data are conducted to demonstrate the effectiveness and the efficiency of the proposed method. A case study from real scenario shows the usefulness of our proposed method in practice.

1.4 Summary and Roadmap

Large and complex systems with a large number of heterogeneous components are difficult to monitor, manage and maintain. Traditional approaches to system management mainly rely on the knowledge from the domain experts, where the domain knowledge is used for composing operational rules, policies, and dependency models. However, those routine maintenance procedures are well known and experienced as a cumbersome, labor intensive, and error prone processes. In the dissertation, focusing on alleviating human effort involvement, we design and implement several data-driven approaches to optimize the IT service management.

To facilitate the reading and understanding the research problems, the organization of this dissertation is outlined as follows. First, we briefly presents the preliminaries and related work of the aforementioned three research directions in Chapter 2. To be continue, we study the problems related to these research directions in Chapter 3, Chapter 4 and Chapter 5, respectively. Particularly, in Chapter 3, the IT problem category determination is studied, where both the category hierarchy and domain knowledge are utilized. In Chapter 4, we focus on the temporal pattern discovery from fluctuating system events, where those patterns facilitate the root cause analysis for IT problems. In Chapter 5, we study the problem about how to instantly infer the time varying temporal dependencies among time series. Those temporal dependencies among time series help to tracking the states of complex system. Finally, in Chapter 6, we conclude the work of this dissertation and discuss the future work along our research.

CHAPTER 2

PRELIMINARIES AND RELATED WORK

This dissertation studies the concrete problems along the aforementioned three research directions in IT service management and the corresponding solutions are exhaustively discussed as well. In this chapter, we highlight existing literature studies that are related to our work in this dissertation. In particular, Section 2.1 reviews the existing work related to the problem determination as well as the relevant techniques such as text classification, multi-label classification, hierarchical multi-label classification. Section 2.2 introduces diverse types of temporal patterns used for representing knowledge from events, and further describes the corresponding methodologies to extracting these patterns. Section 2.3 presents existing literature of temporal dependency discovery among the time series, and surveys both offline and online techniques for inferring the underlying temporal relations from the observed time series.

2.1 Related Work of IT Incident Ticket Classification

The IT problem categories are determined by inspecting the corresponding ticket, where the symptom information is accumulated during the IT routine maintenance procedures. Therefore, the IT problem determination is typically achieved by classifying the IT incident ticket into different problem categories. Our work as to IT incident ticket classification is related to text classification, multi-label classification and hierarchical multi-label classification.

2.1.1 Text Classification

Text classification techniques are commonly applied to address problems in a wide variety of application domains [AZ12]. Popular relevant applications include news filter and organization [Lan95], document organization and retrieval [CDAR97], opinion

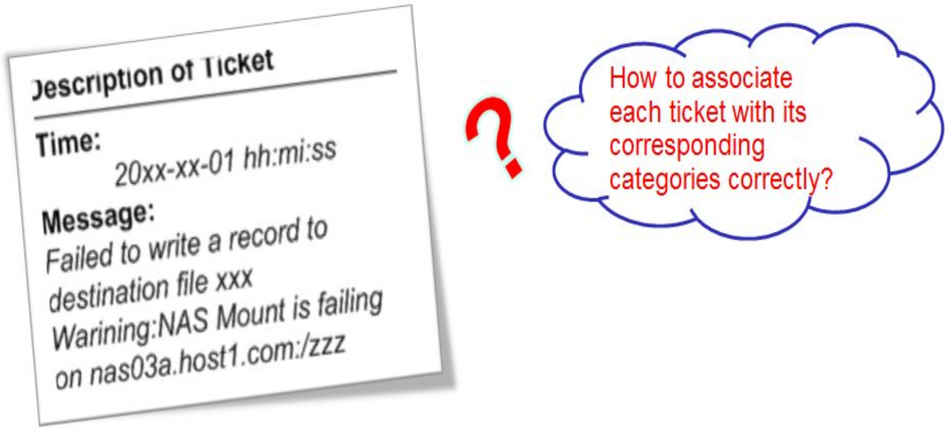

Description of Ticket
Time:
20xx-xx-01 hh:mi:ss
Message:
Failed to write a record to destination file xxx
Warining:NAS Mount is failing on nas03a.host1.com:/zzz
How to associate each ticket with its corresponding categories correctly?

Figure 2.1: The problem determination is referred to as a text classification problem based on the text content of a ticket.

mining [LZ12], email categorization and email spam filtering [CC05, CCM04, LK97, SDHH98], etc. In the scenario of IT service management, the symptom of the underlying IT problem is described by the text description of a ticket as shown in Figure 2.1. Accordingly, the problem determination adopts text classification techniques for IT problem categorization with respect to the ticket description [DJL09].

Text is characterized by its words, where the word attributes are typically sparse, high dimensional, and with low frequencies on most of the words. Therefore, one critical task is to represent text with appropriate features, which are most relevant to the classification process. For the purpose of classification, it is reasonable to utilize supervised feature selection methods, which take the class labels into account, to identify the most relevant features. Numerous feature selection methods for text categorization like Gini Index, Information Gain, Mutual Information, χ^2-Statistic are exhaustively discussed in [YP97, Yan95, AZ12]. As well as the feature selection methods, feature transformation approaches are also utilized for text classification improvement. The difference between them consists in that the former methods reduces the dimensionality of the data by picking some features from the original feature set, while the latter ones attempt to create a new set of features as a function of the original feature set. Popular feature transformation methods include Principal Component Analysis (abbr., PCA) [Jol02], Singular Value Decomposition (abbr., SVD) [HJP03, HP04], Latent Semantic Indexing (abbr., LSI) [DDF$^+$90], Probabilistic Latent Semantic Analysis (abbr., PLSI) [Hof99], Latent Dirichlet Allocation(abbr., LDA) [BNJ03], and so forth.

Provided with feature construction, another critical task is to design classification methods which effectively account for the characteristics of text. Various classification methods can be used for text classification. Classifiers based on decision tree utilize a condition on an attribute value to obtain a hierarchical decomposition of the entire data space. Typically, in the context of text data, the conditions indicate the presence or absence of one or more words in a document. The class label of the given text is predicted by traversing the decision tree from the root to a leaf node [Qui86]. A rule-based classifier employs a set of rules, generated from the training data, to model the mapping from the features to the class labels [Ma98]. The naive bayes classifier models the distribution of the documents in each class using a probabilistic model, assuming that the distribution of the features are independent from each other. The naive bayes classifiers are typically referred to as the most straightforward and commonly used generative classifiers [AZ12]. The linear classifiers separate the instances from different classes with a linear hyperplane, which are leant from the text data. Many linear classifiers can be used for text categorization, like Support Vector Machines (abbr., SVM) [CV95, Joa98, Vap13], Logistic Regression [Jor02], Neural Networks [LL99, RS99, WWP99, YL99], etc. The main idea of proximity-based classifiers is that the documents belonging to the same class are likely to be close to one another in terms of similarity measures [SM86]. K-Nearest Neighbors method is a proximity-based classifiers, where the class label of an instance is determined by the majority class of its K neighbors [CH98, HKK01, YC94]. In the past years, Meta-Algorithms have obtained much attention for classification strategies because of their ability to improve the performance of existing classification algorithms by combining them. Bagging, Stacking and Boosting belong to the category of meta-algorithms [BDH02, DHS12, HPS96, LL01, LC96, LJ98, YAP00].

2.1.2 Multi-Label Classification

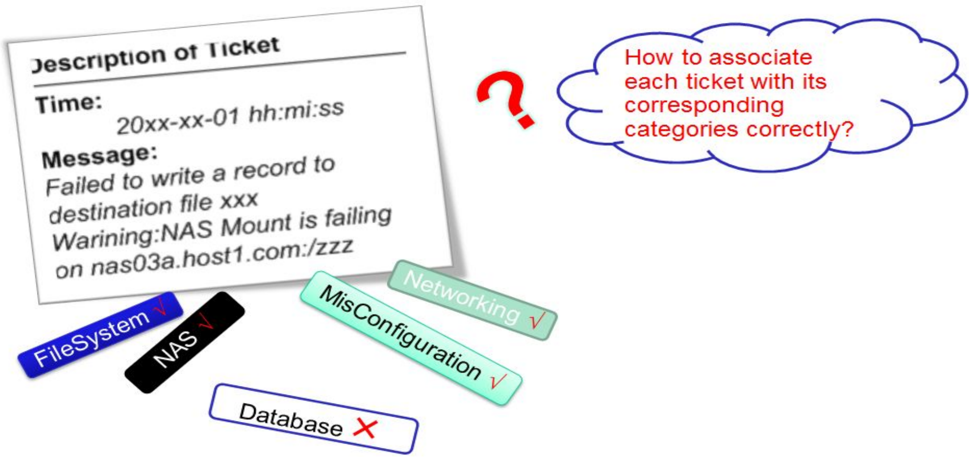

Description of Ticket
Time:
20xx-xx-01 hh:mi:ss
Message:
Failed to write a record to destination file xxx
Warining:NAS Mount is failing on nas03a.host1.com:/zzz
How to associate each ticket with its corresponding categories correctly?
FileSystem √
NAS √
MisConfiguration √
Networking √
Database ✕

Figure 2.2: Considering the fact that an IT problem may be associated with multiple labels, the problem determination is referred as a multi-label classification problem. The labels with $\surd$ are the categories of the underlying IT problem of the given ticket.

Traditional classification problems assume that each instance is associated with a single label. However, this assumption is not always true in practice. In the scenario of IT service management, a ticket may associate with multiple categories such as database problem, file system problem, networking problem as shown in Figure 2.2. The purpose of multi-label classification methods is to learn a mapping function which is capable of associating each instance with multiple class labels simultaneously [ZZ14, TK06].

The main challenge of the multi-label classification lies in the overwhelming size of output space, where the number of label sets grows exponentially as the number of class labels increases. To address the challenge, the correlation of labels is exploited to facilitate the classification process. Three strategies to capture the label correlation are explored in the literature. The simplest strategy takes each label independently and obtains a binary classification task per label [BLSB04, CK01, ZZ07]. Although it is straightforward to apply many existing binary classification algorithms in this strategy, it might not be optimal because of the ignorance of the label correlation. The second strategy accounts for the pairwise relations among labels, where the relations describe the ranking between relevant label and irrelevant label [EW01, FHMB08, ZZ06] and interaction between any two labels [GM05, QHR$^+$07, US02, ZJXG05]. The second strategy acquires a better generalization performance than the first one. The third strategy conducts multi-label classification by considering the relationship among all the labels, particularly by either imposing the influence of all other labels on each label [CH09, GS04, JTYY08, YTS07], or making use of the connections among a random subsets of labels [RPH08, RPHF11, TV07]. The third strategy is more capable of modeling the label correlations, while its related methods are more complex and less scalable than the methods related to the other two strategies.

The algorithms for learning the multi-label classification model are categorized into two groups, i.e., problem transformation methods and algorithm adaptation methods [ZZ14]. The former category of algorithms deal with the multi-label classification problems by transforming the original problems into the existing well-studied problems. Binary Relevance [BLSB04] and Classifier Chains [RPHF11, RPHF09] are proposed, where the multi-label classification task is converted into a set of binary classification tasks. In [FHMB08], the Calibrated Label Ranking method is developed by transforming the multi-label classification problem into a label ranking task. Random k-label sets [TV07] tackles the multi-label classification problems after transforming them into the multi-class classification problems. Some other existing methods of this category include Label Powerset [TKV09], a triple random ensemble multi-label classification [NKT10], constructing ensembles of pruned sets [RPH08], etc. The latter category of algorithms adapt the existing classification algorithms to address the multi-label classification problem directly. Based on the K Nearest Neighbor (abbr., KNN) algorithm, ML-kNN [ZZ07, WDH10, YADS11], as a lazy learning method is proposed. In [CK01], the ML-DT algorithm is acquired by adapting the decision tree technique. In light of the popular SVM algorithm, Rank-SVM [EW01, GS11] makes use of the kernel technique to address the multi-label classification problems. Utilizing the information theory, CML [GM05] is developed to handle the multi-label classification task directly.

Besides the algorithms as mentioned above, evaluation metrics act as inevitable parts in multi-label classification task. Performance evaluation in multi-label classification is more complicated than traditional classification problem. In [ZZ14], the evaluation metrics of multi-labels fall into two categories, i.e., example-based metrics and label-based metrics. Example-based metrics include Subset Accuracy, Hamming Loss, One-Error, Coverage, Ranking Loss, Average Precision, and so on [GM05, GS04, SS00, DWCH10, DWCH12, DKH12], while Macro-Averaging and Micro-Averaging belong to label-based metrics [TV07].

2.1.3 Hierarchical Multi-Label Classification

In the scenario of IT service management, the labels of IT problems are usually organized in a hierarchy as shown in Figure 2.3. Taking the hierarchy of labels into account, the problem determination is referred to as a hierarchical multi-label classification problem.

The hierarchical multi-label classification problem is a special multi-label classification problem, where all the labels are correlated and organized in a hierarchical structure. The hierarchical structure is a directed acyclic graph (abbr., DAG), where

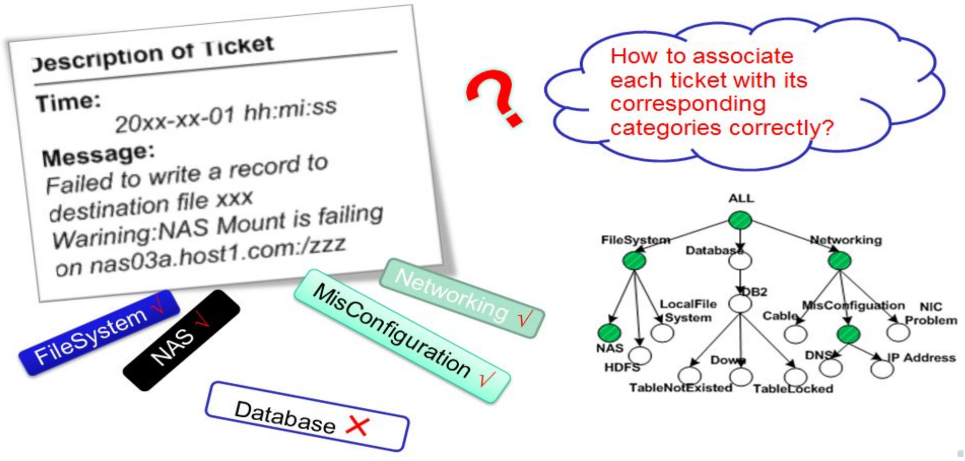


Figure 2.3: Accounting for the hierarchical relations among labels, the problem determination is referred to as a hierarchical multi-label classification problem. The labels corresponding to the green nodes in the hierarchy are related to the underlying IT problem.

a node represents a label and the directed edge typically denotes an Part-Of relation between two labels [Vat12, BST06]. The hierarchical structure can be classified into two groups: hierarchical tree and DAG structure. As a matter of fact, hierarchical tree is a special DAG structure. Each node of the hierarchical tree has one parent node at most, while the node in DAG may have more than one parent nodes. Hierarchical tree structure gains great popularity in the literature due to its simplicity. In hierarchical multi-label classification, Mandatory Leaf Node Problem (abbr., MLNP) is referred to the case where every instance is required to be associated with classes at the leaf nodes, otherwise the problem is called Non-Mandatory Leaf Node Problem (abbr., NMLNP) [BK12]. The hierarchical multi-label classification is generally defined as a task to label an instance with nodes belonging to more than one paths which may not end on leaf nodes in the hierarchy [ZLSG14a].

The hierarchical classification problem has been extensively investigated in the past decades [CBGZ06a, DC00, DKS05, DKS04, Gra03, HCC03, RS02, SL01]. The algorithms for hierarchical multi-label classification are summarized in [SJF11] and fall into three categories including flat classification approaches, global classification approaches and local classification approaches. The flat classification approaches ignore the class hierarchy and treat the original problem as a multi-label classification problem or a set of binary classification problems, where only the labels on the leaves are taken into account. As a result, the original problem can be solved by traditional classification methods such as neural network [JGB+02, WL04], decision tree, SVM, etc. However, this category of approaches can only handle MLNP classification problems, lacking in capability of addressing NMLMP classification problems [Vat12]. The global classification approaches take the entire hierarchy as input and learn a single classifier for all labels in the hierarchy. This category of approaches are usually developed by adapt existing classification algorithms (e.g., decision tree) and can capture all the dependencies of labels in the hierarchy, but the complexity of algorithms is increased [CK03, BSS+06, VSS+08, SVS+10]. The local classification approaches are the most common approaches for hierarchical multi-label classification problems because of their simplicity, efficiency. In this category, one of more classifier are learnt for every label node in the hierarchy and a post-process is applied to guarantee the hierarchical consistency [CBGZ06b, CBV10, BK11, WJ12]. We focus on the local classification approaches in the dissertation.

When considering hierarchical multi-label classification problem, adoption of a proper performance measure for a specific application domain is of the most importance. Zero-one loss and Hamming loss that were one of the first loss functions proposed for multi-label classification, are also commonly used in hierarchical multi-label classification problem [HZMVV10, CH04]. Taking hierarchy information into account, hierarchical loss (H-loss) has been proposed in [CBGZ06b]. The main idea is that any mistake occurring in a subtree does not matter if the subtree is rooted with a mistake as well. The HMC-loss [WJ12] loss function is proposed by weighting the misclassification with the hierarchy information while avoiding the deficiencies of the H-loss. It also differentiates the misclassification between the false negative (i.e., FN) and the false positive (i.e., FP) with different penalty costs.

Cesa-Bianchi et al. [CBGZ06b] introduce an incremental algorithm to apply a linear-threshold classifier for each node of the hierarchy with performance evaluation in terms of H-loss. Moreover, the Bayes-optimal decision rules are developed by Wei in [CBGZ06a]. And Cesa-Bianchi et al. [CBV10] extend the decision rules to the cost-sensitive learning setting by weighting false positive and false negative differently. Wei and James [WJ12] propose the HIROM algorithm to obtain the optimal decision rules with respect to HMC-loss by extending the CSSA algorithm in [BK11], which has a strong assumption that the number of classes related to each instance is known.

2.2 Related Work of Temporal Pattern Mining from Event Data

With the rapid development in data collection and storage technologies during last two decades, the discovery of hidden information from temporal data has gained great interest. Temporal data is a collection of data items associated with time stamps, describing the discrete or continual state changes, or evolving trends over time. In light of the types of item values, temporal data is categorized into two types. If the value of each item is continuous, the temporal data is referred to as time series data. By contrast, the temporal data is called event data if the item value is categorical. This section mainly studies the temporal pattern discovery from event data.

2.2.1 Temporal Data

Temporal data is collected and analyzed across widespread application domains. A typical application of temporal data collection and analysis is in system management [urla]. System monitoring, one of the important components in system management, periodically assesses the state of a system by collecting system information such as CPU utilization, memory usage, network connection status, file transfer and data transmission status, etc. All system information is collected with a fixed frequency, and each data item is recorded with its time stamp. Some preliminary temporal data analysis in system management is included in the event dependence analysis and in the root cause analysis. The prevalence in social network applications such as Twitter, Facebook and Linkedin also leads to a large scale of temporal data generated by millions of active users everyday. Examples from this domain are posts and articles related to certain events such as car accidents, earthquakes, national election campaigns and sport games. Time stamps are attached to those posts and articles. Besides the aforementioned two domains, temporal data can also be found in other application domains such as health care, stock market and climate.

A considerable amount of literature has been published on temporal pattern mining. Based on the type of targeted temporal pattern, we can roughly group those preliminary research studies into the following categories: *sequential pattern mining*, *dependent pattern mining*, *temporal correlation analysis* and others [ZL15]. Our work here falls into the category of *dependent pattern mining.*

2.2.2 Sequential Pattern Mining

The sequential pattern mining problem is first introduced by Agrawal and Srikant in [AS95], with the purpose of discovering frequent subsequences as patterns from a sequence database. It focuses on the patterns across different transactions by considering their sequential order. This differs from frequent pattern mining [AIS93] which is aiming at finding frequent item sets within each transaction. Two classes of approaches have been proposed to solve *sequential pattern mining.* They are *Apriori-Based*

Algorithms [IA12, SA96a, GRS99, Zak01, AFGY02a] and *Pattern-Growth-Based Algorithms* [HPMA$^+$00a, PHMAZ00, PHMA$^+$01a]. *Apriori-Based Algorithms* borrow the core ideas from classical Apriori algorithms and often require multiple scans of the sequence database with a high I/O cost. On the other hand, *Pattern-growth-based Algorithms* are capable of efficient memory management [IA12] since they utilize a data structure, usually a tree, to partition search space.

2.2.3 Dependent Pattern Mining

Mining temporal dependencies among events has been proven to provide essential improvement to enterprise system management as the discovered temporal dependencies used for tuning monitoring system configurations [MHPG02], [PTG$^+$03]. Prior works in the temporal dependency discovery use transactional data and algorithms such as GSP [SA96b], FreeSpan [HPMA$^+$00b], PrefixSpan [PHMA$^+$01b], and S-PAM [AFGY02b]. In our scenario no information is given to show what items belong to the same transaction; only the time stamp of items could be utilized as a basis for discovering the temporal dependencies from sequential data (items with time stamps and events are used interchangeably here). Several types of dependent pattern mining tasks have been induced from practical problems and carefully studied, such as *Fully Dependent Pattern* [LMH02], *Partially Periodic Dependent Pattern* [MH01], *Mutually Dependent Pattern* [MH01] and *T-Pattern* [HMP02, HCF95, LLMP05]. Based on the fact that potentially related items tend to happen within a certain time interval, some previous work of temporal mining focuses on frequent itemsets given a predefined time window [HHAI95]. However, it's difficult to determine a proper window size. A fixed time window fails to discover the temporal relationship longer than the window size. Setting the size of time window to a large number makes the problem intractable, due to the exponential complexity of finding frequent itemsets on the maximal number of items per transaction.

A temporal relationship is typically represented as a pair of items within a specific time lag, commonly denoted as $A \rightarrow_{[t_1,t_2]} B$. It means that an event B will happen within time interval $[t_1, t_2]$ after an event A occurs. A lot of work was devoted to finding such temporal dependencies characterized with time lag [MH01], [LLMP05], [LM04] and [TLS12]. However, their efficiencies will be compromised if time lag L is random. In this dissertation we successfully mine temporal dependency under the condition that the time lag L is random. We extract the probability distribution of L along with the dependent items. The lag probability distribution allows for more insights and flexibility than just a fixed interval. In our previous work, we encountered a challenge of checking a large number of possible time lags due to the complexity of combinatorial explosion (even though we used an optimized algorithm with pruning techniques [TLS12]). In the dissertation, we propose an EM-based approximation method to efficiently learn the distribution of time lag in temporal dependency discovery.

2.2.4 Temporal Correlation Analysis

Existing work in correlation analysis between continuous temporal data and discrete temporal data, i.e., event data, could be classified into three categories: correlation between two events, correlation between two time series and correlation between time series and event. Most aforementioned pattern mining studies belong to the correlation analysis between discrete temporal data. In [ALA07], A. Arnold and Y. Liu conducted an interesting work to construct a causal graph from multiple continues temporal data series using graphical modeling with concept of Granger causality [Gra69]. In [LLL$^+$14], a novel approach is proposed to identify the correlation between two different types of temporal data in three aspects: (a) determining the existence of correlation between the time series and events, (b) finding the time delay of the correlation, (c) identifying the monotonic effect describing whether the correlation is positive or negative. All of these works give profound insight and provide efficient approaches for correlation analysis.

Some other related works on temporal data analysis still have been excluded from the aforementioned categories, such as *Frequent Episode Mining* [MTV97], *Event Burst Detection* [Kle03] and *Rare Event Detection* [VM02]. Some works have explored complex event processing in [CM12, Luc08, WDR06, AC06, ZU99]. However, my dissertation focuses on the analysis of the temporal information associated with events.

2.3 Related Work of Temporal Dependency Discovery among Time Series

Time series data contain a series of continuous values associated with time stamps. Large scale time series data are prevalent across diverse application domains including system management, biomedical informatics, social networks, finance, climate, etc. In recent years, applying data mining techniques for time series analysis becomes increasingly popular and has been received more and more attention. In this section, we highlight existing literature studies that are related to our proposed approach for online temporal dependency inference from time series.

2.3.1 Temporal Causality Analysis

One of the major data mining tasks for time series data is to reveal the underlying temporal causal relationship among the time series. Currently, two popular approaches prevail in the literature for causal relationship inference from time series data. One is the Bayesian network inference approach [Hec98][Mur02][JYG$^+$03][SKX09], while the other approach is the Granger Causality [Gra69][Gra80][ALA07]. Comparing with Bayesian network, Granger Causality is more straightforward, robust and extendable. Our proposed method is more related to the approach based on Granger Causality.

Since Granger causality is originally defined for a pair of time series, the causal relationship identification among multivariate time series can not be addressed directly until the appearance of some pioneering work on combining the notion of Granger causality with graphical model [Eic06]. The Granger causality inference among multivariate time series is typically developed by two techniques, i.e., statistical significance test and Lasso-Granger [ALA07]. Lasso-Granger is more preferable due to its robust performance even in high dimensions [BL12]. Our method takes the advantage of Lasso-Granger, but conducts the inference from the Bayesian perspective in a sequential online mode, borrowing the idea of Bayesian Lasso [PC08]. However, most of these methods assume a constant dependency structure among time series.

In order to capture the dynamic temporal dependency typically happening in real practice, a hidden Markov model regression [LKJ09] and time-varying dynamic Bayesian network [SKX09] have been proposed. However, the number of hidden states in [LKJ09] and the decaying weights in [SKX09] are difficult to determine without any domain knowledge. Furthermore, both methods infer the underlying dependency structure in an off-line mode. In this dissertation, we explicitly model the dynamic changes of the underlying temporal dependencies and infer the model in an online manner.

2.3.2 Online Inference

Our proposed model makes use of sequential online inference to infer the latent state and learn unknown parameters simultaneously. Popular sequential learning methods include sequential monte carlo sampling [Hal62], and particle learning [CJLP10].

Sequential Monte Carlo (SMC) methods consist of a set of Monte Carlo methodologies to solve the filtering problem [DGA00]. It provides a set of simulation based methods for computing the posterior distribution. These methods allow inference of full posterior distributions in general state space models, which may be both nonlinear and non-Gaussian.

Particle learning provides state filtering, sequential parameter learning and smoothing in a general class of state space models [CJLP10]. Particle learning is for approximating the sequence of filtering and smoothing distributions in light of parameter uncertainty for a wide class of state space models. The central idea behind particle learning is the creation of a particle algorithm that directly samples from the particle approximation to the joint posterior distribution of states and conditional sufficient statistics for fixed parameters in a fully-adapted `resample-propagate` framework. We borrow the idea of particle learning for both latent state inference and parameter learning.

2.4 Summary

This chapter highlights the existing works in the literature, which are highly related to the three research directions of my dissertation, i.e., IT problem determination with ticket classification, temporal pattern mining from events and temporal dependency discovery from time series. For each research direction, both the related approaches and evaluation metrics are exhaustively surveyed.

CHAPTER 3

AUTOMATIC INCIDENT TICKET CLASSIFICATION

Maximal automation of routine IT maintenance procedures is an ultimate goal of IT service management. System monitoring, an effective and reliable means for IT problem detection, generates monitoring ticket. In light of the ticket description, system administrators determine the categories of the IT problem and triage the ticket to the corresponding processing teams for problem resolving. Automatic IT problem category determination acts as a critical part during the routine IT maintenance procedures. In practice, IT problem categories are naturally organized in a hierarchy by specialization.

Utilizing the category hierarchy, this chapter comes up with a hierarchical multi-label classification method to classify the monitoring tickets. In order to find the most effective classification, a novel contextual hierarchy (CH) loss is introduced in accordance with the problem hierarchy. Consequently, an arising optimization problem is solved by a new greedy algorithm named *GLabel*. An extensive empirical study over ticket data was conducted to validate the effectiveness and efficiency of our method.

Furthermore, as well as the ticket instance itself, the knowledge from the domain experts, which partially indicates some categories the given ticket may or may not belong to, can also be leveraged to guide the hierarchical multi-label classification. Accordingly, in this chapter, a multi-label inference with the domain expert knowledge is conducted on the basis of the given label hierarchy. The experiment demonstrates the great performance improvement by incorporating the domain knowledge during the hierarchical multi-label classification over the ticket data.

3.1 Introduction

3.1.1 Background

Changes in the economic environment force companies to constantly evaluate their competitive position in the market and implement innovative approaches to gain competitive advantages. Without solid and continuous delivery of IT services, no value-creating activities can be executed. Complexity of IT environments dictates usage of analytical approaches combined with automation to enable fast and efficient delivery of IT services. Incident management, one of the most critical processes in IT Service Management [urlb], aims at resolving the incident and quickly restoring the provision of services while relying on monitoring or human intervention to detect the malfunction of a component. Thus, it is essential to provide an efficient architecture for the IT routine maintenance.

A typical architecture of the IT routine maintenance is illustrated in Figure. 3.1, where four components are involved.

1. In the case of detection provided by a monitoring agent on a server, alerts are generated and, if the alert persists beyond a predefined delay, the monitor emits an event.
2. Events coming from an entire account IT environment are consolidated in an enterprise console, which analyzes the monitoring events and determines whether to create an incident ticket for IT problem reporting.
3. Tickets are collected by IPC (abbr. $\underline{\text{I}}$ncident, $\underline{\text{P}}$roblem and $\underline{\text{C}}$hange) system and stored in the ticket database [TLS+13].
4. A ticket accumulates the symptom description of an IT problem with a short text message and a time stamp provided. According to the description of a tick-

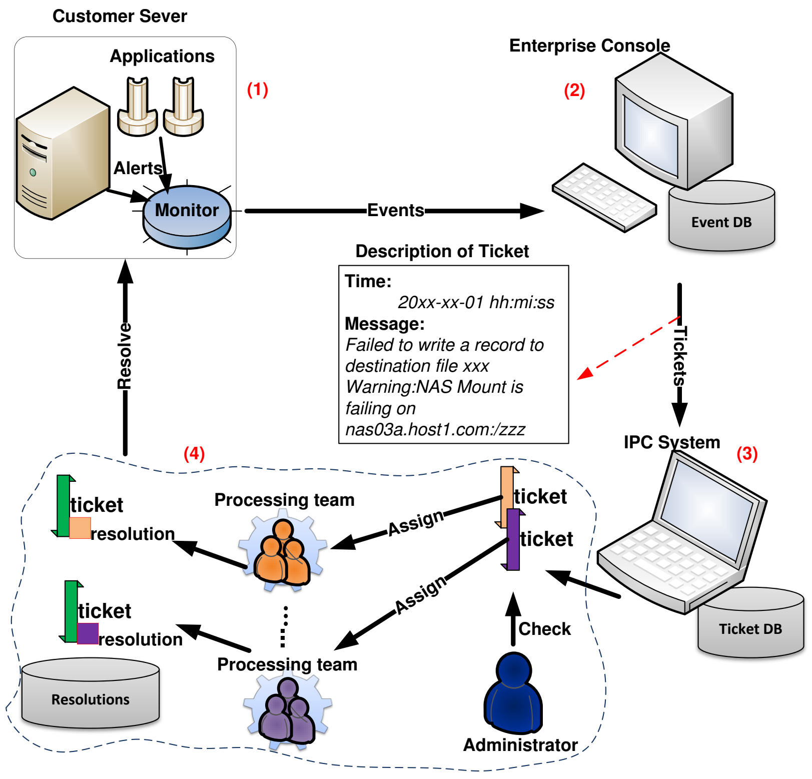


Figure 3.1: The overview of the IT routine maintenance procedure.

et, the system administrators (i.e., sysAdmins) perform the problem category determination and assign the ticket to its corresponding processing teams for problem diagnosis and resolution.

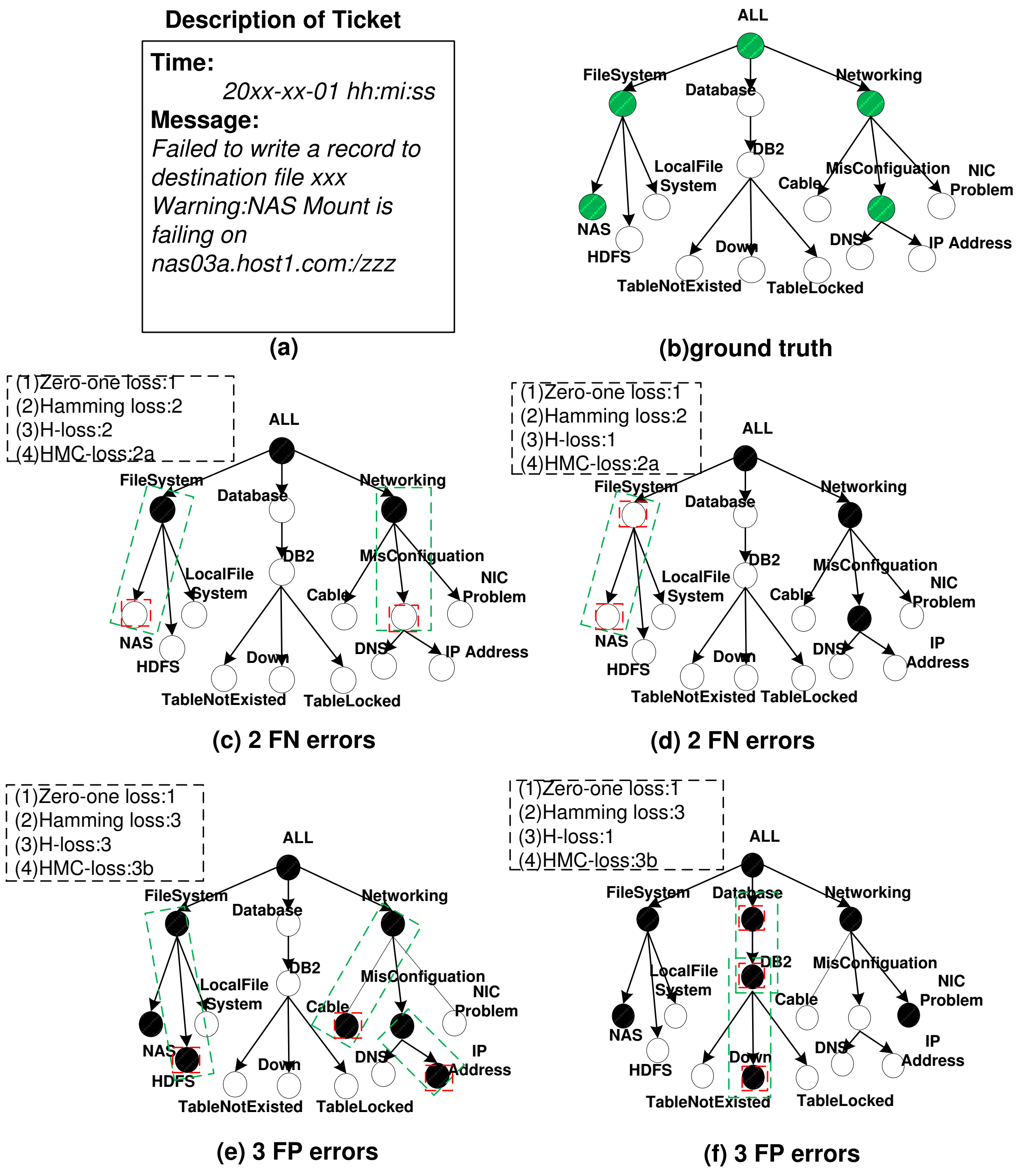
The last component gets involved with much labor-intensive effort to resolve each ticket.

The efficiency of these transient resources is critical for the provisioning of the services [JPLC12]. Many IT Service Providers rely on a partial automation for incident diagnosis and resolution, with an intertwined operation of the sysAdmins and an automation script. Often the sysAdmins' role is limited to executing a known remediation script, while in some scenarios the sysAdmin performs a complex root cause analysis. Removing the sysAdmin from the process completely, if it was feasible, would reduce human error and speed up restoration of service. The move from partially to fully automated problem remediation would elevate service delivery to a new qualitative level where automation is a complete and independent process, and where it is not fragmented due to the need for adapting to human-driven processes. However, the sysAdmin involvement is required due to the ambiguity of service incident description in a highly variable service delivery environment.

3.1.2 Motivation

In order to enhance the efficiency of the routine IT maintenance procedure, our work focuses on the labor-intensive component and tries to reduce human involvement by maximizing the automation of the problem category determination. In this chapter, we come up with a domain knowledge guided hierarchical multi-label classification method to facilitate the problem determination with both problem hierarchy preservation and domain knowledge integration from system administrators.

As shown in Figure 3.2.(a), a sample ticket describes a failure of an application to write data to NAS (Network-Attached Storage) [WIK16] file system. To identify a root cause of the problem, it is rational to limit a search space by classifying the incident tickets with their related class labels. Based on the message in Figure 3.2.(a) the ticket presents a problem related to FileSystem, NAS, Networking and Misconfiguration. Therefore, root cause analysis should be limited to four classes. Moreover, the collection of class labels is hierarchically organized according to the relationship among them. For example, as shown in Figure 3.2.(b), because NAS is a type of FileSystem, the label FileSystem is the parent of the label NAS in the hierarchy. This taxonomy could be built automatically ([DKFH12, LSLW12]) or it could be created with the help of domain experts [LL13]. In IT environments, hierarchical multi-label classification could be used not only for the problem diagnosis ([DKS04, CBGZ06b, CBV10, BK11]), but also for recommending resolutions


Description of Ticket

Time:
20xx-xx-01 hh:mi:ss
Message:
Failed to write a record to destination file xxx Warning:NAS Mount is failing on nas03a.host1.com:/zzz

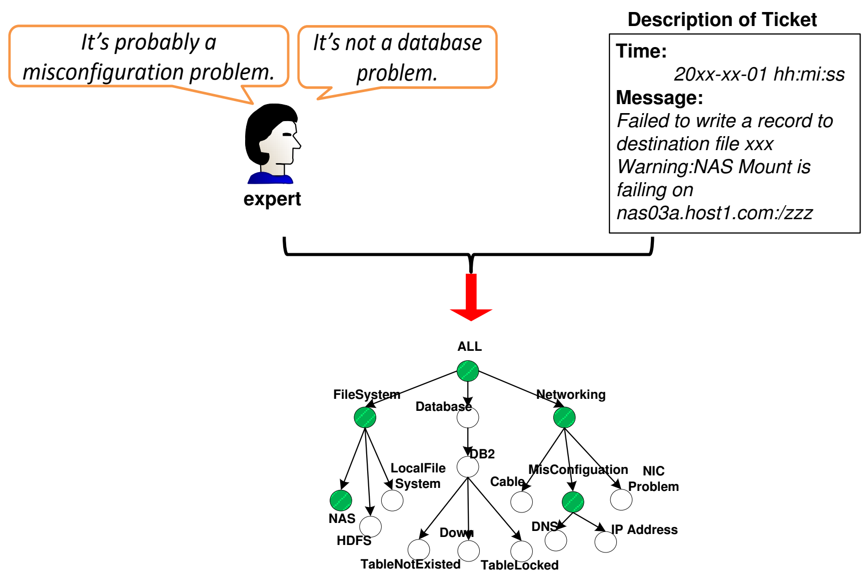
(c) 2 FN errors: (1)Zero-one loss:1 (2)Hamming loss:2 (3)H-loss:2 (4)HMC-loss:2a

(d) 2 FN errors: (1)Zero-one loss:1 (2)Hamming loss:2 (3)H-loss:1 (4)HMC-loss:2a

(e) 3 FP errors: (1)Zero-one loss:1 (2)Hamming loss:3 (3)H-loss:3 (4)HMC-loss:3b

(f) 3 FP errors: (1)Zero-one loss:1 (2)Hamming loss:3 (3)H-loss:1 (4)HMC-loss:3b

Figure 3.2: A hierarchical multi-label classification problem in the IT environment. A ticket instance is shown in (a). (b) presents the ground truth for the ticket with multiple class labels. (c), (d), (e) and (f) are four cases with misclassification. Assuming the cost of each wrong class label is 1, Zero-one loss, Hamming loss, H-loss, HMC-loss are given for misclassification. Notably, to calculate the HMC-loss, the cost weights for FN and FP are a and b respectively. The misclassified nodes are marked with a red square. The contextual misclassification information is indicated by the green rectangle.



Description of Ticket

Time: 20xx-xx-01 hh:mi:ss
Message: Failed to write a record to destination file xxx Warning:NAS Mount is failing on nas03a.host1.com:/zzz

Figure 3.3: Knowledge guided ticket hierarchical multi-label classification.

([TLSG13]) or auto-check scripts. The ticket in our example could have a solution that addresses FileSystem, NAS, Networking and/or Misconfiguration - a highly diversitified action recommendation. Furthermore, based on the hierarchical multi-label classification, actions with different levels in the hierarchy are recommended, where the actions from NAS category are more specific than the ones from FileSystem.

In real practice, as well as the text description of a given ticket, the prior knowledge from the domain experts is involved into the ticket classification by additional check (shown in Figure 3.1). For example, in Figure 3.3, given a ticket with its text description, the domain expert, based on his expertise in the system management, claims that the problem presented in the ticket is probably (e.g., with 60% confidence) a MisConfiguration problem, and definitely not a Database problem. Intuitively, the prior knowledge from the domain experts should also contribute to the ticket hierarchical multi-label classification for performance improvement. It is abrupt to employ the traditional hierarchical multi-label classification algorithms to deal with the ticket classification problem without taking any prior knowledge into account. However, the prior knowledge integration is not a trivial task in the hierarchical multi-label classification problem. The prior knowledge, about the likelihood that a given ticket should be assigned with a particular label, also contributes to the decision on whether the ticket belong to those class labels which are highly correlated to the particular label. For example, in Figure 3.3, given the ticket description, each label in the hierarchy is assigned with a probability to be positive. After inspecting the ticket description, the domain expert further confirm that this ticket is not MisConfiguration problem. Then the probabilities for labels DNS and IP Address being positive become 0, and the probability to be a Networking problem changes accordingly. It's challenging to determine the probability change for each label in the hierarchy, provided with the prior knowledge. To incorporate the prior knowledge, *Kilo* (**K**nowledge guided h**i**erarchical mutli-**l**abel classificati**o**n), a *sum-product* based algorithm, is proposed for hierarchical multi-label inference. Existing work in [CH07], takes the known hierarchical relationship between categories as knowledge and integrates the hierarchy for multi-label classification, where the knowledge is different from the one discussed in this chapter. The work of this chapter makes use of the prior knowledge, which partially indicates some categories that a given ticket may or may not belongs to. To the best of our knowledge, this is first work to utilize such prior knowledge to guide the hierarchical multi-label classification.

The Kilo algorithm is not only capable of fully exploring both domain knowledge and the data itself, but also provides an effective way for interactive hierarchical multi-label learning. Concretely, based on the current hierarchical multi-label classification result, the experts provide expertise to hint Kilo for further refinement. Kilo takes the hints from the experts as an input to refine the hierarchical multi-label inference. This iterative process continues until the domain experts are satisfied with the hierarchical multi-label classification result.

In this chapter we first define a new loss function, which takes the contextual misclassification information for each label into consideration and is a generalization of Hamming-loss, H-loss and HMC-loss function [WJ12]. Second, using Bayes decision theory, we develop the optimal prediction rule by minimizing the conditional risk in terms of proposed loss function. Next, knowledge from the experts are applied for hierarchical multi-label inference. Finally, we propose an efficient algorithm to search the optimal hierarchical multi-labels for each data instance.

The rest of this chapter is organized as follows. In Section 3.2, new loss function for better evaluation of the performance of the hierarchical multi-label classification is proposed and the knowledge from domain experts is formulated. In Section 3.3, the optimal prediction rule is derived with respect to our loss function. Section 3.4 describes the algorithm for hierarchical multi-label classification. Section 3.5 illustrates the empirical performance of our method. The last section is devoted to the conclusion of this chapter.

3.2 Hierarchical Multi-Label Classification

3.2.1 Problem Description

Let $\mathbf{x} = (x_0, x_1, ..., x_{d-1})$ be an instance from the d-dimensional input feature space χ, and $\mathbf{y} = (y_0, y_1, ..., y_{N-1})$ be the N-dimensional output class label vector where $y_i \in \{0, 1\}$. A multi-label classification assigns to a given instance $\mathbf{x}$ a multi-label vector $\mathbf{y}$, where $y_i = 1$ if $\mathbf{x}$ belongs to the ith class, and $y_i = 0$ otherwise. We denote the logical compliment of y_i by $\widetilde{y_i} = 1 - y_i$.

The hierarchical multi-label classification is a special type of multi-label classification when a hierarchical relation H is predefined on all class labels. The hierarchy

H can be a tree, or an arbitrary DAG (directed acyclic graph). For simplicity, we focus on H being the tree structure leaving the case of the DAG to future work.

In the label hierarchy H, each node i has a label $y_i \in \mathbf{y}$. Without loss of generality, we denote root node by 0, and its label by y_0. For each node i, let $pa(i)$ and $ch(i)$ be the parent and children nodes respectively of the node i . An indicator function I_e of a boolean expression e is defined as

$$I_e = \begin{cases} 1, & e\ is\ true; \\ 0, & e\ is\ false. \end{cases} \tag{3.1}$$

A hierarchical multi-label classification assigns an instance $\mathbf{x}$ an appropriate multi-label vector $\hat{\mathbf{y}} \in \{0,1\}^N$ satisfying the Hierarchy Constraint below.

Definition 3.2.1 (Hierarchy Constraint) *Any node i in the hierarchy* H *is labeled positive (i.e.,* 1*) if it is either the root node or its parent labeled positive. In other words,*

$$y_i = 1 \Rightarrow \{i = 0 \vee y_{pa(i)} = 1\}. \tag{3.2}$$

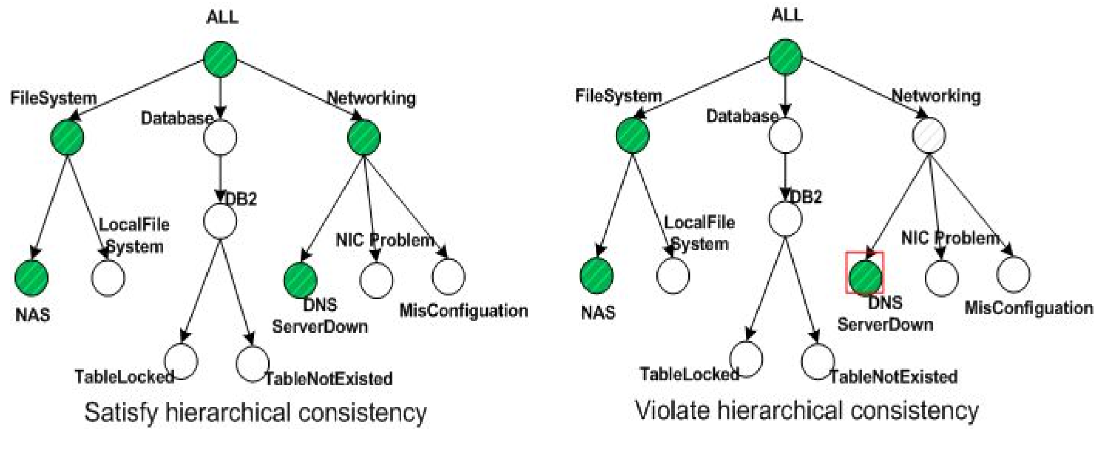


Figure 3.4: An example is given to illustrate the hierarchy constraint. Green and white nodes denote positive and negative, respective.

The hierarchy constraint is illustrated in Figure 3.4, where the labeled hierarchy at left satisfies the hierarchy constraint. The labeled hierarchy at right violates the hierarchy constraint since the DNS node is positive while its parent node Networking is negative.

Definition 3.2.2 (Knowledge) *Given an instance $\boldsymbol{x}$, the knowledge about the N-dimensional output class label vector $\boldsymbol{y}$ from the domain experts is represented by a N-dimensional vector $\boldsymbol{k}$. According to the domain knowledge, the i^{th} component of $\boldsymbol{k}$ can be assigned with 0, 1 and -1 for negative, positive and unknown respectively.*

Therefore, the knowledge guided hierarchical multi-label classification takes $\mathbf{x}$ and $\mathbf{k}$ as inputs, and outputs the class label vector $\mathbf{y}$.

3.2.2 Hierarchical Loss Function

When considering hierarchical multi-label classification problem, adoption of a proper performance measure for a specific application domain is of the most importance. Zero-one loss and Hamming loss, two of the first loss functions proposed for multi-label classification, are also commonly used in hierarchical multi-label classification problem [HZMVV10, CH04]. As shown in Figure 3.2, it is abrupt to adopt the zero-one loss measurement since all the imperfect predictions suffer the same penalty without any distinction. Although much more informative than zero-one loss, Hamming loss suffers a major deficiency since it does not incorporate hierarchy information. Comparing (c) and (d) in Figure 3.2, which both fail to label the ticket with NAS. The only difference between them is that, except the label NAS, (c) has a misclassification error with label MisConfiguration, while (d) gets a mistake with label FileSystem. Intuitively, the prediction in (d) should incur more penalty than one in (c) because a label FileSystem is at a higher level than label MisConfiguration in the hierarchy. However, the Hamming loss can not differentiate the two cases.

Taking hierarchy information into account, hierarchical loss (H-loss) has been proposed in [CBGZ06b]. The main idea is that any mistake occurring in a subtree does not matter if the subtree is rooted with a mistake as well. As illustrated in (f) of Figure 3.2, the H-loss only counts once for the label Database even though a mistake also takes place in label DB2 and Down (i.e., db2 is down). This idea is consistent with the scenario of problem diagnosis, since there is no need for further diagnosis in the successive children labels if the reason for the problem has already been excluded in the parent label. However, H-loss could be misleading. Considering the example (f) in Figure 3.2, after the solution related to Database is wrongly recommended, it is bad to refer the solutions belonging to the successive categories, such as DB2 and DOWN.

The HMC-loss [WJ12] function is proposed by weighting the misclassification with the hierarchy information while avoiding the deficiencies of the H-loss. It also differentiates the misclassification between the false negative (i.e., FN) and the false positive (i.e., FP) with different penalty costs. In Figure 3.2, assuming a and b are the misclassification penalties for FN and FP respectively, (c) and (d) have 2 FN misclassification errors, so both of them incur $2a$ HMC-loss. Moreover, (e) and (f) suffer $3b$ HMC-loss since they get 3 FP misclassification errors. However, HMC-loss fails to show the distinction between (c) and (d). In the scenario of the resolution recommendation, based on (c), more diverse solutions are recommended since the ticket is related to both FileSystem and Networking, while only the solutions related to Networking are considered as the solution candidates in (d). Moreover, HMC-loss can not differentiate predictions in (e) and (f). In the scenario of problem diagnosis, intuitively, we prefer (e) to (f) because the minor mistakes in multiple branches are not worse than the major mistakes in a single branch. Based on the discussion above, the main problem of HMC-loss is that it does not hierarchically consider the contextual information for each misclassification label (the contextual misclassification information is indicated with a green rectangle in Figure 3.2). Hence, we come up with a concept of the contextual information for each misclassified label.

We denote the prediction vector by $\hat{\mathbf{y}}$ and the ground truth by $\mathbf{y}$. To account for Hierarchy Constraint 3.2.1 as well as consider finding optimal prediction, we define Contextual Misclassification Information as follows.

Definition 3.2.3 (Contextual Misclassification Information) *Given a node i in hierarchy* H*, the contextual misclassification information depends on whether the parent node of i is misclassified when a misclassification error occurs in node i.*

There are four cases of misclassification of node i using Contextual Misclassification Information as shown in Figure 3.5.

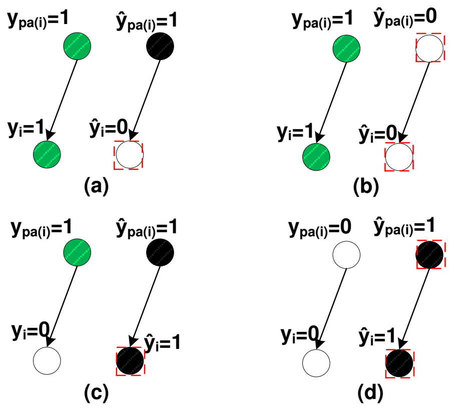


Figure 3.5: Four cases of contextual misclassification are shown in (a-d) for node i. Here the left pair is the ground truth; the right pair is the prediction. The misclassified nodes are marked with a red square.

We incorporate the following four cases of the contextual misclassification information into the loss function to solve the optimization problem, i.e. the best predicted value compatible with the hierarchy H.

- case (a): False negative error occurs in node i, while the parent node $pa(i)$ is correctly predicted.

- case (b): False negative error occurs in both node i and $pa(i)$.
- case (c): False positive error occurs in node i, while the parent node $pa(i)$ is correctly labeled with positive.
- case (d): Both node i and $pa(i)$ are labeled with false positive.

Referring to [WJ12],[CBV10], a misclassification cost C_i is given according to the position information of node i in the hierarchy H. And $\{w_i | 1 \leq i \leq 4\}$ are the different penalty costs for the above four cases, respectively. Accordingly, a new flexible loss function named CH-loss (<u>C</u>ontextual <u>H</u>ierarchical loss) is defined as follows:

$$\ell(\hat{\mathbf{y}}, \mathbf{y}) = w_1 \sum_{i>0}^{N-1} y_i y_{pa(i)} \widetilde{\hat{y}}_i \hat{y}_{pa(i)} C_i + w_2 \sum_{i>0}^{N-1} y_i y_{pa(i)} \widetilde{\hat{y}}_i \widetilde{\hat{y}}_{pa(i)} C_i + w_3 \sum_{i>0}^{N-1} \widetilde{y}_i y_{pa(i)} \hat{y}_i \hat{y}_{pa(i)} C_i + w_4 \sum_{i>0}^{N-1} \widetilde{y}_i \widetilde{y}_{pa(i)} \hat{y}_i \hat{y}_{pa(i)} C_i. \tag{3.3}$$

Next we show that the popular loss functions, such as HMC-loss, Hamming-loss and H-loss, are special cases of CH-loss function. We formulate the exact results below.

By setting α and β to be the penalty costs for false negative (FN) and false positive (FP) respectively, and noting that root node, indicating all categories, is always correctly labelled, the HMC-loss function defined in [WJ12] can be expressed as

$$\ell_{HMC}(\hat{\mathbf{y}}, \mathbf{y}) = \alpha \sum_{i>0}^{N-1} y_i \widetilde{\hat{y}}_i C_i + \beta \sum_{i>0}^{N-1} \widetilde{y}_i \hat{y}_i C_i. \tag{3.4}$$

Proposition 3.2.4 *The HMC-loss function is the special case of CH-loss function when $w_1 = w_2 = \alpha$ and $w_3 = w_4 = \beta$.*

Proof. The HMC-loss function can be rewritten as follows.

$$\begin{aligned}\ell_{HMC}(\hat{\mathbf{y}},\mathbf{y}) = {} & \alpha\sum_{i>0}^{N-1} y_i y_{pa(i)} \widetilde{\hat{y}}_i \hat{y}_{pa(i)} C_i + \alpha\sum_{i>0}^{N-1} y_i \widetilde{y}_{pa(i)} \widetilde{\hat{y}}_i \hat{y}_{pa(i)} C_i \\ & + \alpha\sum_{i>0}^{N-1} y_i y_{pa(i)} \widetilde{\hat{y}}_i \widetilde{\hat{y}}_{pa(i)} C_i + \alpha\sum_{i>0}^{N-1} y_i \widetilde{y}_{pa(i)} \widetilde{\hat{y}}_i \widetilde{\hat{y}}_{pa(i)} C_i \\ & + \beta\sum_{i>0}^{N-1} \widetilde{y}_i y_{pa(i)} \widetilde{\hat{y}}_i \hat{y}_{pa(i)} C_i + \beta\sum_{i>0}^{N-1} \widetilde{y}_i \widetilde{y}_{pa(i)} \hat{y}_i \hat{y}_{pa(i)} C_i \\ & + \beta\sum_{i>0}^{N-1} \widetilde{y}_i y_{pa(i)} \hat{y}_i \widetilde{\hat{y}}_{pa(i)} C_i + \beta\sum_{i>0}^{N-1} \widetilde{y}_i \widetilde{y}_{pa(i)} \hat{y}_i \widetilde{\hat{y}}_{pa(i)} C_i.\end{aligned}$$

Based on expression (3.2), we can derive the following expression of HMC-loss by removing the terms that violate the Hierarchy Constraint.

$$\begin{aligned}\ell_{HMC}(\hat{\mathbf{y}},\mathbf{y}) = {} & \alpha\sum_{i>0}^{N-1} y_i y_{pa(i)} \widetilde{\hat{y}}_i \hat{y}_{pa(i)} C_i + \alpha\sum_{i>0}^{N-1} y_i y_{pa(i)} \widetilde{\hat{y}}_i \widetilde{\hat{y}}_{pa(i)} C_i \\ & + \beta\sum_{i>0}^{N-1} \widetilde{y}_i y_{pa(i)} \hat{y}_i \hat{y}_{pa(i)} C_i + \beta\sum_{i>0}^{N-1} \widetilde{y}_i \widetilde{y}_{pa(i)} \hat{y}_i \hat{y}_{pa(i)} C_i.\end{aligned}$$

The equation above is the exact one when w_1,w_2,w_3 and w_4 in equation (3.3) are substituted with α, α, β and β respectively. □

Proposition 3.2.5 *The Hamming-loss function is the special case of CH-loss function when $w_1 = w_2 = w_3 = w_4 = 1$ and $C_i = 1$.*

It is established in [WJ12] that the Hamming-loss function is a special case of HMC-loss when $\alpha = \beta = 1$ and $C_i = 1$. Combining the result with the Proposition 3.2.4, the Proposition 3.2.5 is obvious.

The H-loss function (see [WJ12]) cannot be reduced to HMC-loss function, while H-loss is a special case of CH-loss function. Remember that the H-loss function is defined in [CBGZ06b] as follows:

$$\ell_H(\hat{\mathbf{y}},\mathbf{y}) = \sum_{i>0}^{N-1} I_{\hat{y}_i \neq y_i} I_{\hat{y}_{pa(i)} = y_{pa(i)}} C_i. \tag{3.5}$$

Proposition 3.2.6 *The H-loss function is the special case of CH-loss function when* $w_1 = 1$, $w_2 = 0$, $w_3 = 1$ *and* $w_4 = 0$.

Proof. H-loss defined in (3.5) is equivalent to the following equation:

$$\begin{aligned}\ell_H(\hat{\mathbf{y}}, \mathbf{y}) = &\sum_{i>0}^{N-1} y_i y_{pa(i)} \widetilde{\hat{y}}_i \hat{y}_{pa(i)} C_i + \sum_{i>0}^{N-1} \widetilde{y}_i y_{pa(i)} \hat{y}_i \hat{y}_{pa(i)} C_i \\ &+ \sum_{i>0}^{N-1} y_i \widetilde{y}_{pa(i)} \widetilde{\hat{y}}_i \widetilde{\hat{y}}_{pa(i)} C_i + \sum_{i>0}^{N-1} \widetilde{y}_i \widetilde{y}_{pa(i)} \hat{y}_i \widetilde{\hat{y}}_{pa(i)} C_i .\end{aligned}$$

Based on expression (3.2), the following expression of H-loss is derived by removing the terms which violate the Hierarchy Constraint:

$$\ell_H(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{i>0}^{N-1} y_i y_{pa(i)} \widetilde{\hat{y}}_i \hat{y}_{pa(i)} C_i + \sum_{i>0}^{N-1} \widetilde{y}_i y_{pa(i)} \hat{y}_i \hat{y}_{pa(i)} C_i .$$

The equation above is the exact equation (3.3) when $w_1 = 1$, $w_2 = 0$, $w_3 = 1$ and $w_4 = 0$. □

We summarize special cases of CH-loss in the Table 3.1.

Goal	**CH-loss parameter settings**
Minimize Hamming loss	$w_1 = w_2 = w_3 = w_4 = 1$, $C_i = 1$
Minimize HMC-loss	$w_1 = w_2 = \alpha$, $w_3 = w_4 = \beta$, C_i is defined by user
Minimize H-loss	$w_1 = w_3 = 1$, $w_2 = w_4 = 0$, $C_i = 1$
Increase recall	w_1 and w_2 are larger than w_3 and w_4
Increase precision	w_3 and w_4 are larger than w_1 and w_2
Minimize misclassification errors occur in both parent and children nodes	$w_2 > w_1$ and $w_4 > w_3$

Table 3.1: special cases of CH-loss

3.3 Expected Loss Minimization

In this section we use the previously defined CH-loss function to predict $\hat{\mathbf{y}}$ given instance $\mathbf{x}$ by minimizing expected CH-loss. Let $\mathbf{y}$ be the true multi-label vector of $\mathbf{x}$, and $P(\mathbf{y}|\mathbf{x})$ be the conditional probability that $\mathbf{y}$ holds given $\mathbf{x}$. The expected loss of labeling $\mathbf{x}$ with $\hat{\mathbf{y}}$ is defined by the following equation:

$$LE(\hat{\mathbf{y}}, \mathbf{x}) = \sum_{\mathbf{y} \in \{0,1\}^N} \ell(\hat{\mathbf{y}}, \mathbf{y}) P(\mathbf{y}|\mathbf{x}). \tag{3.6}$$

Let $\hat{\mathbf{y}}^*$ be (one of) the optimal multi-label vector(s) that minimizes expected CH-loss. Based on Bayesian decision theory, the problem is described as follows:

$$\begin{aligned} &\hat{\mathbf{y}}^* = \arg\min_{\hat{\mathbf{y}} \in \{0,1\}^N} LE(\hat{\mathbf{y}}, \mathbf{x}) \\ &\text{s.t. } \hat{\mathbf{y}} \text{ satisfies the hierarchy constraint 3.2.1.} \end{aligned} \tag{3.7}$$

The key step in solving the problem (3.7) consists in how to estimate $P(\mathbf{y}|\mathbf{x})$ in equation (3.6) from the training data. By following the work in [CBGZ06b, CBV10, WJ12], in order to simplify the problem, we assume that all the labels in the hierarchy are conditionally independent from each other given the labels of their parents. Since all the data instances are labeled positive at root node 0, we assume that $P(y_0 = 1|\mathbf{x}) = 1$ and $P(y_0 = 0|\mathbf{x}) = 0$. Due to an independency assumption we have:

$$P(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{N-1} P(y_i|y_{pa(i)}, \mathbf{x}). \tag{3.8}$$

Thus to estimate $P(\mathbf{y}|\mathbf{x})$, we need to estimate $P(y_i|y_{pa(i)}$ for each node i. The node-wise estimation may be done by utilizing binary classification algorithms, such as logistic regression or support vector machine. To deal with a significant computational load of the node-wise estimation, we parallelize the calculation. The details of the parallelization step are discussed in the next section.

The hierarchy constraint implies that $P(y_i = 1|y_{pa(i)} = 0) = 0$ and $P(y_i = 1|\mathbf{x}) = P(y_i = 1, y_{pa(i)} = 1|\mathbf{x})$. In order to simplify the notation, we denote:

$$p_i = P(y_i = 1|\mathbf{x}) = P(y_i = 1, y_{pa(i)} = 1|\mathbf{x}). \tag{3.9}$$

Then p_i can be computed based on $P(y_i = 1|y_{pa(i)} = 1, \mathbf{x})$ as:

$$p_i = P(y_i = 1|\mathbf{x}) = P(y_i = 1|y_{pa(i)} = 1, \mathbf{x})p_{pa(i)}. \tag{3.10}$$

By combining the definition of CH-loss with equations (3.6) and (3.9), the computation of loss expectation $LE(\hat{\mathbf{y}}, \mathbf{x})$ can be rewritten using p_i notation as follows:

Proposition 3.3.1 (Expected Loss)

$$\begin{aligned} LE(\hat{\boldsymbol{y}}, \boldsymbol{x}) = w_1 \sum_{i>0}^{N-1} \widetilde{\hat{y}}_i \hat{y}_{pa(i)} C_i p_i + w_2 \sum_{i>0}^{N-1} \widetilde{\hat{y}}_i \widetilde{\hat{y}}_{pa(i)} C_i p_i + \\ w_3 \sum_{i>0}^{N-1} \hat{y}_i \hat{y}_{pa(i)} C_i (p_{pa(i)} - p_i) + w_4 \sum_{i>0}^{N-1} \hat{y}_i \hat{y}_{pa(i)} C_i (1 - p_{pa(i)}). \end{aligned} \tag{3.11}$$

Proof. Combining both equation (3.3) and equation (3.6), we get:

$$LE(\hat{\mathbf{y}}, \mathbf{x}) = T_1 + T_2 + T_3 + T_4,$$

where

$$\begin{aligned} T_1 &= \sum_{\mathbf{y}} (w_1 \sum_{i>0}^{N-1} y_i y_{pa(i)} \widetilde{\hat{y}}_i \hat{y}_{pa(i)} C_i) P(\mathbf{y}|\mathbf{x}), \\ T_2 &= \sum_{\mathbf{y}} (w_2 \sum_{i>0}^{N-1} y_i y_{pa(i)} \widetilde{\hat{y}}_i \widetilde{\hat{y}}_{pa(i)} C_i) P(\mathbf{y}|\mathbf{x}), \\ T_3 &= \sum_{\mathbf{y}} (w_3 \sum_{i>0}^{N-1} \widetilde{y}_i y_{pa(i)} \hat{y}_i \hat{y}_{pa(i)} C_i) P(\mathbf{y}|\mathbf{x}), \\ T_4 &= \sum_{\mathbf{y}} (w_4 \sum_{i>0}^{N-1} \widetilde{y}_i \widetilde{y}_{pa(i)} \hat{y}_i \hat{y}_{pa(i)} C_i) P(\mathbf{y}|\mathbf{x}). \end{aligned}$$

we split the proof into four parts:

(a) T_1 can be written as

$$T_1 = w_1 \sum_{i>0}^{N-1} \widetilde{\hat{y}}_i \hat{y}_{pa(i)} C_i \sum_{\mathbf{y}} P(y_i = 1, y_{pa(i)} = 1, \mathbf{y} | \mathbf{x}).$$

Easily, we get:

$$T_1 = w_1 \sum_{i>0}^{N-1} \widetilde{\hat{y}}_i \hat{y}_{pa(i)} C_i P(y_i = 1, y_{pa(i)} = 1 | \mathbf{x}).$$

$$\therefore T_1 = w_1 \sum_{i>0}^{N-1} \widetilde{\hat{y}}_i \hat{y}_{pa(i)} C_i p_i.$$

(b) Following (a), we can obtain:

$$T_2 = w_2 \sum_{i>0}^{N-1} \widetilde{\hat{y}}_i \widetilde{\hat{y}}_{pa(i)} C_i p_i.$$

(c) Following (a), we have:

$$T_3 = w_3 \sum_{i>0}^{N-1} \hat{y}_i \hat{y}_{pa(i)} C_i P(y_i = 0, y_{pa(i)} = 1 | \mathbf{x}).$$

Since $P(y_i = 0, y_{pa(i)} = 1 | \mathbf{x}) = P(y_{pa(i)} = 1 | \mathbf{x})$

$- P(y_i = 1, y_{pa(i)} = 1 | \mathbf{x}) = p_{pa(i)} - p_i$, then

$$T_3 = w_3 \sum_{i>0}^{N-1} \hat{y}_i \hat{y}_{pa(i)} C_i (p_{pa(i)} - p_i).$$

(d) Following (c),

$$T_4 = w_4 \sum_{i>0}^{N-1} \hat{y}_i \hat{y}_{pa(i)} C_i P(y_i = 0, y_{pa(i)} = 0 | \mathbf{x}).$$

Since $P(y_i = 0, y_{pa(i)} = 0 | \mathbf{x}) = 1 - P(y_{pa(i)} = 1 | \mathbf{x})$

$= 1 - p_{pa(i)}$, then:

$$T_4 = w_4 \sum_{i>0}^{N-1} \hat{y}_i \hat{y}_{pa(i)} C_i (1 - p_{pa(i)}).$$

After substituting T_1, T_2, T_3, T_4 into $LE(\hat{\mathbf{y}}, \mathbf{y})$, it is the exact equation (3.11). $\square$

Based on the Expected Loss described in equation (3.11), the problem (3.7) is re-formulated as follows:

Proposition 3.3.2 *The minimization problem (3.7) is equivalent to the maximization problem below.*

$$\begin{aligned} &\hat{\boldsymbol{y}}^* = \underset{\hat{y} \in \{0,1\}^N}{\arg\max}\, LE_\delta(\hat{\boldsymbol{y}}, \boldsymbol{x}) \\ &\textit{s.t. } \hat{\boldsymbol{y}} \textit{ satisfies the hierarchy constraint.} \end{aligned} \tag{3.12}$$

where

$$LE_\delta(\hat{\boldsymbol{y}}, \boldsymbol{x}) = \sum_{i>0}^{N-1} \hat{y}_{pa(i)}(w_2 - w_1)C_i p_i + \\ \sum_{i>0}^{N-1} \hat{y}_i[w_1 C_i p_i - w_3 C_i(p_{pa(i)} - p_i) - w_4 C_i(1 - p_{pa(i)})].$$

Proof. Equation (3.11) is equivalent to:

$$\begin{aligned} LE(\hat{\mathbf{y}}, \mathbf{x}) = w_1 \sum_{i>0}^{N-1} (\hat{y}_{pa(i)} - \hat{y}_i)C_i p_i + w_2 \sum_{i>0}^{N-1} (1 - \hat{y}_{pa(i)})C_i p_i \\ + w_3 \sum_{i>0}^{N-1} \hat{y}_i C_i (p_{pa(i)} - p_i) + w_4 \sum_{i>0}^{N-1} \hat{y}_i C_i (1 - p_{pa(i)}) \end{aligned}$$

$$\Rightarrow LE(\hat{\mathbf{y}}, \mathbf{x}) = \sum_{i>0}^{N-1} w_2 C_i p_i - (\sum_{i>0}^{N-1} y_{pa(i)}(w_2 - w_1)C_i p_i + \\ \sum_{i>0}^{N-1} y_i[w_1 C_i p_i - w_3 C_i(p_{pa(i)} - p_i) - w_4 C_i(1 - p_{pa(i)})]).$$

$$\therefore LE(\hat{\mathbf{y}}, \mathbf{x}) = \sum_{i>0}^{N-1} w_2 C_i p_i - LE_\delta(\hat{\mathbf{y}}, \mathbf{x}).$$

So, the solution to minimize $LE(\hat{\mathbf{y}}, \mathbf{x})$ is equivalent to the one to maximize $LE_\delta(\hat{\mathbf{y}}, \mathbf{x})$. □

The problem (3.12) is still challenging since it contains two free variables y_i and $y_{pa(i)}$ under the hierarchy constraint.

To simplify the problem further, we introduce notations $\sigma_1(i)$ and $\sigma_2(i)$ as follows:

$$\sigma_1(i) = \sum_{j \in child(i)} (w_2 - w_1) C_j p_j. \tag{3.13}$$

Particularly, if $ch(i) = \varnothing$, $\sigma_1(i) = 0$, and

$$\sigma_2(i) = w_1 C_i p_i - w_3 C_i (p_{pa(i)} - p_i) - w_4 C_i (1 - p_{pa(i)}). \tag{3.14}$$

Let $\sigma(i)$ be a function of node i defined as

$$\sigma(i) = \begin{cases} \sigma_1(i), & i = 0; \\ \sigma_1(i) + \sigma_2(i), & i > 0. \end{cases} \tag{3.15}$$

The equation (3.15) implies:

Proposition 3.3.3

$$LE_\delta(\hat{\boldsymbol{y}}, \boldsymbol{x}) = \sum_i \hat{y}_i \sigma(i). \tag{3.16}$$

Proof. Let $T = \sum_i y_i \sigma(i)$. Our goal is to prove $LE_\delta(\hat{\mathbf{y}}, \mathbf{x}) = T$. According to equation(3.15), we have:

$$T = \sum_i \sigma_1(i) + \sum_{i>0} \sigma_2(i).$$

Let $T_1 = \sum_i \sigma_1(i)$ and $T_2 = \sum_{i>0} \sigma_2(i)$. Then,

(a)

$$T_1 = \sum_i y_i \sum_{j \in child(i)} (w_2 - w_1) C_j p_j$$

$$\Leftrightarrow T_1 = \sum_i \sum_{j \in child(i)} y_{pa(j)} (w_2 - w_1) C_j p_j.$$

Since j is a child of node i, $j > 0$,

$$\therefore T_1 = \sum_{j>0} y_{pa(j)}(w_2 - w_1)C_j p_j.$$

$$(b) T_2 = \sum_{i>0} y_i(w_1 C_i p_i - w_3 C_i(p_{pa(i)} - p_i) - w_4 C_i(1 - p_{pa(i)})).$$

(c) Combining both T_1 and T_2, we prove $T = LE_\delta(\hat{\mathbf{y}}, \mathbf{x})$. $\square$

Based on the equation (3.16), the solution to the problem (3.12) is equivalent to the one of problem (3.17).

$$\begin{aligned} \hat{\mathbf{y}}^* &= \arg\max_{\hat{y} \in \{0,1\}^N} \sum_i y_i \sigma(i) \\ &\text{s.t. } \hat{\mathbf{y}} \text{ satisfies the hierarchy constraint.} \end{aligned} \tag{3.17}$$

The solution of the problem (3.17) by a greedy algorithm is described in the next section.

3.4 Algorithms and Solutions

As discussed in previous sections, there are three key steps to obtain the hierarchical multi-labeling of the instances having minimal CH-loss.

1. Estimate the probability p_i for each node i based on the training data.
2. Incorporate the domain knowledge $\mathbf{k}$ and adjust the probability p_i for each node i accordingly.
3. Use p_is to compute the $\sigma(i)$ defined by the equation (3.15).
4. Obtain the optimal predictor $\hat{\mathbf{y}}^*$ as a solution of the problem (3.17).

3.4.1 Estimating Probability p_i

According to the equation (3.10), p_i can be computed by estimating the probability $P(y_i = 1|y_{pa(i)} = 1, \mathbf{x})$. For each node i with positively labeled the parent node, a binary classifier is built based on existing methods, such as logistic regression or support vector machine. Given an instance $\mathbf{x}$, we apply thresholding described in [P$^+$99] to convert the real output of the classifier to estimate $P(y_i = 1|y_{pa(i)} = 1, \mathbf{x})$.

The task of building classifiers for all the nodes is a significant load. Since the building process of the classifier on each node only relies on the related training data and all the classifiers are mutually independent, we parallelize the task to improve the performance [ZJZ$^+$13b].

Then, the values of p_is are computed by applying formula 3.9 while traversing the nodes in the hierarchy. Figure 3.6(b) illustrates the marginal probabilities by consider the hierarchical label tree in Figure 3.6(a). The time complexity of p_i computation is $O(N)$, where N is the number of nodes in the hierarchy.

3.4.2 Incorporating Prior Knowledge k

In a probabilistic graphical model, each node represents a random variable and the link between two nodes expresses the probabilistic relationship between them [B$^+$06]. The graph captures the way in which the joint distribution over all of the random variables can be decomposed into a product of factors each depending only on a subset of the variables. Accordingly, it is straightforward to interpret the hierarchical tree H as a probabilistic graphical model where each label node y_i corresponds to a random variable taking value either 0 or 1 and each link from parent to child represents the hierarchical constraint 3.2.1. The label inference on a tree-structured graph can be efficiently addressed by the *sum-product* algorithm [B$^+$06]. In this section, *Kilo* (**K**nowledge guided h**i**erarchical mutli-**l**abel classificati**o**n), a *sum-product* based algorithm, is proposed to adjust the marginal probability p_i for knowledge incorporation.

Assuming y_i and y_j to be two label nodes in H where a link occurs between them, $\mu_{y_i \to y_j}(\hat{y}_j)$ denotes the message passed from node y_i to node y_j, when the random variable y_j takes the value $\hat{y}_j$.

Definition 3.4.1 (Accumulated Message) *Given a node y_j and a node set U where any node $y_t \in U$ is a neighbour of y_j, $Prod_{U \to y_j}(\hat{y}_j)$ is referred as the message accumulated on node y_j from U when $y_j = \hat{y}_j$. It is defined as follows:*

$$Prod_{U \to y_j}(\hat{y}_j) = \prod_{y_t \in U} \mu_{y_t \to y_j}(\hat{y}_j). \tag{3.18}$$

Especially, when $U = \emptyset$, $Prod_{U \to y_j}(\hat{y}_j) = 1$.

Definition 3.4.2 (Passed Message) *Given a node y_j and its neighbour y_i, let $U_{i/j}$ denote a set containing all the neighbours of y_i except y_j. The message passed from y_i to y_j when $y_j = \hat{y}_j$ is defined as follows:*

$$\mu_{y_i \to y_j}(\hat{y}_j) = \begin{cases} \sum_{y_i} p(y_i | y_j = \hat{y}_j) Prod_{U_{i/j} \to y_i}(y_i), & y_i \text{ is child of } y_j; \\ \sum_{y_i} p(y_j = \hat{y}_j | y_i) Prod_{U_{i/j} \to y_i}(y_i), & y_j \text{ is child of } y_i. \end{cases} \tag{3.19}$$

Proposition 3.4.3 *Let U be the node set containing all the neighbours of y_j, then the marginal probability*

$$p(y_j = \hat{y}_j) = Prod_{U \to y_j}(\hat{y}_j). \tag{3.20}$$

This proposition can be simply verified by an example in Figure 3.6. According to the D-separation property of probabilistic graphical model [B+06], the joint probability of the label vector in the example is given by

$$p(y_0, y_1, y_2, y_3, y_4) = p(y_0)p(y_1|y_0)p(y_2|y_0)p(y_3|y_1)p(y_4|y_1).$$

Therefore, the marginal probability $p(y_1)$ can be computed as follows:

$$p(y_1) = \sum_{y_0} \sum_{y_2} \sum_{y_3} \sum_{y_4} p(y_0)p(y_1|y_0)p(y_2|y_0)p(y_3|y_1)p(y_4|y_1). \tag{3.21}$$

According to the definition of accumulated message, $Prod_{U \to y_1}(y_1)$ can be computed in the following.

$$\begin{aligned} Prod_{U \to y_1}(y_1) &= \mu_{y_0 \to y_1}(y_1)\mu_{y_3 \to y_1}(y_1)\mu_{y_4 \to y_1}(y_1) \\ &= \sum_{y_0} p(y_1|y_0)\mu_{y_2 \to y_0}(y_0) \cdot \sum_{y_3} p(y_3|y_1) \cdot \sum_{y_4} p(y_4|y_1) \\ &= \sum_{y_0} (p(y_1|y_0) \cdot \sum_{y_2} p(y_2|y_0)) \cdot \sum_{y_3} p(y_3|y_1) \cdot \sum_{y_4} p(y_4|y_1) \\ &= \sum_{y_0} \sum_{y_2} \sum_{y_3} \sum_{y_4} p(y_1|y_0)p(y_2|y_0)p(y_3|y_1)p(y_4|y_1). \end{aligned} \tag{3.22}$$

Since $p(y_0 = 1) = 1.0$, $p(y_0 = 0) = 0.0$, $p(y_1|y_0 = 0) = 0.0$ and $p(y_2|y_0 = 0) = 0.0$, we get $p(y_1) = Prod_{U \to y_1}(y_1)$ based on Equation (3.21) and (3.22).

Algorithm 1 Kilo

```
1: procedure $Kilo(\mathbf{H}, \mathbf{k})$
   ▷$\mathbf{H}$ is the label hierarchy tree, with $P(y_i = 1|y_{pa(i)} = 1, \mathbf{x})$ on its corresponding link.
   ▷$\mathbf{k}$ is the knowledge vector.
2:     Initialize a 3-dimensional array $T$ with size $N \times N \times 2$, where $N$ is the number of labels. $T_{ijk}$ corresponds to the passed message $\mu_{y_i \to y_j}(y_j = k)$, where k is either 0 or 1.
   ▷compute $\mu_{y_i \to y_j}(y_j = k)$ from bottom to top.
3:     Starting from leaf nodes along the links between their parents, compute $\mu_{y_i \to y_j}(y_j = k)$ and fill it in the $T_{ijk}$.
   ▷compute $\mu_{y_i \to y_j}(y_j = k)$ from top to bottom.
4:     Starting from root node along the links between their children, compute $\mu_{y_i \to y_j}(y_j = k)$ and fill it in the $T_{ijk}$.
5:     Compute the marginal probability for each label according to Equation (3.20).
6:     Normalize all the marginal probability with the marginal probability of root label.
7:     return a vector containing all the marginal probabilities for all the labels.
8: end procedure
```

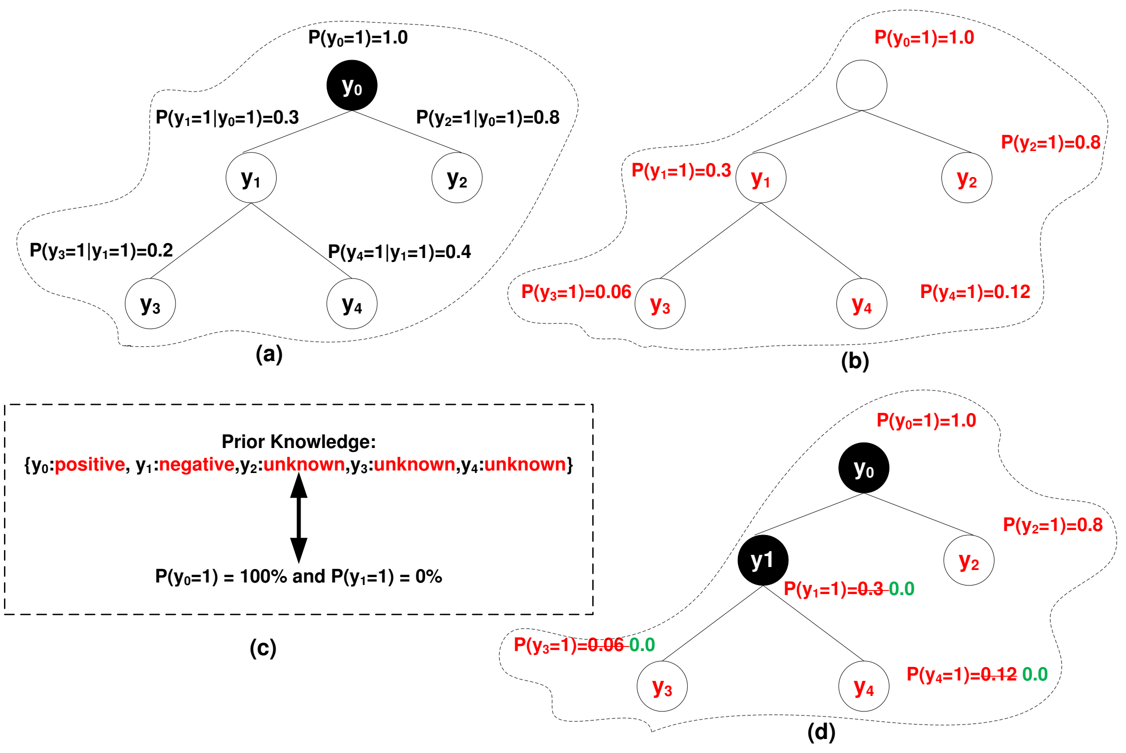


Figure 3.6: This figure illustrates the marginal probability by incorporating the domain knowledge.The label nodes in black circle are observed. (b) can be inferred by marginalization from (a), while (d) can be inferred by considering both (a) and prior knowledge in (c). The prior knowledge in (c) can be represented with a likelihood vector about being positive.

So far, we have considered the label inference without any knowledge. According to Definition 3.2.2, the knowledge is represented as a vector **k** (shown in Figure 3.6 (c)). If $k_i = 0$ or $k_i = 1$, it indicates that the i^{th} label is observed as *negative* or *positive*. While $k_i = -1$, it means the i^{th} label is hidden as *unknown*. In order to incorporate the knowledge vector **k**, an indicator function is defined as follows:

$$I(y_i, k_i) = \begin{cases} 1, if\ k_i = -1 \vee y_i = k_i; \\ 0, otherwise. \end{cases} \tag{3.23}$$

The knowledge incorporation can be implemented by multiplying the joint probability $p(\mathbf{y})$ with $\prod_i I(y_i, k_i)$. The product corresponds to the joint probability with some observed labels, which is an un-normalized version of posterior probability given the observable labels. Hence, we can get $p(\mathbf{y}|\mathbf{k})$ by normalization. Figure 3.6(d) shows the final marginal probabilities inferred by incorporating knowledge vector in Figure 3.6(c).

According to the above formulation, *Kilo* algorithm is given in Algorithm 1.

The *Kilo* algorithm takes the hierarchical tree and the knowledge vector as the inputs and return the marginal probability vector after knowledge incorporation. It traverses the tree along the links twice in both bottom-to-top and top-to-bottom directions. It needs to normalize the marginal probability for each node. Therefore, the overall time complexity is $O(N + E)$, where N and E are the number of nodes and number of links respectively.

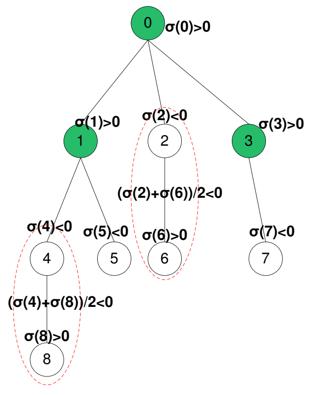


Figure 3.7: Figure illustrates hierarchy with 9 nodes and steps of Algorithm 2. Nodes labeled positive are green. A dotted ellipse marks a super node composed of the nodes in it.

3.4.3 Computing Variable $\sigma(i)$

With p_i available, σ can be computed based on equation (3.15) by recursively traversing each node of the hierarchy. Since each node in hierarchy needs to be accessed twice, one for computing σ_1 and the other for computing σ_2. Therefore, time complexity of $\sigma(i)$ evaluation is also $O(N)$.

3.4.4 Obtaining Label $\hat{\mathbf{y}}^*$

Algorithm 2 GLabel

```
 1: procedure GLabel(H)
    ▷H is the label hierarchy, with $\sigma$ available
 2:     define L as a set, and initialize $L = \{0\}$
 3:     define U as a set, and initialize $U = \mathbf{H}\backslash\{0\}$
 4:     while TRUE do
 5:         if all the nodes in H are labeled then
 6:             return L
 7:         end if
 8:         find the node i with maximum $\sigma(i)$
 9:         if $\sigma(i) < 0$ then
10:             return L
11:         end if
12:         if all the parents of i are labeled then
13:             put i into L, and remove it from U
14:         else
15:             merge i with its parent as a super node $i^*$
16:             $\sigma(i^*)$ =average $\sigma$ values of the two nodes
17:             put the $i^*$ into U
18:         end if
19:     end while
20: end procedure
```

The value $\hat{\mathbf{y}}^*$ is obtained by solving the maximization problem (3.17). [BK11] proposed the greedy algorithm CSSA, based on the work in [BJ94] that allows for solving the problem (3.17) efficiently. However, CSSA only works under an assumption that the number of labels to be associated with a predicted instance is known. That assumption rarely holds in practice. In [WJ12], the HIROM algorithm is proposed to avoid the deficiency of CSSA by giving the maximum number of labels related to a predicting instance. During the process of finding maximum number of labels, HIROM gets the optimal $\hat{\mathbf{y}}^*$ by comparing all possible $\hat{\mathbf{y}}$s with different numbers of labels related to a predicting instance.

We suggest a novel greedy labeling algorithm GLabel(**Algorithm 2**) to solve the problem (3.17). This algorithm finds the optimal $\hat{\mathbf{y}}^*$ without knowing the maximum number of labels for the predicting instance. It labels the node (or super node) i with maximum $\sigma(i)$ to be positive by searching in the hierarchy. If the parent node of i is negative, then i and its parent are merged into a super node whose σ value is the average σ value of all the nodes contained in the super node (Figure 3.7). The labeling procedure stops when the maximum σ value is negative or all nodes are labeled positive.

Since the labeling procedure for each node may involve a merging procedure, the time complexity is no worse than $O(Nlog(N))$, the same as HIROM. However, as shown in the experimentation section below, GLabel performs more efficiently than HIROM while not requiring knowledge of the maximum number of labels.

3.5 Experiments

3.5.1 Setup

We perform the experiment over the ticket data set generated by monitoring of the IT environments of a large IT service provider. The number of tickets in the experiment amounts to about 23,000 in total. The experiment is executed 10 times and we use the mean value of the corresponding metric to evaluate the performance of our proposed method. At each time, 3000 tickets are sampled randomly from the whole ticket data set to build the testing data set, while the rest of the tickets are used to build the training data set. The class labels come from the predefined catalog information for problems occurring during maintenance procedures. The whole catalog information of problems is organized in a hierarchy, where each node refers to a class label. The catalog contains 98 class labels; hence there are 98 nodes in the hierarchy. In addition, the tickets are associated with 3 labels on average and the height of the hierarchy is 3 as well.

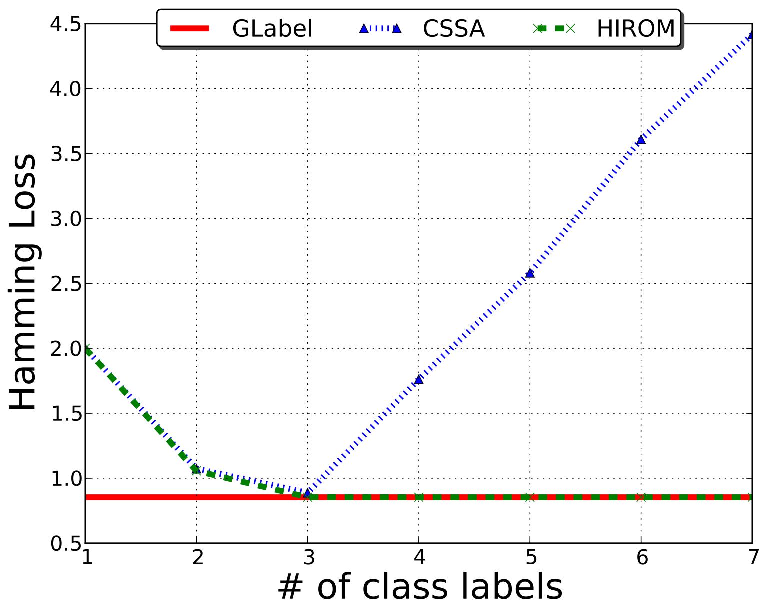


Figure 3.8: The lowest Hamming loss: CSSA gets 0.8876 at # 3; HIROM gets 0.8534 at # 4; GLabel gets 0.8534.

The features for each ticket are built from the short text message describing the symptoms of the problem. First, Natural language processing techniques are applied to remove the stop words and build Part-Of-Speech tags for the words in the text. The nouns, adjectives and verbs in the text are extracted for each ticket. Second, we compute the TF-IDF [MRS08] scores of all words extracted from the text of tickets. And the words with the top 900 TF-IDF score are kept as the features for the tickets. Third, the feature vector of each ticket has 900 components, where value of each feature is the frequency of the feature word occurring in the text of the ticket.

Based on the features and labels of the tickets, we build a binary classifier for each node in the hierarchy with the SVM algorithm by using library libSVM [url15]. The training data for each node i are the tickets with a positive parent label. To speed up evaluation of the 98 SVM classifiers, we parallelize the process of training classifiers, using the fact that all the classifiers are independent.

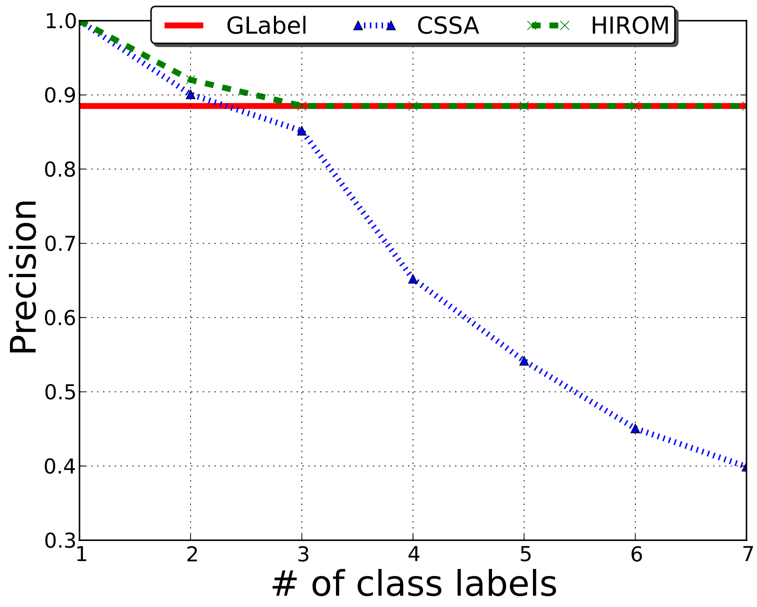


Figure 3.9: Varying precision during minimizing the Hamming loss

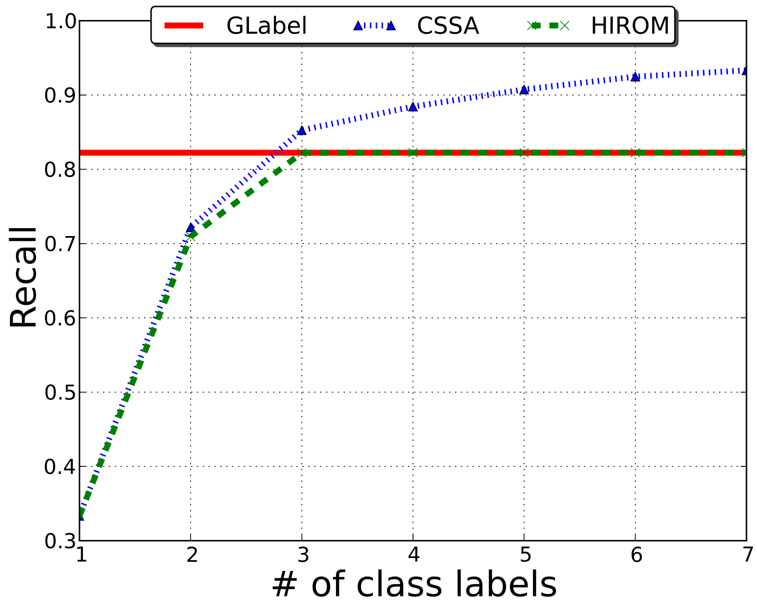


Figure 3.10: Varying recall during minimizing the Hamming loss

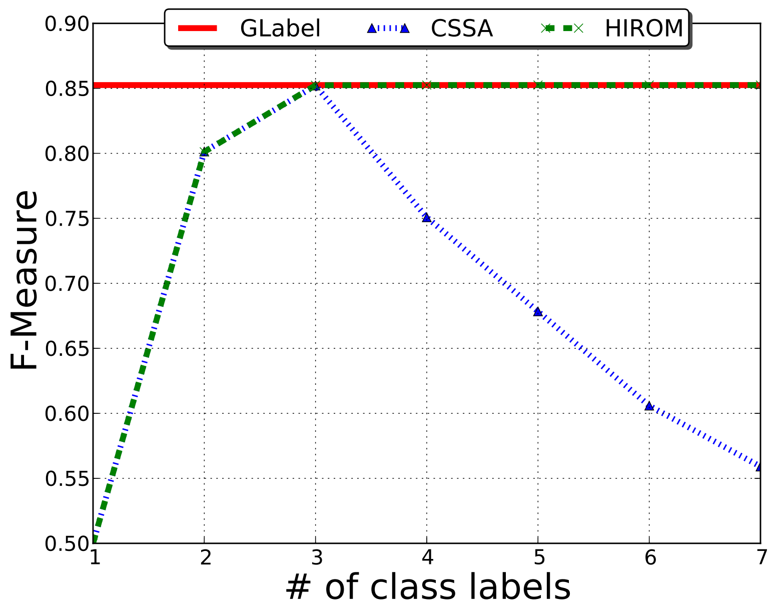


Figure 3.11: Varying FMeasure score during minimizing the Hamming loss

The experiments are mainly conducted by comparing the proposed GLabel algorithm with state-of-the-art algorithms such as CSSA and HIROM. Note, that in the end, we also show benefits of hierarchical classification in comparison to the "Flat" classification.

3.5.2 Hamming Loss

The GLabel algorithm can obtain optimal $\hat{\mathbf{y}}^*$ with minimum Hamming loss by setting the parameters for Hamming loss, since Hamming loss is a special case of CH-loss. Given $w_1 = w_2 = w_3 = w_4 = 1$ for GLabel, $\alpha = \beta = 1$ for HIROM and $C_i = 1$ for both of them, empirical results are displayed in Figure 3.8 - 3.11. Sub-figure (a) shows that the GLabel algorithm can automatically find the optimal $\hat{\mathbf{y}}^*$ with minimum Hamming loss, while both CSSA and HIROM require the number of class labels and the maximum number of class labels, respectively, to get the optimal $\hat{\mathbf{y}}^*$. With the increasing number of class labels, HIROM gets lower Hamming loss until it reaches the optimal $\hat{\mathbf{y}}^*$ with minimum Hamming loss by choosing large enough number of class labels. However, CSSA may get larger Hamming loss as the number of class labels increases. The sub-figure (b)-(d) show as in comparison to Hamming loss, Glabel algorithm shows good performance in Precision, Recall and FMeasure score.

3.5.3 HMC-Loss

The HMC-loss considers loss with respect to the node position in the hierarchy. Following [WJ12], we define the C_i as follows.

$$C_i = \begin{cases} 1, & i = 0; \\ \frac{C_{pa(i)}}{\text{\# of i's siblings}}, & i > 0. \end{cases} \tag{3.24}$$

To simplify, we set $w_1 = w_2 = w_3 = w_4 = 1$ for GLabel and $\alpha = \beta = 1$ for HIROM as well. The Figure 3.12 shows that GLabel algorithm obtains the same lowest HMC-loss as HIROM algorithm does, with the HIROM tuned for minimizing the HMC-loss.

3.5.4 H-Loss

In order to get the minimum H-loss, we set $w_1 = w_3 = 1$,$w_2 = w_4 = 0$ for GLabel and $\alpha = \beta = 1$ for HIROM, $C_i = 1$ for all the three algorithms. The Figure 3.13 shows that GLabel gets the lowest H-loss in comparison to HIROM and CSSA minimums. HIROM and CSSA algorithms cannot get the optimal $\hat{\mathbf{y}}^*$ with minimal H-loss.

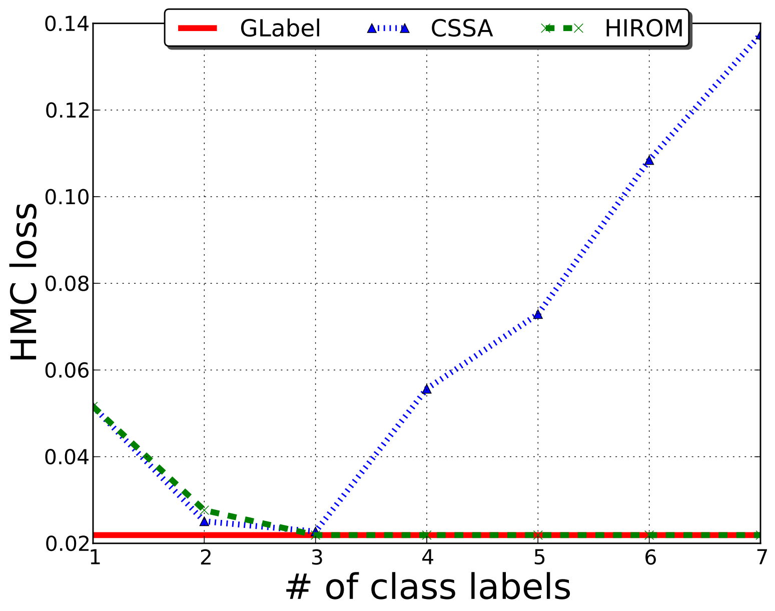


Figure 3.12: The lowest HMC-Loss: CSSA gets 0.0227 at # 3; HIROM gets 0.0219 at # 4; GLabel gets 0.0219.

3.5.5 Misclassifications Occur in Both Parent and Child Labels

The worse error from the loss point of view is the misclassification of both parent and child nodes. We call such misclassification a parent-child error. In terms of CH-loss, GLabel can minimize the number of such cases by setting $w_1 = w_3 = 1$, $w_2 = w_4 = 10$ with more penalties in parent-child errors. To compare, we set in CSSA and HIROM $\alpha = \beta = 1$, and C_i according to the equation (3.24). In Figure 3.14, GLabel reaches the minimum average number of parent-child errors, while CSSA and HIROM algorithms do not minimize the parent-child errors since they do not consider the contextual misclassification information in their loss function.

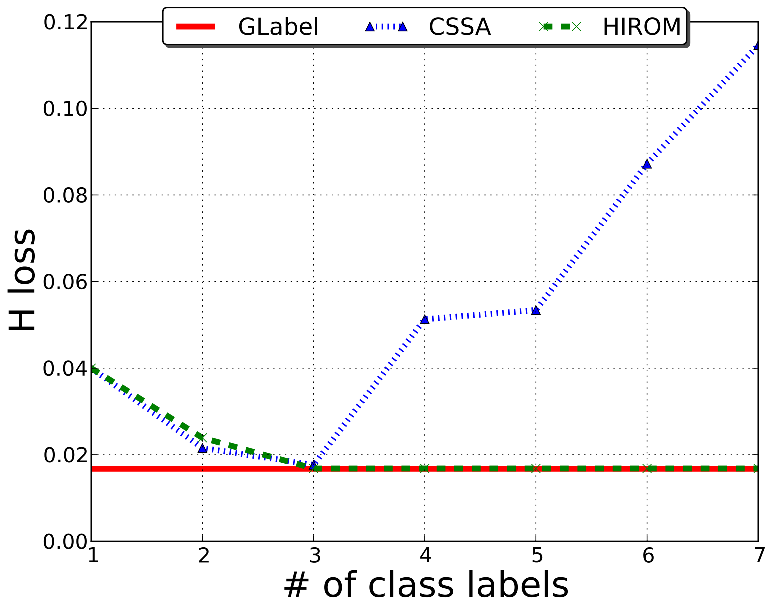


Figure 3.13: The lowest H-Loss: CSSA gets 0.0176 at # 3; HIROM gets 0.0168 at # 3; GLabel gets 0.0167.

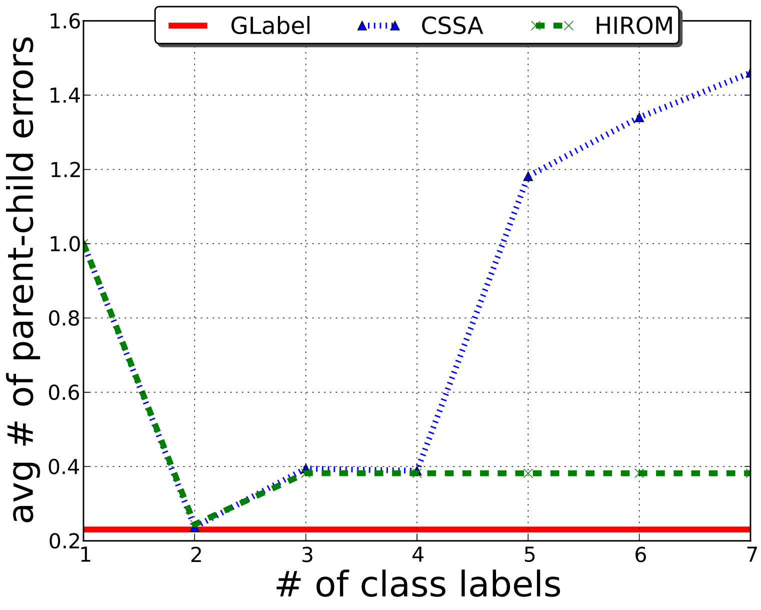


Figure 3.14: The lowest AVG. parent-child error: CSSA gets 0.237 at # 2; HIROM gets 0.2440 at #2; Glabel gets 0.2304.

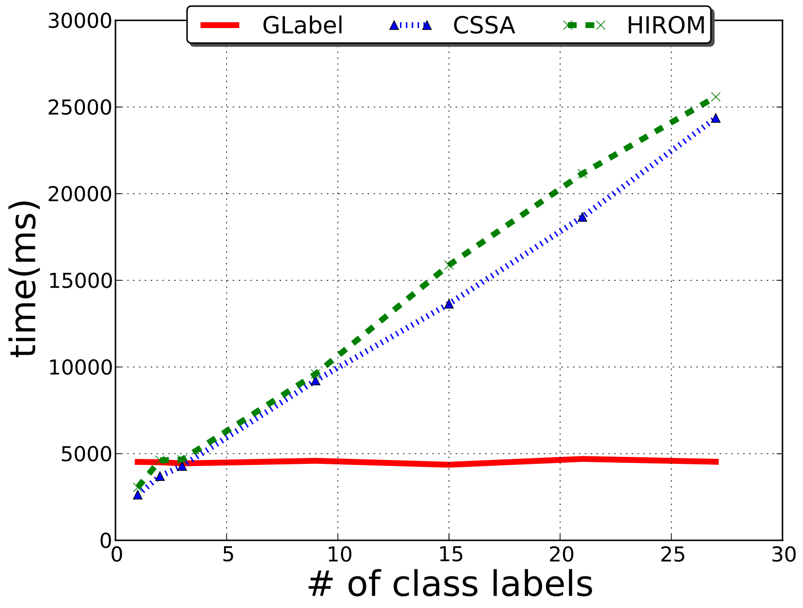


Figure 3.15: Time complexity with respect to the number of classes related to each predicting ticket.

3.5.6 Time Complexity

In order to evaluate the time complexity, we fix the same parameters but increase the number of classes labels, see Figure 3.15. We run three algorithms for 40 rounds and get the average time consumed. Figure 3.15 shows that run time of GLabel is independent from the number of labels, while other algorithms require more time as the number of labels increases. Hence, the GLabel algorithm is more efficient than other two algorithms, especially in the cases with large number of class labels.

3.5.7 Comparison Study With Flat Classifier

To set up a "Flat" classification, a classifier is built for each label independently without considering the hierarchy constraint. The SVM algorithm is one of the best performing algorithms used to classify the ticket data with each binary class label. In order to decrease the parent-child error, we set $w_1 = w_3 = 1$, $w_2 = w_4 = 10$, and

C_i as the equation (3.24). In addition, we define the hierarchy error as the average number of violated hierarchy constraints.

Metric	**SVM**	**GLabel**
CH-loss	4.2601	2.6889
Parent-child error	0.3788	0.1729
Hierarchy error	0.0102	0.0

Table 3.2: Comparison with "Flat" classification

The table 3.2 shows that GLabel algorithm has better performance in terms of CH-loss and parent-child error. Furthermore, the "Flat" SVM classifiction suffers on average 0.0102 hierarchy errors with each ticket, while GLabel complies with the hierarchy constraint and does not have hierarchy errors.

3.5.8 Experiment With Prior Knowledge

Setup for Knowledge Incorporation

In order to demonstrate the effectiveness of the proposed Kilo algorithm for knowledge guided hierarchical multi-label classification, we perform the experiments over the same real ticket data set described in section 3.5.1. The only difference lies in the additional knowledge construction. Each ticket instance in the test data set are associated with a knowledge vector $\mathbf{k}$, where the i^{th} component are exposed with its ground true value randomly according to a predefined prior knowledge ratio $\gamma \in [0, 1]$. The prior knowledge ratio γ is the probability that true labels are observed as either *postive* or *negative*, while $1 - \gamma$ denotes the probability that the label can not be observed before classification. Figure 3.16 - 3.23 show the performances in terms of different metrics in comparing GLabel algorithm without prior knowledge and the one with prior knowledge incorporated by Kilo algorithm.

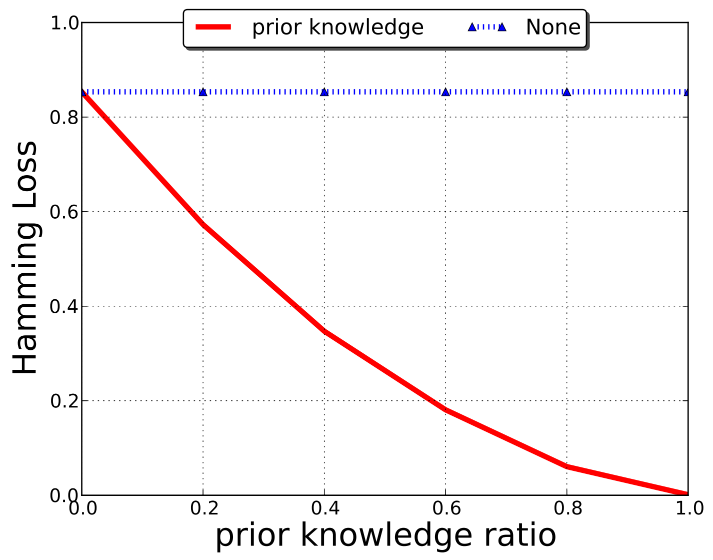


Figure 3.16: Experiment with prior knowledge is conducted over the ticket data in terms of Hamming Loss. Hamming Loss decreases as more prior knowledge provided. "prior knowledge" denotes the hierarchical multi-label classification with knowledge incorporation. "None" denotes the hierarchical multi-label classification without knowledge incorporation.

Hamming Loss with Changing Prior Knowledge Ratio

Similar to previous configuration, we obtain Hamming loss by setting the CH-loss parameters $w_1 = w_2 = w_3 = w_4 = 1, C_i = 1$. As illustrated in Figure 3.16, Hamming loss drops as the prior knowledge ratio increases and reaches to zero as prior knowledge ratio achieves one. Figure 3.17 - 3.19 illustrate the performance in terms of Precision, Recall and F-Measure, and it shows that the knowledge guided algorithm gets better performance as prior knowledge ratio increases.

HMC-Loss

The HMC-Loss is obtained by the same settings as provided in section 3.5.3. As expected, HMC-Loss decreases as more prior knowledge is provided, shown in Figure 3.20.

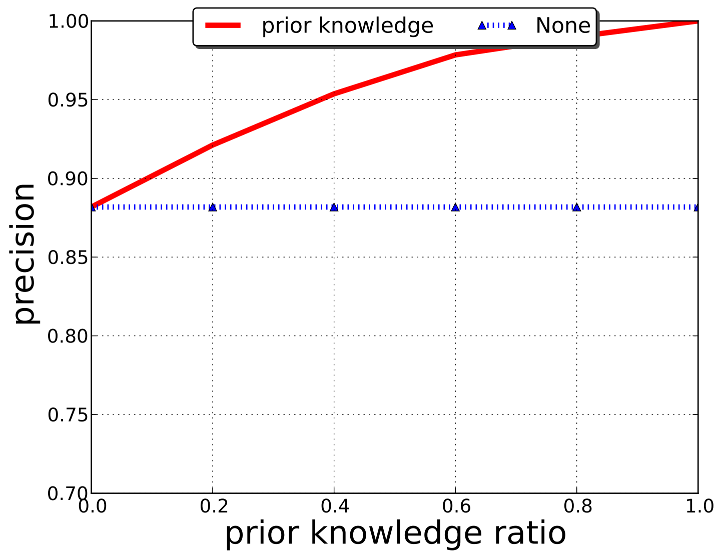


Figure 3.17: Experiment with prior knowledge is conducted over the ticket data in terms of precision. Precision increases as more prior knowledge provided. “prior knowledge” denotes the hierarchical multi-label classification with knowledge incorporation. “None” denotes the hierarchical multi-label classification without knowledge incorporation.

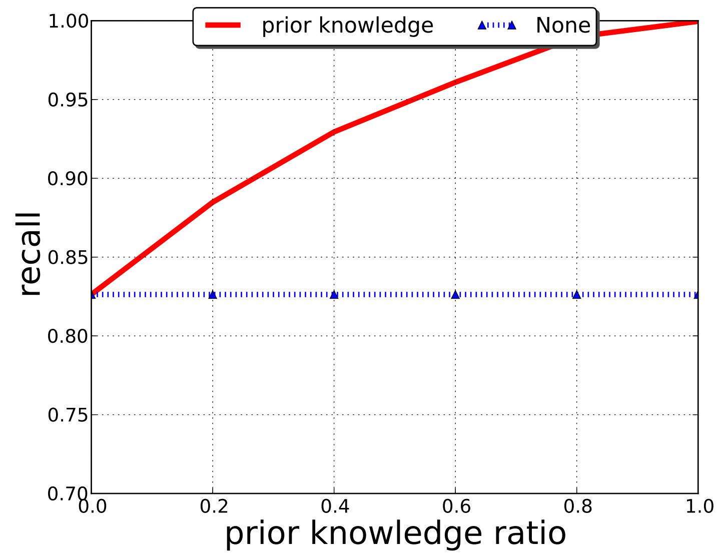


Figure 3.18: Experiment with prior knowledge is conducted over the ticket data in terms of recall. Recall increases as more prior knowledge provided. “prior knowledge” denotes the hierarchical multi-label classification with knowledge incorporation. “None” denotes the hierarchical multi-label classification without knowledge incorporation.

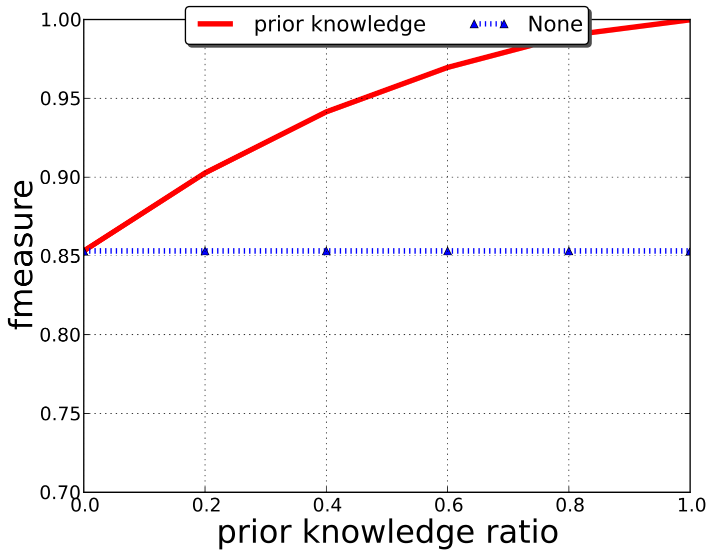


Figure 3.19: Experiment with prior knowledge is conducted over the ticket data in terms of F-Measure. F-Measure increases as more prior knowledge provided. “prior knowledge” denotes the hierarchical multi-label classification with knowledge incorporation. “None” denotes the hierarchical multi-label classification without knowledge incorporation.

H-Loss

The H-Loss is obtained by the same parameter settings as provided in section 3.5.4. By increasing the prior knowledge ratio, H-Loss caused by knowledge guided algorithm decreases, shown in Figure 3.21.

Parent-Child Error

The Parent-Child Error is obtained by the same parameter settings as presented in Section 3.5.5. By increasing the prior knowledge ratio, Parent-Child errors caused by knowledge guided algorithm decreases, shown in Figure 3.22.

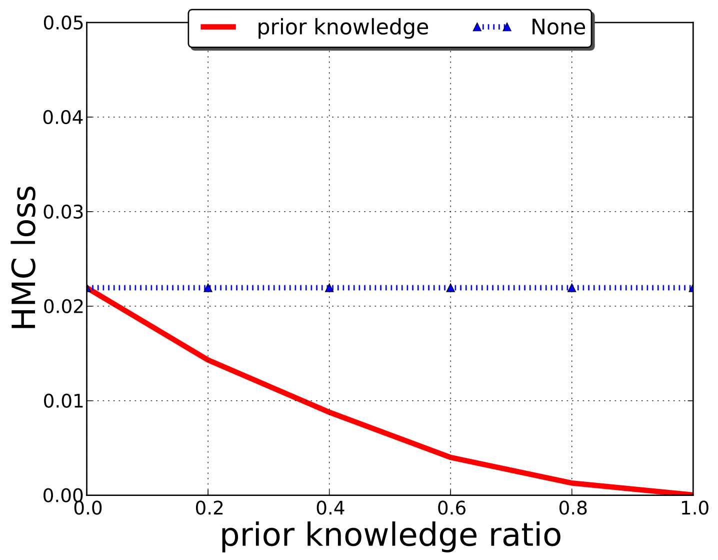


Figure 3.20: Varying HMC-Loss with changing prior knowledge ratio.Experiment with prior knowledge is conducted over the ticket data in terms of HMC-Loss. HMC-Loss decreases as more prior knowledge is provided. “prior knowledge” denotes the hierarchical multi-label classification with knowledge incorporation. “None” denotes the hierarchical multi-label classification without knowledge incorporation.

CH-Loss

The CH-Loss is obtained by the same parameter settings as given in Section 3.5.5. By increasing the prior knowledge ratio, CH-Loss caused by knowledge guided algorithm decreases, shown in Figure 3.23.

3.6 Summary

In this chapter, we employ hierarchical multi-label classification over ticket data to facilitate the problem diagnosis, determination and an automated action, such as auto-resolution or auto-check for enriching or resolving the ticket in the complex IT environments. CH-loss is proposed by considering the contextual misclassification information to support different scenarios in IT environments. In terms of CH-loss, an optimal prediction rule is developed based on Bayes decision theory. This chapter

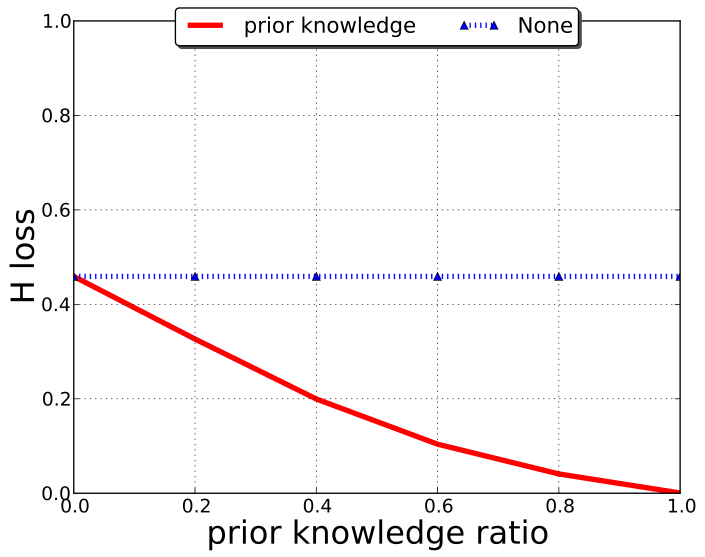


Figure 3.21: Experiment with prior knowledge is conducted over the ticket data in terms of H-Loss. H-Loss decreases as more prior knowledge is provided. “prior knowledge” denotes the hierarchical multi-label classification with knowledge incorporation. “None” denotes the hierarchical multi-label classification without knowledge incorporation.

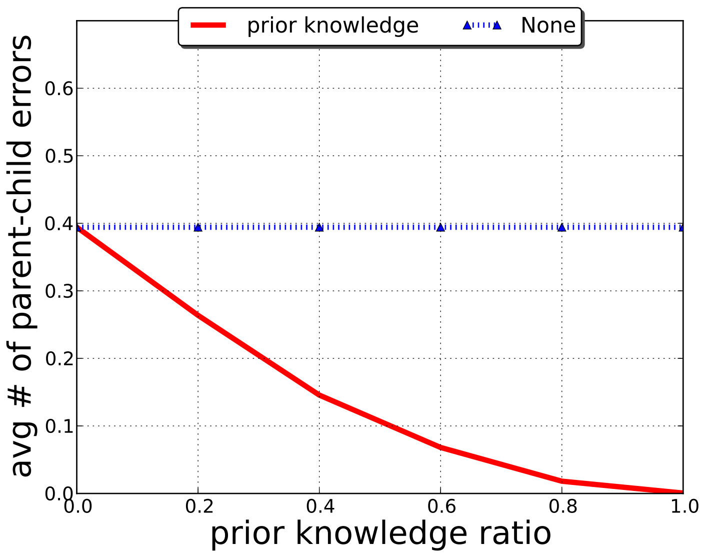


Figure 3.22: Experiment with prior knowledge is conducted over the ticket data in terms of Parent-Child Error. Parent-Child Error decreases as more prior knowledge is provided. “prior knowledge” denotes the hierarchical multi-label classification with knowledge incorporation. “None” denotes the hierarchical multi-label classification without knowledge incorporation.

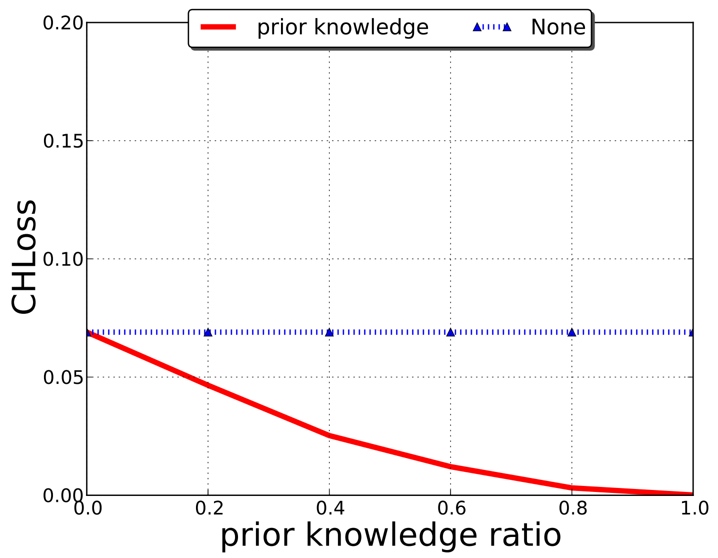


Figure 3.23: Experiment with prior knowledge is conducted over the ticket data in terms of CH Loss. CH Loss decreases as more prior knowledge is provided. "prior knowledge" denotes the hierarchical multi-label classification with knowledge incorporation. "None" denotes the hierarchical multi-label classification without knowledge incorporation.

comes up with a greedy algorithm GLabel by extending the HIROM algorithm to label the predicting ticket without knowing the number or the maximum number of class labels related to the ticket. Additionally, taking the real scenario in practice into account, KILO algorithm is proposed to utilize the knowledge from the domain expert during routine IT maintenance procedure to effectively improve the IT problem category determination.

CHAPTER 4

TEMPORAL PATTERN MINING FROM SYSTEM EVENTS

The importance of mining time lags of hidden temporal dependencies from sequential data is highlighted in many domains including system management, stock market analysis, climate monitoring, and more. Mining time lags of temporal dependencies provides useful insights into the understanding of sequential data and predicting its evolving trend. Traditional methods mainly utilize the predefined time window to analyze the sequential items, or employ statistical techniques to identify the temporal dependencies from a sequential data. However, it is a challenging task for existing methods to find the time lag of temporal dependencies in the real world, where time lags are fluctuating, noisy, and interleaved with each other. In order to identify temporal dependencies with time lags in this setting, this chapter comes up with an integrated framework from both system and algorithm perspectives. Specifically, a novel parametric model is introduced to model the noisy time lags for temporal dependencies discovery between events. Based on the parametric model, an efficient expectation maximization approach is proposed for time lag discovery with maximum likelihood. Furthermore, this chapter also contributes an approximation method for learning time lag to improve the scalability in terms of the number of events, without incurring significant loss of accuracy.

4.1 Introduction

The operational cost of IT service delivery is believed to continue to decrease while the quality of IT service delivery is expected to increase.

Service Providers seek to employ intelligent solutions that provide deep analytical and automation capabilities for optimizing problem detection, root cause analysis and automated resolution [urlb],[ZLSG14b]. Detection is usually realised by an automated monitoring system. This system provides an effective and reliable means of ensuring that degradation of vital signs is flagged as a problem candidate (monitoring event), and sent to the service delivery teams as an incident ticket. When correlated, monitoring events, discrete in nature, could also provide effective and reliable means for problem determination.

There has been a great deal of effort spent on developing methodologies for event correlation and, subsequently, root cause analysis in IT Service Management. One fruitful line of research has involved the development of techniques for traversing graphs dependencies of application configuration. Although these methods have been successful in understanding the systems' failures, they have had a limited impact due to overhead associated with constructing such graphs and keeping them up-to-date. Another approach has focused on the mining temporal properties of events. The essence of this approach is to rely mostly on temporal data from event management systems rather than external data.

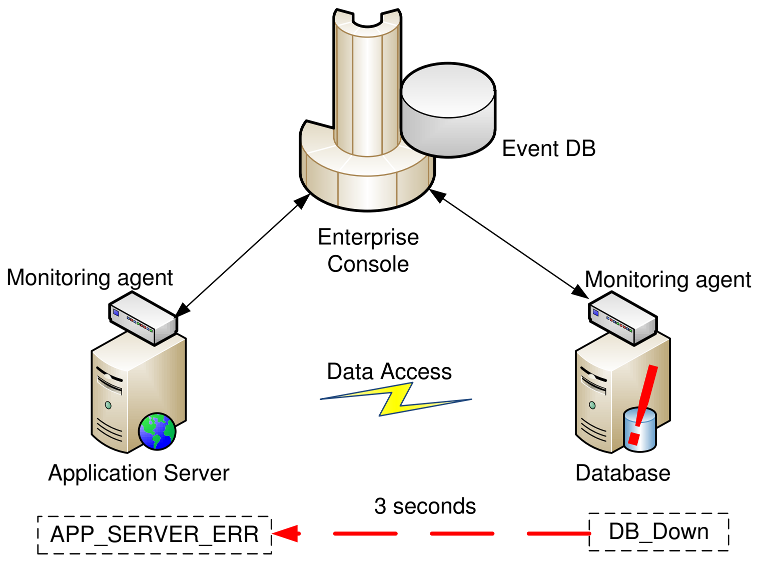


Figure 4.1: A scenario from IT service management demonstrates the event dependency.

Time lag, one of the key features in temporal dependencies, plays an important role in discovering the evolving trends of the upcoming events and predicting future behavior. Take the scenario shown in Figure 4.1 for instance. Applications are deployed on the application server. The availability of services provided by applications relies on access to database. To monitor the statuses of both the application server and database server, two monitoring agents are deployed to the two servers. The enterprise console collects the monitoring events from both monitoring agents, and all the events are stored in the event database. In this scenario, the database goes down, this leads to generation of DB_Down event by its corresponding monitoring agent. After approximately 3 seconds, the monitoring agent on the application server generates a APP_SERVER_ERR event suffering several unsuccessful retries for communication to the database server. The APP_SERVER_ERR event is temporally dependent on the DB_Down event, with a time lag of about 3 seconds.

The temporal dependencies among events are characterized by the time lags. Time lags provide temporal information for building a fault-error-failure chain [ALR01] which is useful for root cause analysis. In addition, events triggered by a single issue can be correlated, given the appropriate time lags. Merging those correlated events in one ticket reduces an administrative effort for problem diagnosis and incident resolution. Thus, the discovery of time lag is a very important task during temporal dependency mining.

The situation in real-world systems becomes complicated due to the limitation of sequential data collecting methods and the inherent complexity of the systems. However, events detected by monitoring systems are typically studied with the assumption that the time lag between correlated events is constant, and fluctuations are limited and can be ignored [TLS12]. Although such an approach is undoubtedly applicable to a wide range of systems, fluctuations can render the deterministic classical picture qualitatively incorrect, especially when correlating events are limited. Taking the randomness of the time lag into account makes the detection of the hidden time lags between interleaved events a challenging task.

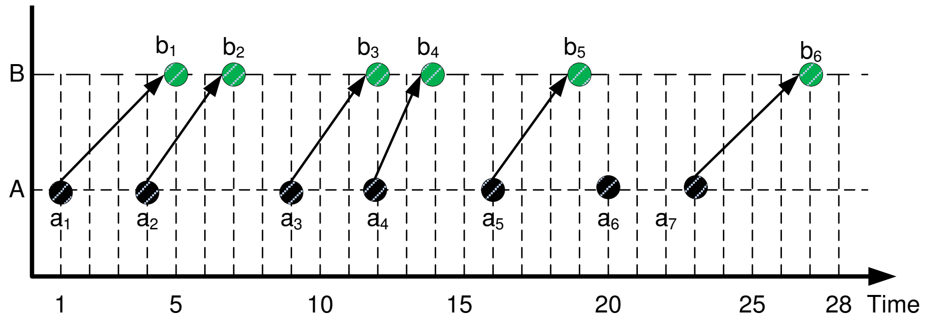


Figure 4.2: The temporal dependencies between A and B are denoted as direct edges, where A and B correspond to DB_Down and APP_SERVER_ERR.

First, the fluctuating interleaved temporal dependencies pose a challenging problem when attempting to discover the correct hidden time lag between two events. For example, two events A and B, corresponding to DB_Down and APP_SERVER_ERR events, respectively, are shown in Figure 4.2. Both A and B occur with multiple instances in the sequential data set. The i^{th} instance of A and the j^{th} instance of B are associated with their time stamps a_i and b_j. Because the true temporal dependencies are interleaved, it is difficult to determine which b_j is implied by a given a_i. The different mapping relationships between a_i and b_j lead to varying time lags. In this example, a_1 can be mapped to any b_j. Therefore, the time lag ranges from $b_1 - a_1$ to $b_6 - a_1$ time units. It is infeasible to find the time lag with exhaustive search from large scale sequential event data.

Second, due to the clocks being out of sync and the limitations of the data collecting method, the time lags presented in the sequential data may oscillate with noise. In Figure 4.2, a_6 does not correspond to any instance of event B for several possible reasons: (1) its corresponding instance of B is missing from the sequential data set, (2) the database returns to normality in such a short time that there is no need to eject an instance of B since the application successfully continues to work after only one try, and (3) even though the correct instance mapping relations between A and B are provided, time lags are still observed with different values due to recording errors brought about by system monitoring.

The above difficulties pose a big challenge for time lag mining. The work in [TLS12] applies a non-parametric method to identify the correlation between events with high time complexity. It's not easy to further improve the efficiency of the non-parametric method. This chapter proposes a parametric model to identify the time lag of the temporal dependencies. With the help of the parametric model, an approximation method is applied to improve the efficiency, without losing much accuracy, for time lag mining.

In summary, our contribution includes:

1. A novel parametric model used tomodel noisy time lags for temporal dependencies' discovery between events.
2. an efficient expectation maximization (abbr., EM) approach for time lag discovery with maximum likelihood.
3. an approximation method for learning time lag to improve the scalability in terms of the number of events without incurring significant loss of accuracy.
4. algorithms and design of a system named TDMS (Temporal Data Mining System) that has the capability of handling a large number of event types and presenting users with a complete solution for temporal lag mining
5. extensive experiments on both synthetic and real data to illustrate the efficiency and effectiveness of the proposed approaches.

The remainder of the chapter is organized as follows: In Section 4.2, the overview of our system is described. In Section 4.3 we formulate the problem for finding time lag with a parametric model. In Section 4.4, an EM-based solution is proposed to solve the formulated problem. Extensive experiments are conducted in Section 4.5. The running system is demonstrated in Section 4.6. Section 4.7 provides a conclusion of our work and discussion of our future work.

4.2 System Architecture

Before diving into the detail of the temporal dependency mining algorithm, we first present the architecture of our integrated system framework.

This chapter mainly focuses on mining temporal dependency with time lag between a pair of event types. The pairwise temporal dependency can be discovered in parallel without interacting with other pairs of events. Accordingly, it is necessary to build a system on distributed computing environment for mining temporal dependencies among large number of event type pairs.

A temporal dependency mining system named TDMS is proposed based on distributed environment. There are two external components shown in the left part of Figure 4.3, working with TDMS. Users as one external component interact with TDMS by issuing HTTP requests for both events query and temporal dependencies discovery. The other external component is composed of a large number of monitored customer servers, where all the alerts and events generated by hosted applications are collected and stored in TDMS.

TDMS has three layers displayed in the right part of Figure 4.3. The bottom layer depicts the storage for event data. To leverage the computing power of the distributed environment for temporal dependency discovery, the whole event data are available to all the computing nodes by placing it on distributed file system such as HDFS (Hadoop Distributed File System).

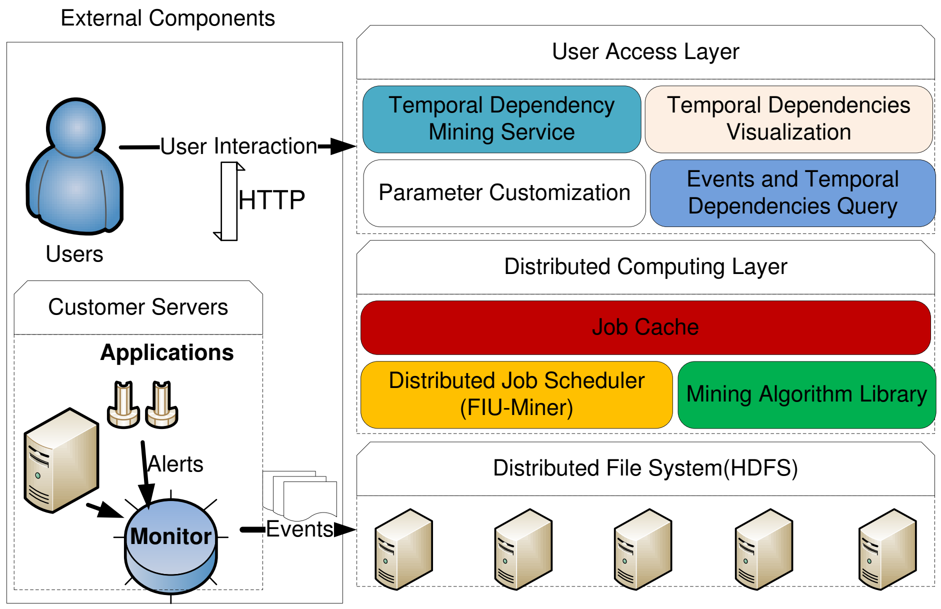

External Components
Users
User Interaction
HTTP
Customer Servers
Applications
Alerts
Monitor
Events
User Access Layer
Temporal Dependency Mining Service
Temporal Dependencies Visualization
Parameter Customization
Events and Temporal Dependencies Query
Distributed Computing Layer
Job Cache
Distributed Job Scheduler (FIU-Miner)
Mining Algorithm Library
Distributed File System(HDFS)

Figure 4.3: The overview of temporal dependency mining system (TDMS).

The middle layers is the distributed computing layer containing three components: Job Cache, Distributed Job Scheduler and Mining Algorithm Library. Job Cache component is used to alleviate the computing burden of the system. The mining results are indexed by its parameters and dataset names, and stored in the cache. As a result, the mining result of the requested job can be retrieved directly from the cache without redundant computation if the same job has been computed before. Distributed Job Scheduler is responsible for scheduling the requested jobs in the distributed environment by considering the resource balance. Distributed Job Scheduler is implemented by FIU-Miner, a Fast, Integrated, and User-Friendly System for Data Mining in Distributed Environment [ZJZ$^+$13a]. The third component in this layer is the algorithm library. The detailed algorithms for temporal dependency mining are given in the subsequent sections.

The top layer is the User Access Layes serving for user interaction with TDMS. Users are able to access all the stored events and query the discovered temporal dependencies in two ways including Temporal Dependency Service and Web-based

Temporal Dependencies Visualization. Moreover, users can customize the parameters of the mining algorithms and specify the event data set to discover the temporal dependencies for the interests of users. The temporal dependencies are demonstrated in Section 4.6.

4.3 Problem Formulation

4.3.1 Problem Description

In temporal pattern mining, the input data is a sequence of events. Given the event space $\mathbf{\Omega}$ of all possible events, an event sequence $\mathbf{S}$ is defined as ordered finite sequence $\mathbf{S} =< e_1, e_2, ..., e_i, ..., e_k >$, where e_i is an instance of an event. We consider temporal events, i.e., each e_i is a tuple $e_i = (E_i, t_i)$ of event $E_i \in \mathbf{\Omega}$ and a time stamp t_i of event occurrence.

Let A and B be two types of events from the event space $\mathbf{\Omega}$. We define $\mathbf{S_A} =< (A, a_1), ..., (A, a_m) >$ to be a subsequence from $\mathbf{S}$, where only the instances of A are kept and a_i is the time stamp of i^{th} event A. Since all the instances happening in the sequence $\mathbf{S_A}$ belong to the same type of event A, $\mathbf{S_A}$ can be simply denoted as a sequence of time stamps, i.e., $\mathbf{S_A} =< a_1, ..., a_m >$. Similarly, $\mathbf{S_B}$ is denoted as $\mathbf{S_B} =< b_1, ..., b_n >$. Discovering the temporal dependency between A and B is equivalent to finding the temporal relation between $\mathbf{S_A}$ and $\mathbf{S_B}$.

Specifically, if the j^{th} instance of event B is associated with the i^{th} instance of event A after a time lag $(\mu + \epsilon)$, it indicates

$$b_j = a_i + \mu + \epsilon, \tag{4.1}$$

where b_j and a_i are the time stamps of two instances of B and A respectively, μ is the true time lag to describe the temporal relationship between A and B, and ϵ is a

Table 4.1: Notations.

Notation	**Description**
$\mathbf{\Omega}$	the event space containing all the event types.
$\mathbf{S}$	an event sequence.
$\mathbf{S_A}, \mathbf{S_B}$	the event sequence with a specific event type.
A , B	the event type.
a_i , b_i	the time stamps of the i^{th} instances of event A, B.
μ	the true time lag between two events.
ϵ	the noise associated with the time lag.
ξ	the probability sum of the components is neglected in the approximation algorithm.
L	a random variable denoting the time lag.
$\mathbf{\Theta}$	the parameters to model the time lag.
z_{ij}	an indicator variable to determine whether the i^{th} event A implies the j^{th} event B.
π_{ij}, r_{ij}	the probability of $z_{ij} = 1$.
σ^2	the variance of time lag distribution.
m	the number of event A.
n	the number of event B.

random variable used to represent the noise during data collection. Because of the noise, the observed time lag between a_i and b_j is not constant. Since μ is a constant, the time lag $L = \mu + \epsilon$ is a random variable.

Definition 4.3.1 *The temporal dependency between A and B is denoted as $A \rightarrow_L B$, which means that the occurrence of A is followed by the occurrence of B with a time lag L. Here L is a random variable.*

In order to discover the temporal dependency rule $A \to_L B$, we need to learn the distribution of random variable L.

We assume that the distribution of L is determined by the parameters $\mathbf{\Theta}$, which is independent from the occurrence of A. The occurrence of an event B is defined by the time lag L and the occurrence of A. Thus, the problem is equivalent to learning the parameter $\mathbf{\Theta}$ for the distribution of L. The intuitive idea to solve this problem is to find maximal likelihood parameter $\mathbf{\Theta}$ given both sequences S_A and S_B. It is expressed formally by the following Equation (4.2).

$$\hat{\mathbf{\Theta}} = \arg\max_{\mathbf{\Theta}} P(\mathbf{\Theta}|\mathbf{S_A}, \mathbf{S_B}). \tag{4.2}$$

The value of $P(\mathbf{\Theta}|\mathbf{S_A}, \mathbf{S_B})$ in Equation (4.2) can be obtained with the Bayes Theory.

$$P(\mathbf{\Theta}|\mathbf{S_A}, \mathbf{S_B}) = \frac{P(\mathbf{S_B}|\mathbf{S_A}, \mathbf{\Theta}) \times P(\mathbf{\Theta}) \times P(\mathbf{S_A})}{P(\mathbf{S_A}, \mathbf{S_B})}. \tag{4.3}$$

Applying ln to both sides of Equation (4.3), we get:

$$\begin{aligned} \ln P(\mathbf{\Theta}|\mathbf{S_A}, \mathbf{S_B}) = {} & \ln P(\mathbf{S_B}|\mathbf{S_A}, \mathbf{\Theta}) + \ln P(\mathbf{\Theta}) \\ & + \ln P(\mathbf{S_A}) - \ln P(\mathbf{S_A}, \mathbf{S_B}). \end{aligned} \tag{4.4}$$

In Equation (4.4), only $\ln P(\mathbf{S_B}|\mathbf{S_A}, \mathbf{\Theta})$ and $\ln P(\mathbf{\Theta})$ are related to $\mathbf{\Theta}$. A large number of small factors contribute to the time lag L and we do not have prior knowledge about the distribution of $\mathbf{\Theta}$. Thus, given a non-informative prior distribution for $\mathbf{\Theta}$, we mainly focus on the likelihood defined in the first term. As a result, the problem is reduced to maximizing the likelihood defined by

$$\hat{\mathbf{\Theta}} = \arg\max_{\mathbf{\Theta}} \ln P(\mathbf{S_B}|\mathbf{S_A}, \mathbf{\Theta}). \tag{4.5}$$

Therefore, the temporal dependency $A \to_L B$ can be found by solving Equation (4.5).

4.3.2 Computing Log-likelihood

To solve Equation (4.5) we need to compute the log-likelihood $\ln P(\mathbf{S_B}|\mathbf{S_A}, \mathbf{\Theta})$.

Given the sequence $\mathbf{S_A}$ and value of parameters $\mathbf{\Theta}$, we assume that time stamps b_j in $\mathbf{S_B}$ are mutually independent if event B is caused by A. This assumption allows us to advance our calculation while thorough tests of the final results are described in latter sections.

Therefore,

$$P(\mathbf{S_B}|\mathbf{S_A}, \mathbf{\Theta}) = \prod_{j=1}^{n} P(b_j|\mathbf{S_A}, \mathbf{\Theta}). \tag{4.6}$$

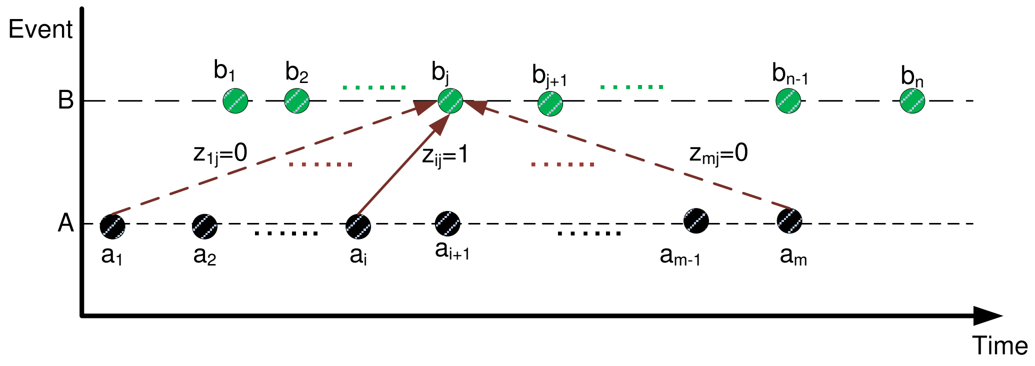


Figure 4.4: The j^{th} event B occurring at b_j can be implied by any event A. Variable $z_{ij} = 1$ if the j^{th} event B is associated with i^{th} event A, and 0 otherwise.

Given the sequence of time stamps $\mathbf{S_A}$ of event A, the instance of event B occurring at b_j is identified by possible instances of A happening at a specific time stamp a_i in the sequence $\mathbf{S_A}$ as shown in Figure 4.4. In order to model the relation between a_i and b_j, we introduce a latent variable z_{ij} defined as follows

$$z_{ij} = \begin{cases} 1, & \textit{the } i^{th} \textit{ event } A \textit{ implies the } j^{th} \textit{ event } B; \\ 0, & \textit{otherwise.} \end{cases} \tag{4.7}$$

Thus, given the sequence $\mathbf{S_A}$, each b_j is associated with a binary vector $\mathbf{z_{\bullet j}}$, where each component is either 1 or 0 and only one component of $\mathbf{z_{\bullet j}}$ is 1. For instance in Figure 4.4, only the i^{th} component z_{ij} is 1 since a_i implies b_j, and the remaining components are 0s. We apply the latent matrix $\mathbf{Z} = \{z_{ij}\}_{m\times n}$ to denote the relation mapping between two sequences $\mathbf{S_A}$ and $\mathbf{S_B}$.

Using $\mathbf{Z}$ we obtain the following equations:

$$P(b_j|\mathbf{z_{\bullet j}}, \mathbf{S_A}, \mathbf{\Theta}) = \prod_{i=1}^{m} P(b_j|a_i, \mathbf{\Theta})^{z_{ij}}. \tag{4.8}$$

$$P(\mathbf{z_{\bullet j}}) = \prod_{i=1}^{m} P(z_{ij} = 1)^{z_{ij}}. \tag{4.9}$$

Combining Equations (4.8) and (4.9), we rewrite $P(b_j|\mathbf{S_A}, \mathbf{\Theta})$ as follows:

$$P(b_j, \mathbf{z_{\bullet j}}|\mathbf{S_A}, \mathbf{\Theta}) = \prod_{i=1}^{m} (P(b_j|a_i, \mathbf{\Theta}) \times P(z_{ij} = 1))^{z_{ij}}. \tag{4.10}$$

Furthermore, the joint probability $P(b_j|\mathbf{S_A}, \mathbf{\Theta})$ in Equation (4.10) is described by Proposition 4.3.2.

Proposition 4.3.2 (marginal probability) *Given* $\mathbf{S_A}$ *and* $\mathbf{\Theta}$*, the marginal probability of* b_j *is as follows*

$$P(b_j|\mathbf{S_A}, \mathbf{\Theta}) = \sum_{i=1}^{m} P(z_{ij} = 1) \times P(b_j|a_i, \mathbf{\Theta}). \tag{4.11}$$

Proof. (Proposition 4.3.2). The marginal probability is acquired by summing up the joint probability over all the $\mathbf{z_{\bullet j}}$, i.e.,

$$P(b_j|\mathbf{S_A}, \mathbf{\Theta}) = \sum_{\mathbf{z_{\bullet j}}} \prod_{i=1}^{m} (P(b_j|a_i, \mathbf{\Theta}) \times P(z_{ij} = 1))^{z_{ij}}.$$

Among all m components in vector $\mathbf{z_{\bullet j}}$, there is only one component with value 1. Without any loss of generality, let $z_{ij} = 1$ given $\mathbf{z_{\bullet j}}$. Then,

$$\prod_{i=1}^{m} (P(b_j|a_i, \mathbf{\Theta}) \times P(z_{ij} = 1))^{z_{ij}} = P(b_j|a_i, \mathbf{\Theta}) \times P(z_{ij} = 1).$$

Thus,

$$P(b_j|\mathbf{S_A}, \mathbf{\Theta}) = \sum_{\mathbf{z_{\bullet j}}} P(b_j|a_i, \mathbf{\Theta}) \times P(z_{ij} = 1).$$

There are m different $\mathbf{z_{\bullet j}}$ with $z_{ij} = 1$ where i ranges from 1 to m. Thus,

$$P(b_j|\mathbf{S_A}, \mathbf{\Theta}) = \sum_{i=1}^{m} P(z_{ij} = 1) \times P(b_j|a_i, \mathbf{\Theta}).$$

□

Based on Equation (4.5),(4.6) and (4.11), the log-likelihood is:

$$\ln P(\mathbf{S_B}|\mathbf{S_A}, \mathbf{\Theta}) = \sum_{j=1}^{n} \ln \sum_{i=1}^{m} P(z_{ij} = 1) \times P(b_j|a_i, \mathbf{\Theta}). \tag{4.12}$$

According to Equation (4.12), the evaluation of log-likelihood relies on the description of $P(b_j|a_i, \mathbf{\Theta})$. The explicit form of $P(b_j|a_i, \mathbf{\Theta})$ expressed in terms of time lag model is presented in the following section.

4.3.3 Modeling Time Lag

According to the discussion regarding Equation (4.1), the time lag L is a random variable that is the sum of the true time lag μ and the noise ϵ. The noise contributed to the true lag is as a result of diverse factors, such as missing records, incorrect values, recording delay and so forth that happen during log collecting. To illustrate the noise of the time lags, a set of event instances with a single event type (i.e., SVC_TEC_HEARTBEAT) are collected from the real IT environment by IBM Tivoli monitoring system [urla]. The event SVC_TEC_HEARTBEAT is periodically triggered to monitor the status of service every ten minutes. The time lags between all close pairs are computed and the time lag distribution are given in Figure 4.5. As it is shown, a sharp spike in the number of occurrences happens at the time lag around 600 seconds, and much less frequent occurrences at other time lags.

According to the time lag distribution from the real event data and in light of the Central Limit Theorem, it is reasonable to assume that the noise ϵ follows the normal distribution with zero-mean value, since we can always move the mean of the distribution to the constant μ. Let σ^2 be the variance of the lags distribution. Then,

$$\epsilon \sim \mathcal{N}(0, \sigma^2). \tag{4.13}$$

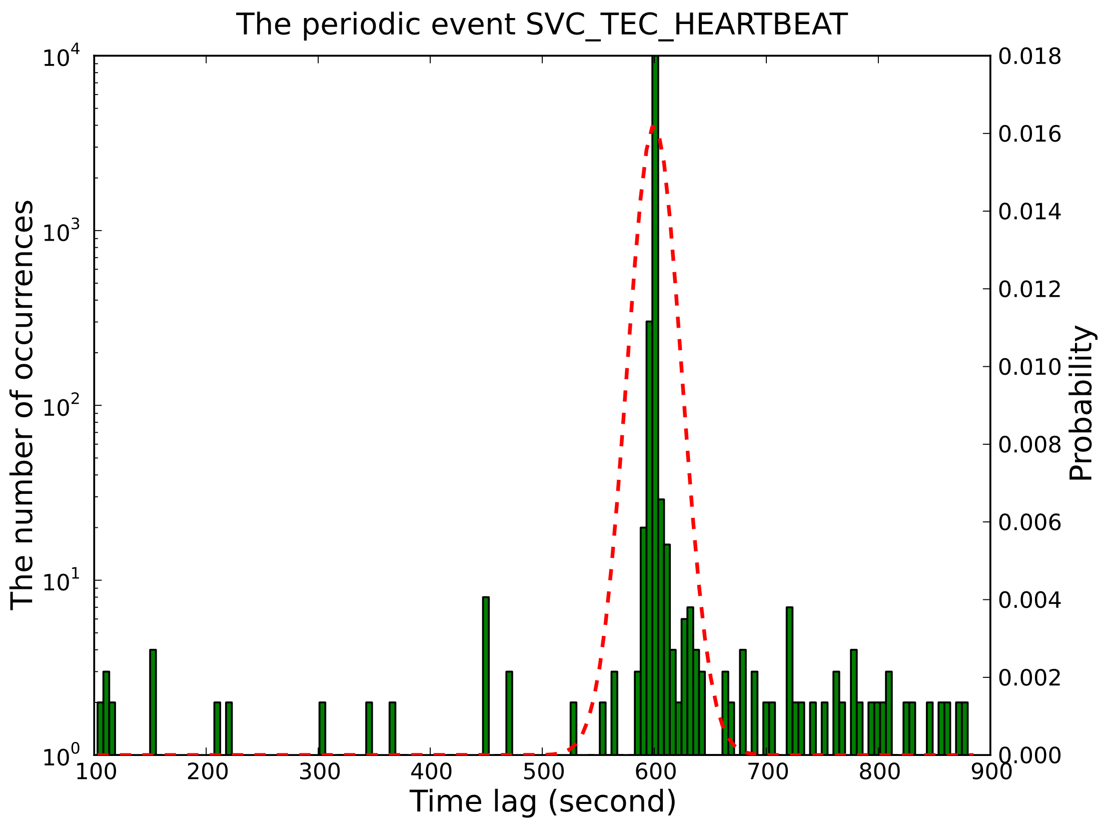


Figure 4.5: It shows the time lag distribution of event SVC_TEC_HEARTBEAT from the real data collected by IBM Tivoli monitoring system [urla]. The time lag distribution is fitted with a normal distribution $\mathcal{N}(600.01, 24.60^2)$. The spikes on counts diagram for large deviation from average are typical for log scale when counting function is fitted by normal distribution.

Since $L = \mu + \epsilon$ where μ is a constant, the distribution of L can be expressed as

$$L \sim \mathcal{N}(\mu, \sigma^2). \tag{4.14}$$

Parameters $\mathbf{\Theta}$ determines the distribution of L. Based on the model of L described in Equation (4.14), apparently $\mathbf{\Theta} = (\mu, \sigma^2)$. Thus, the discovery of time lag L is reduced to learning the parameters μ and σ^2.

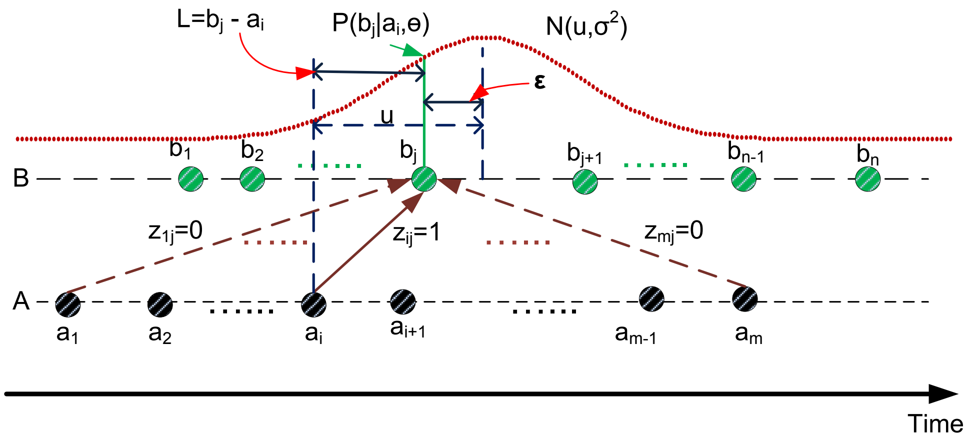


Figure 4.6: $A \rightarrow_L B$, where $L \sim \mathcal{N}(\mu, \sigma^2)$. An event A that occurred at time a_i is associated to an event B that occured at b_j with probability $\mathcal{N}(b_j - a_i|\mu, \sigma^2)$. Here μ is the expected time lag of an occurrence of B after a_i.

Assume that the event A is followed by the event B with a time lag L, here $L \sim \mathcal{N}(\mu, \sigma^2)$. Specifically, as shown in Figure 4.6, the i^{th} event A is associated to the j^{th} event B where the time lag $(b_j - a_i)$ between the two events is a random variable L distributed as $\mathcal{N}(b_j - a_i|\mu, \sigma^2)$. Thus,

$$\begin{aligned} P(b_j|a_i, \mathbf{\Theta}) &= P(b_j|a_i, \mu, \sigma^2) \\ &= \mathcal{N}(b_j - a_i|\mu, \sigma^2). \end{aligned} \tag{4.15}$$

Hence, by replacing $P(b_j|a_i, \mathbf{\Theta})$ based on Equation (4.15), the log-likelihood in equation (4.12) is expressed as:

$$\ln P(\mathbf{S_B}|\mathbf{S_A}, \mathbf{\Theta}) = \sum_{j=1}^{n} \ln \sum_{i=1}^{m} P(z_{ij} = 1) \times \mathcal{N}(b_j - a_i|\mu, \sigma^2). \tag{4.16}$$

Here $P(z_{ij} = 1)$ denotes the probability that the j^{th} event B is implied by the i^{th} event A. Assume that there are m events A, so we assume that

$$\sum_{i=1}^{m} P(z_{ij} = 1) = 1.$$

To simplify the description, let

$$\pi_{ij} = P(z_{ij} = 1).$$

Based on the expression of log-likelihood in Equation (4.16), the Equation (4.5) is equivalent to the following

$$\begin{aligned} &(\hat{\mu}, \hat{\sigma}^2) = \arg\max_{\mu,\sigma^2} \sum_{j=1}^{n} \ln \sum_{i=1}^{m} \pi_{ij} \times \mathcal{N}(b_j - a_i|\mu, \sigma^2). \\ &\mathbf{s.t.} \sum_{i=1}^{m} \pi_{ij} = 1 \end{aligned} \tag{4.17}$$

We describe the algorithms to maximize the log-likelihood of parameters μ and σ^2 in the following section.

4.4 Algorithm and Solution

4.4.1 Maximize Log-likelihood

Equation (4.17) is an optimization problem. Gradient ascent method is supposed to be used to solve it. However, this method is not applicable here since we cannot directly derive the closed-form partial derivatives with respect to the unknown parameters μ and σ^2. The problem described by Equation (4.17) is a typical mixture model. It can be solved by using iterative expectation maximization i.e., EM-based method [B+06].

Given $\mathbf{S_A}$ and $\mathbf{\Theta}$, by the Equation (4.10), the expectation of $\ln P(\mathbf{S_B}, \mathbf{Z}|\mathbf{S_A}, \mathbf{\Theta})$ with respect to $P(z_{ij}|\mathbf{S_B}, \mathbf{S_A}, \mathbf{\Theta}')$ is as follows:

$$\begin{aligned} &E(\ln P(\mathbf{S_B}, \mathbf{Z}|\mathbf{S_A}, \mathbf{\Theta})) = \\ &\sum_{j=1}^{n} \sum_{i=1}^{m} E(z_{ij}|\mathbf{S_B}, \mathbf{S_A}, \mathbf{\Theta}') \times (\ln \pi_{ij} + \ln \mathcal{N}(b_j - a_i|\mu, \sigma^2)), \end{aligned} \tag{4.18}$$

where $\mathbf{\Theta}'$ is the parameter estimated on the previous iteration.

Since z_{ij} is an indicator variable,

$$E(z_{ij}|\mathbf{S_B},\mathbf{S_A},\mathbf{\Theta}') = P(z_{ij}=1|\mathbf{S_B},\mathbf{S_A},\mathbf{\Theta}').$$

Let

$$r_{ij} = E(z_{ij}|\mathbf{S_B},\mathbf{S_A},\mathbf{\Theta}').$$

Then,

$$r_{ij} = \frac{\pi'_{ij} \times \mathcal{N}(b_j - a_i|\mu',\sigma'^2)}{\sum_i^m \pi'_{ij} \times \mathcal{N}(b_j - a_i|\mu',\sigma'^2)}. \tag{4.19}$$

The new parameters π_{ij} as well as μ and σ^2 can be learned by maximizing $E(\ln P(\mathbf{S_B},\mathbf{Z}|\mathbf{S_A},\mathbf{\Theta}))$

$$\mu = \frac{1}{n}\sum_{j=1}^{n}\sum_{i=1}^{m} r_{ij}(b_j - a_i), \tag{4.20}$$

$$\sigma^2 = \frac{1}{n}\sum_{j=1}^{n}\sum_{i=1}^{m} r_{ij}(b_j - a_i - \mu)^2, \tag{4.21}$$

$$\pi_{ij} = r_{ij}. \tag{4.22}$$

Based on Equation (4.22), Equation (4.19) is equivalent to the following:

$$r_{ij} = \frac{r'_{ij} \times \mathcal{N}(b_j - a_i|\mu',\sigma'^2)}{\sum_i^m r'_{ij} \times \mathcal{N}(b_j - a_i|\mu',\sigma'^2)}. \tag{4.23}$$

To find maximum likelihood estimates of parameters, we use EM-based algorithm $lagEM$ (described in Algorithm 3). In Algorithm 3, the parameters are initialized by the code from line 2 to line 5. Since there are $m \times n$ entries r'_{ij}, the time complexity for initialization is $O(mn)$. The optimized parameters are acquired by an iterative procedure from line 6 to line 14, which involves expectation and maximization. The iterative procedure is not terminated until the parameters converge. Let r be the total number of iterations executed. The expectation is implemented by line 7. The time cost of expectation is $O(mn)$ because of $m \times n$ entries r_{ij} to evaluated. The

Algorithm 3 LagEM

```
1: procedure $LagEM(\mathbf{S_A},\mathbf{S_B})$
   ▷Input: two event sequences $\mathbf{S_A}$ and $\mathbf{S_B}$ with length $m$ and $n$ respectively.
   ▷Output: the estimated parameters $\mu$ and $\sigma^2$.
2:     define $r'_{ij}$, $\mu'$ and $\sigma'^2$ as parameters of previous iteration
3:     define $r_{ij}$, $\mu$ and $\sigma^2$ as the parameters of current iteration
   ▷ initialization
4:     initialize $r'_{ij} = \frac{1}{m}$
5:     initialize $\mu'$ and $\sigma'^2$ randomly
6:     while true do
   ▷ expectation
7:         evaluate the $r_{ij}$ following Equation (4.23)
   ▷ maximization
8:         update $\mu$ following Equation (4.20)
9:         update $\sigma^2$ following Equation (4.21)
   ▷test convergence
10:        if parameters converge then
11:            return $\mu$ and $\sigma^2$
12:        end if
13:    end while
14: end procedure
```

maximization part is shown from line 8 to line 10. It takes the time cost of $O(mn)$ to update parameters of current iteration according to Equation (4.20),(4.21). Therefore, the time complexity of iterative procedure is $O(rmn)$. The time cost of Algorithm *lagEM* is $O(rmn)$, where m and n are the number of events A and B, respectively, and r is the number of iterations needed for parameters to stabilize. As the time span of event sequence grows, more events will be collected. Since m and n are the counts of two types of events, it is reasonable to assume that m and n have the same order of magnitude. Therefore, the time cost of Algorithm *lagEM* is a quadratic function of events count.

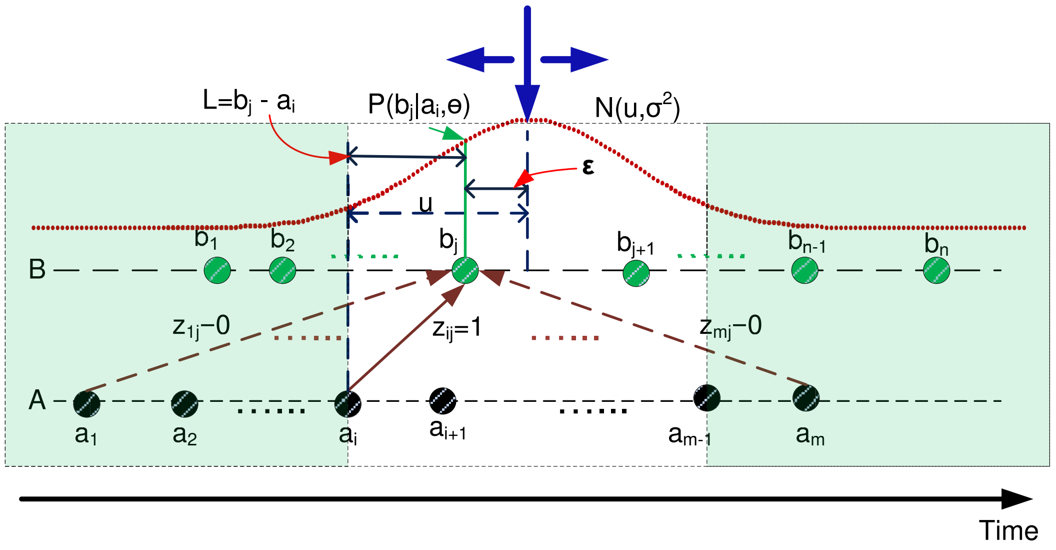


Figure 4.7: Approximate estimation for new parameters μ and σ^2.

4.4.2 Approximate Inference

Observation 1 *During each iteration of Algorithm lagEM, the probability r_{ij} describing the likelihood that the j^{th} event B is implied by the i^{th} event A, becomes smaller when the deviation of $b_j - a_i$ from the estimated time lag μ increases.*

Thus, as $|b_j - a_i - \mu|$ becomes larger, r_{ij} approaches 0, as shown in Figure 4.7. Further, if r_{ij} is small enough, the contribution by b_j and a_i to estimate the new parameters μ and σ^2 according to Equation (4.20) and (4.21) is negligible. As a matter of fact, the time span of the sequence of events is very long. Hence, most of r_{ij} are small. Therefore, we can estimate new parameters μ and σ^2 without significant loss of accuracy by ignoring terms $r_{ij}(b_j - a_i)$ and $r_{ij}(b_j - a_i - \mu)$ with small r_{ij} in both Equation (4.20) and (4.21). The ignoring parts are illustrated with shadow in Figure 4.7. During each iteration of Algorithm *lagEM*, given b_j, we can boost Algorithm *lagEM* by not summing up all the m components for parameters estimation.

Given b_j, let ξ_j be the sum of the probabilities r_{ij} whose component is neglected during the iteration. That is,

$$\xi_j = \sum_{\{i|a_i \ is \ neglected\}} r_{ij}.$$

Let ξ be the largest one among all the ξ_j, i.e., $\xi = \max_{1\leq j\leq n} \{\xi_j\}$. Let μ_δ and σ_δ^2 be neglected parts in the estimate μ and σ^2 during each iteration. Formally, we get,

$$\mu_\delta = \frac{1}{n}\sum_{j=1}^{n} \sum_{\{i|a_i \ is \ neglected\}} r_{ij}(b_j - a_i),$$

$$\sigma_\delta^2 = \frac{1}{n}\sum_{j=1}^{n} \sum_{\{i|a_i \ is \ neglected\}} r_{ij}(b_j - a_i)^2.$$

The following lemma allows to bound the neglected part μ_δ and σ_δ^2.

Lemma 4.4.1 *Let $\bar{b}$ be the mean of all the time stamps of event B, i.e.*

$$\bar{b} = \frac{1}{n}\sum_{j=1}^{n} b_j.$$

Let $\bar{b^2}$ be the second moment of the time stamps of event B, i.e.,

$$\bar{b^2} = \frac{1}{n}\sum_{j=1}^{n} b_j^2$$

Then we get:

$$\mu_\delta \in [\xi(\bar{b} - a_m), \xi(\bar{b} - a_1)]. \tag{4.24}$$

Let $\phi = \max\{\bar{b^2} - 2\bar{b}a_1 + a_1^2, \bar{b^2} - 2\bar{b}a_1 + a_m^2\}$, then

$$\sigma_\delta^2 \in [0, \xi\phi]. \tag{4.25}$$

The proof of Lemma 4.4.1 is provided as follows.

Proof. (Lemma 4.4.1.) Given a time sequence

$$< a_1, a_2, ..., a_m >,$$

it is reasonable to assume that

$$a_1 \leq a_2 \leq ... \leq a_m.$$

Thus,

$$b_j - a_i \in [b_j - a_m, b_j - a_1].$$

Moreover,

$$\xi_j = \sum_{i|a_i \ is \ neglected} r_{ij},$$

where $\xi_j \leq \xi$. Therefore,

$$\frac{1}{n}\sum_{j=1}^{n} \xi(b_j - a_m) \leq \mu_\delta \leq \frac{1}{n}\sum_{j=1}^{n} \xi(b_j - a_1).$$

Then, we get

$$\mu_\delta \in [\xi(\bar{b} - a_m), \xi(\bar{b} - a_1)].$$

In addition,

$$(b_j - a_i)^2 \leq \max\{(b_j - a_1)^2, (b_j - a_m)^2\}.$$

Thus,

$$\sigma_\delta^2 \leq \frac{1}{n}\xi \sum_{j=1}^{n} \max\{(b_j^2 - 2b_j a_1 + a_1^2, b_j^2 - 2b_j a_m + a_m^2)\}.$$

Then, we get

$$\sigma_\delta^2 \leq \xi \max\{\bar{b}^2 - 2\bar{b}a_1 + a_1^2, \bar{b}^2 - 2\bar{b}a_1 + a_m^2\}.$$

So,

$$\sigma_\delta^2 \in [0, \xi\phi].$$

□

Lemma 4.4.1 shows that if the ξ is small enough, $|\mu_\delta|$ and σ_δ^2 approach 0 then the parameters μ and σ^2 are close to the ones without ignoring components.

Given a time stamp b_j, there are m possible corresponding time stamps of event A. Our problem is to choose a subset C_j of time stamps of event A in order to estimate the parameters during each iteration. To guarantee that the probability of the neglected part is less than ξ, the probability for the subset C_j should be greater than $1-\xi$. In order to optimize the time complexity, we minimize the size of C_j. We solve it efficiently by applying a greedy algorithm, which adds a_i to C_j with its r_{ij} in decreasing order until summation of r_{ij} is greater than $1-\xi$.

Based on Observation 1 and the fact that all the time stamps of event A are in increasing order, the index i for time stamps of event A in C_j should be consecutive. Given b_j, the minimum and maximum indexes of a_i in C_j can be found by Algorithm *greedyBound* listed in Algorithm 4.

The time cost of *greedyBound* is $O(\log m + K)$ where $K = |C_j|$ and m is the number of events A. In Algorithm *greedyBound*, line 3 uses binary searching algorithm to locate the nearest a_i. It takes $O(\log m)$ time cost. The loop between line 6 and line 16 consumes $|C_j|$ time units. Let $K = |C_j|$. Then the total time complexity is $O(\log m + K)$.

Based on Lemma 4.4.1 and Algorithm *greedyBound*, we propose an approximation algorithm *appLagEM*. The detail of Algorithm *appLagEM* is given in Algorithm 5. We highlight the key differences between Algorithm 3 and Algorithm 5 by underlining.

In *appLagEM*, let K be the average size of all C_j. Then the time complexity of line 8 is $O(\log m + K)$ and it takes $O(K)$ for line 9. Thus, from line 7 to line 10, the complexity is $O(n(\log m + K))$. Both line 11 and line 12 consume $O(nK)$.

The total time cost of Algorithm *appLagEM* is $O(rn(\log m + K))$ where r is the number of iterations, and K is the average size of all C_j. Typically, in the event

Algorithm 4 greedyBound

```
1: procedure greedyBound($\mathbf{S_A}$,$b_j$,$\mu$, $\xi$)
   ▷Input: $\mathbf{S_A}$ contains all the possible time stamps of event $A$. $b_j$ is the time stamp of the $j^{th}$ event B. $\mu$ is the mean of time lags estimated in the previous iteration. $\xi$ is the probability that the time stamps of event $A$ not in $C_j$.
   ▷Output: $min_j$ and $max_j$ are the minimum and maximum indexes in $C_j$.
2:     $t = b_j - \mu$
3:     Locate the $a_i$ to which $t$ is closed using binary search.
4:     $min_j = i$ and $max_j = i$
5:     $prob = 0.0$
6:     while $prob < 1 - \xi$ do
7:         if $r_{(min_j-1)j} \geq r_{(max_j+1)j}$ then
8:             $i = min_j - 1$
9:             $min_j = i$
10:        else
11:            $i = max_j + 1$
12:            $max_j = i$
13:        end if
14:        add $a_i$ to $C_j$
15:        $prob = prob + r_{ij}$
16:    end while
17:    return $min_j$ and $max_j$.
18: end procedure
```

sequence, $K << n$ and $\log m << n$. Therefore, the time cost of algorithm *appLagEM* is close to a linear function of n in each iteration.

4.5 Experiments

4.5.1 Setup

The performance of proposed algorithms is evaluated by using both synthetic and real event data. The importance of an experiment conducted over synthetic data lies in the fact that the ground truth can be provided in advance. To generate synthetic data, we can fix time lag between dependent events and add noise into synthetic data.

Algorithm 5 appLagEM

```
1: procedure appLagEM($\mathbf{S_A}$,$\mathbf{S_B}$,$\xi$)
   ▷Input: two event sequences $\mathbf{S_A}$ and $\mathbf{S_B}$ with length $m$ and $n$ respectively. $\xi$ is
   the probability of the neglected part for estimating parameters.
   ▷Output: the estimated parameters $\mu$ and $\sigma^2$.
2:     define $r'_{ij}$, $\mu'$ and $\sigma'^2$ as parameters of previous iteration
3:     define $r_{ij}$, $\mu$ and $\sigma^2$ as the parameters of current iteration
   ▷ initialization
4:     initialize $r'_{ij} = \frac{1}{m}$
5:     initialize $\mu'$ and $\sigma'^2$ randomly
6:     while true do
7:         for each $b_j$ do
           ▷find the index bound of $a$ for each $b_j$
8:             Get $min_j$ and $max_j$ by greedyBound
           ▷ expectation
9:             evaluate the $r_{ij}$ where $i \in [min_j, max_j]$
10:        end for
       ▷ maximization
11:        update $\mu$ by Equation (4.20) within the bound
12:        update $\sigma^2$ by Equation (4.21) within the bound
       ▷test convergence
13:        if parameters converge then
14:            return $\mu$ and $\sigma^2$
15:        end if
16:    end while
17: end procedure
```

The empirical study over the synthetic data allows us to demonstrate the effectiveness and efficiency of proposed algorithms.

The experiments over the real data collected from real production environments shows that temporal dependencies with time lags can be discovered by running our proposed algorithm, providing additional support for our assumptions and concluding formulas. Detailed analysis of discovered temporal dependencies allows us to demonstrate the effectiveness and usefulness of our algorithm in practice.

All algorithms are implemented using Java 1.7. All experiments are conducted on the experimental environment running Linux 2.6.32. The computer is equipped with

Intel(R) Xeon(R) CPU with 24 cores running at speed of 2.50GHZ. The total volume of memory is 158 Gb.

4.5.2 Synthetic Data

Synthetic data generation

In this part we describe experiments conducted on six synthetic data sets. The synthetic data generation is defined by the parameters shown in Table 5.2.

Table 4.2: Parameters for synthetic data generation.

Name	Description
β_{min}	Describes the minimum value for choosing the average inter-arrival time β.
β_{max}	Describes the maximum value for choosing the average inter-arrival time β.
N	The number of events in the synthetic event sequence.
μ_{min}	Describes the minimum value for the true time lag μ.
μ_{max}	Describes the maximum value for the true time lag μ.
σ^2_{min}	Describes the minimum value for the variance of time lag.
σ^2_{max}	Describes the maximum value for the variance of time lag.

We employ the exponential distribution to simulate the inter-arrival time between two adjacent events [LLMP05]. The average inter-arrival time β is randomly generated in the range $[\beta_{min}, \beta_{max}]$. The true lag μ is randomly generated in the range

$[\mu_{min}, \mu_{max}]$. And the variance of time lag σ^2 is generated between σ^2_{mix} and σ^2_{max} randomly.

With chosen parameters β, μ and σ^2, the procedure of generating synthetic data for the temporal dependency $A \rightarrow_{\mu} B$ is given below.

- Generate N time stamps for event A, where the inter-arrival time between two adjacent events follows the exponential distribution with parameter β.
- For each time stamp a_i for event A, the time lag is randomly generated according to normal distribution with parameters μ and σ^2.
- Combine all the time stamps associated with their types to build a synthetic data set.

We set $\beta_{min} = 5$, $\beta_{max} = 50$, $\mu_{min} = 25$, $\mu_{max} = 100$, $\sigma^2_{min} = 5$ and $\sigma^2_{max} = 400$ to synthesize the six data sets with different parameters N. The numbers of events for the synthetic data sets are 200, $1k$, $2k$, $10k$, $20k$ and $40k$, respectively. Recall that the number of events only describes the number of events of two types we are interested in. In practice, a real data set typically gets more than hundreds of events types in addition to the two considered types of events. Thus $40k$ events of two types compare with a real data set containing 2 million events of 100 types.

Synthetic data evaluation

Since the EM based approach cannot guarantee the global optimum [B$^+$06], we define a batch as running the experiments 20 rounds with different initial parameters chosen at random, where we empirically find out 20 rounds are reasonable for our problem, shown in Figure 4.10.

We choose the one with the maximum likelihood among 20 rounds as the solution of a batch. Ten such batches are conducted over each data set. With 10 pairs of

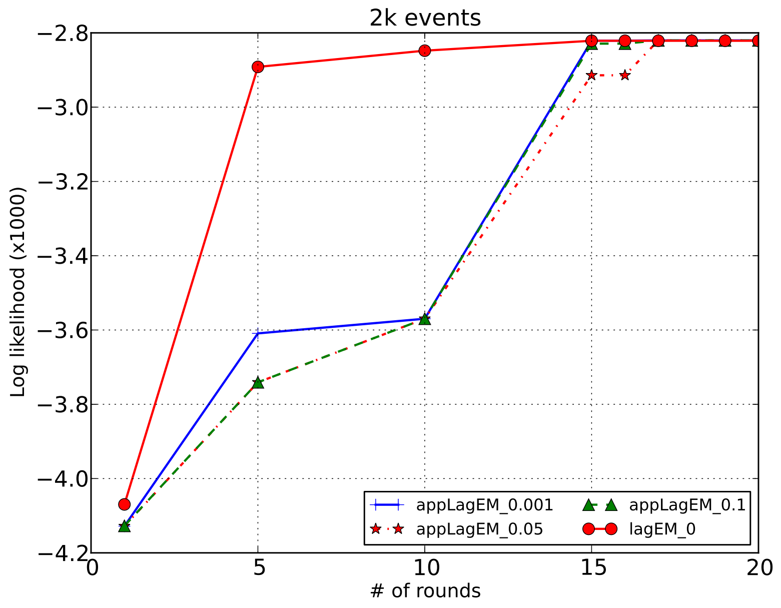


Figure 4.8: The number of events is 2k. The maximum log likelihood improves as the number of rounds increases. It becomes stable when the number of rounds reaches 20.

parameters μ and σ^2 learnt from 10 batches, $\bar{\mu}$ and $\bar{\sigma}^2$ are calculated as an average values of μ and σ^2, respectively. Furthermore, 95% confidence intervals of μ and σ^2 are provided assuming both μ and σ^2 follow the normal distribution as the prior [B+06]. Additionally, LL_{opt} denotes the maximum log-likelihood value learnt by our proposed algorithms. Results of experiments running $lagEM$ and $appLagEM$ are presented in Table 4.3.

Each algorithm stops searching as it converges or the number of iterations exceeds 500. Algorithm $appLagEM$ takes one more parameter ξ as its input, where ξ determines the proportion of the neglected components during the parameter estimation of each iteration. Herein, ξ has been set to 0.001, 0.05 and 0.1. For all data sets listed in Table 4.3, time lags μs learnt by $lagEM$ and $appLagEM$ are quite close to the ground truth. In addition, the smaller ξ is, the more probable that Algorithm $appLagEM$ will get a larger log likelihood.

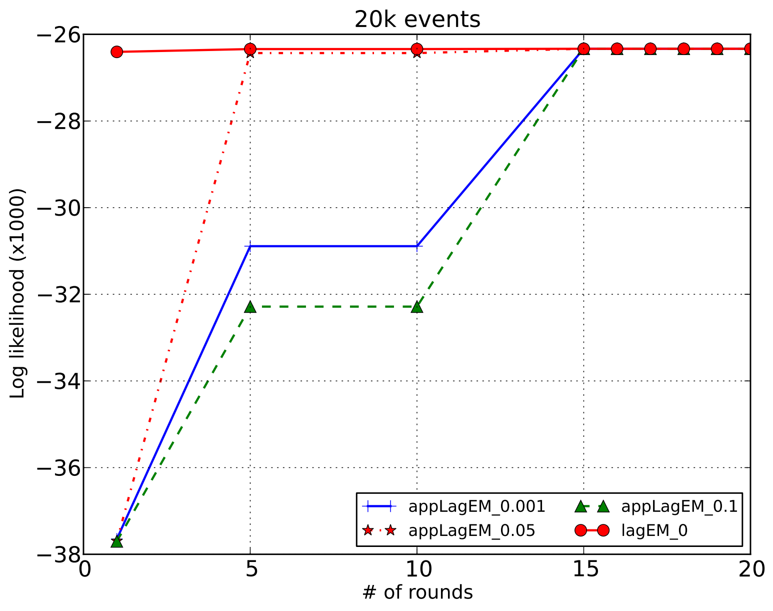


Figure 4.9: The number of events is 20k. The maximum log likelihood improves as the number of rounds increases. It becomes stable when the number of rounds reaches 20.

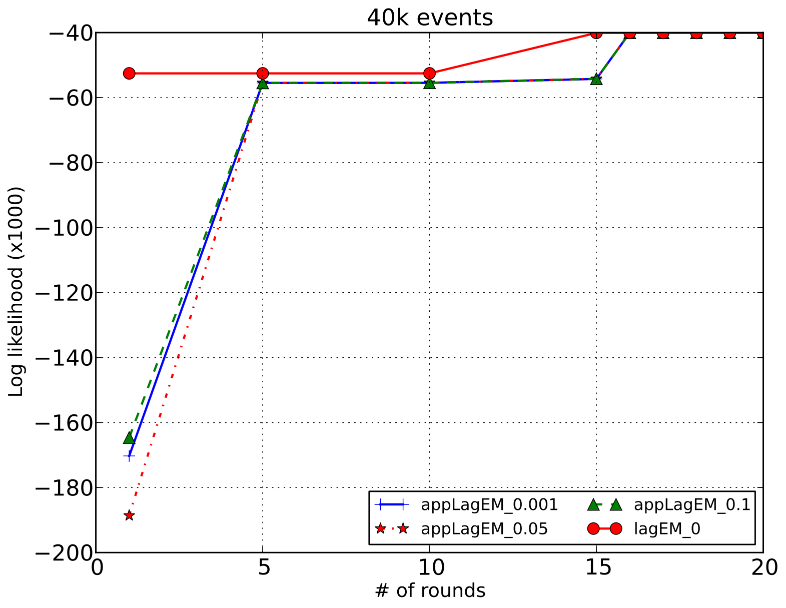


Figure 4.10: The number of events is 40k. The maximum log likelihood improves as the number of rounds increases. It becomes stable when the number of rounds reaches 20.

Table 4.3: The experimental result for synthetic data. The size of data ranges from 200 to 40k; $\bar{\mu}$ and $\bar{\sigma}^2$ are the average values of μ and σ^2; LL_{opt} is the maximum log-likelihood. Assuming μ and σ^2 follow normal distribution, $\bar{\mu}$ and $\bar{\sigma}^2$ are provided with their 95% confidence interval for each algorithm over every data set. Entries with "N/A" are not available since it takes more than 1 day to get corresponding parameters.

	Ground Truth		lagEM			appLagEM $\xi = 0.001$			appLagEM $\xi = 0.05$			appLagEM $\xi = 0.1$		
N	μ	σ^2	$\bar{\mu}$	$\bar{\sigma}^2$	LL_{opt}	$\bar{\mu}$	$\bar{\sigma}^2$	LL_{opt}	$\bar{\mu}$	$\bar{\sigma}^2$	LL_{opt}	$\bar{\mu}$	$\bar{\sigma}^2$	LL_{opt}
200	77.01	44.41	77.41 [73.46, 81.36]	20.74 [18.75, 22.73]	-292.47	77.85 [73.52, 82.18]	24.68 [20.64, 28.72]	-290.99	78.21 [74.38, 82.03]	24.79 [20.62, 28.96]	-299.89	78.135 [74.16, 82.11]	25.02 [21.24, 28.80]	-300.05
1k	25.35	12.51	25.5 [25.0, 25.98]	8.66 [8.52, 8.80]	-1275.5	25.45 [25.12, 25.78]	8.62 [8.52, 8.72]	-1247.01	25.94 [24.39, 27.49]	9.296 [5.83, 12.77]	-1248.34	25.97 [24.35, 27.59]	9.36 [5.71, 13.0]	-1248.35
2k	38.54	30.88	38.68 [38.19, 39.17]	16.57 [16.38, 16.75]	-2847.0	38.81 [38.42, 39.40]	16.51 [16.45, 16.57]	-2820.6	39.78 [37.76, 41.78]	17.82 [14.17, 21.47]	-2822.60	39.32 [37.49, 41.14]	17.26 [14.36, 20.16]	-2822.57
10k	54.92	13.51	55.07 [54.60, 55.54]	8.84 [8.68, 9.0]	-12525.0	55.82 [53.97, 57.66]	10.92 [5.24, 16.60]	-12523.68	55.29 [54.23, 56.34]	9.40 [7.28, 11.52]	-12526.0	55.80 [53.99, 57.60]	10.85 [5.17, 16.53]	-12526.04
20k	59.35	17.22	59.42 [59.27, 59.57]	11.35 [11.32, 11.40]	-26554.2	59.67 [58.86, 60.50]	11.7 [10.35, 13.05]	-26332.68	59.38 [59.1, 59.70]	11.38 [11.30, 11.5]	-26332.06	59.34 [58.96, 59.72]	11.42 [11.2, 11.63]	-26336.39
40k	80.18	8.48	N/A	N/A	N/A	82.51 [77.76, 87.25]	5.26 [0.3, 10.3]	-40024.01	81.7 [78.15, 85.25]	4.45 [0.86, 8.04]	-40187.73	81.59 [77.9, 85.3]	4.4 [0.85, 7.94]	-40185.64

Further, we employ the Kullback-Leibler(KL) divergence as the metric to measure the difference between the distribution of time lag given by the ground truth and the discovered results [KL51]. Since each algorithm with a different initial setting of parameters runs for 10 batches over a given data set, we take the average KL divergence of 10 batches to evaluate the experimental result. As shown in Figure 4.11, the KL divergence caused by $appLagEM$ is almost as small as the one produced by $lagEM$. Since the formulated problem for time lag is not convex. It may end with a local optimal value, partially depending on the randomly initialized parameters. Therefore, it may get a little bit high KL divergence when ξ is small. However, the result shows that the accuracy does not lose much when ξ gets large. It leads us to draw the conclusion that it is reasonable to apply the approximation algorithm for time lag discovery. Besides, as ξ increases, the KL divergence of $appLagEM$ becomes larger.

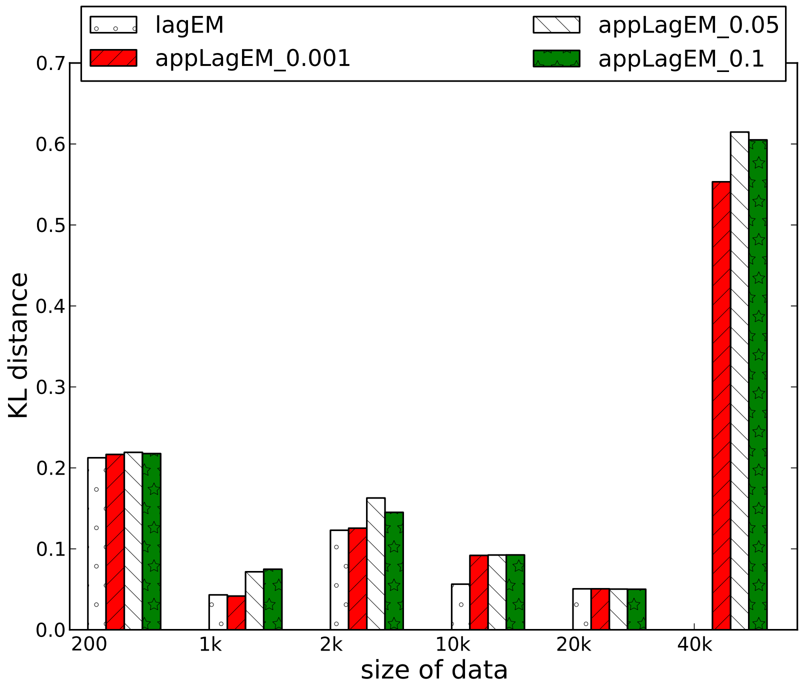


Figure 4.11: The KL distance between the ground truth and the one learnt by each algorithm. Note that $lagEM$ is missing for over 40k events since it takes more than one day to evaluate, see also Table 2.

Figure 4.12 presents the comparison of time cost over the synthetic data sets. It shows that the approximation algorithm $appLagEM$ is much more efficient than $lagEM$. It also shows that the larger the ξ is, the less time $appLagEM$ takes to find the optimal distribution of the time lags. Algorithm $appLagEM$ even with $\xi = 0.001$ is about two orders of magnitude faster than Algorithm $lagEM$. It also demonstrates the efficiency of our approximation algorithm comparing with the algorithm for mining $TPattern$ [LM04] and the $STScan$[TLS12] algorithm for time lag discovery. In order to find the time lag for rule $A \rightarrow B$, $TPattern$ only considers the time lags between the given instance of event A and its K closest instances of event B. In this experiment, K is set to be 300. Therefore, the $TPattern$ mining algorithm is more efficient than $lagEM$, $appLagEM$ with small $\xi = 0.001$ and $STScan$ algorithm. However, as ξ increases, $appLagEM$ turns to be more efficient than $TPattern$ mining algorithm. $STScan$ algorithm considers all time lags between all possible events. It leverages sorted table [TLS12] to boost the speed for time lag discovery. Therefore, it is faster than $LagEM$ and $appLagEM$ with $\xi = 0.001$. But it is slower than $appLagEM$ with larger parameter ξ.

In conclusion, based on the comparative discussion of both $lagEM$ and $appLagEM$, it is possible to achieve a good balance in terms of accuracy and efficiency.

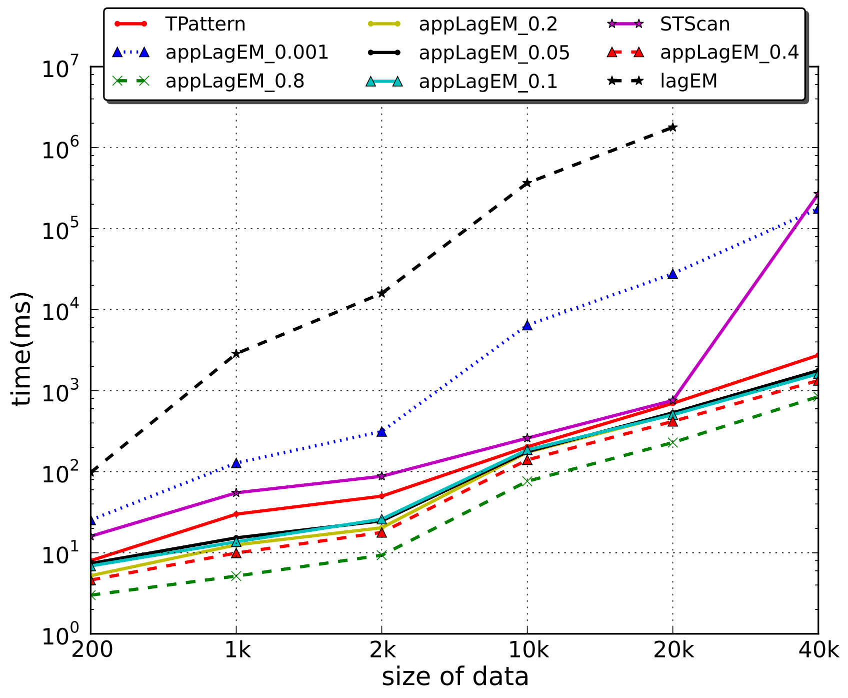


Figure 4.12: Time cost comparison. ξ of $appLagEM$ is set with 0.001, 0.05, 0.1, 0.2, 0.4, 0.8. The existing algorithm for mining TPattern and the algorithm STScan for mining time intervals are from [LM04] and [TLS12] respectively. The size of data set ranges from 200 to $40k$.

Table 4.4: The snippet of discovered time lags.

	Dependency	μ	σ^2	Signal-to-noise ratio
dataset1	$TEC\_Error \rightarrow_L Ticket\_Retry$	0.34059	0.107178	1.04
	$AIX\_HW\_ERROR \rightarrow_L AIX\_HW\_ERROR$	10.92	0.98	11.03
	$AIX\_HW\_ERROR \rightarrow_L NV390MSG\_MVS$	33.89	1.95	24.27
	$AIX\_HW\_ERROR \rightarrow_L Nvserverd\_Event$	64.75	2.99	37.45
	$AIX\_HW\_ERROR \rightarrow_L generic\_postemsg$	137.17	18.81	31.63
	$generic\_postemsg \rightarrow_L TSM\_SERVER\_EVENT$	205.301	39.36	32.72
	$generic\_postemsg \rightarrow_L Sentry2\_0\_diskusedpct$	134.51	71.61	15.90
	$MQ\_CONN\_NOT\_AUTHORIZED \rightarrow_L TSM\_SERVER\_EVENT$	1167.06	142.54	97.75
dataset2	$MSG\_Plat\_APP \rightarrow_L Linux\_Process$	18.53	2053.46	**0.408**
	$SVC\_TEC\_HEARTBEAT \rightarrow_L SVC\_TEC\_HEARTBEAT$	587.6	7238.5	6.90

4.5.3 Real Data

We perform the experiment over two real event data sets collected from several IT outsourcing centers by IBM Tivoli monitoring system [urla][TLP+12]. These events are generated by the automatic monitoring system with software agents running on the servers of an enterprise customer, which computes metrics for the hardware and software performance at regular intervals. The metrics are then compared to acceptable thresholds, known as monitoring situations, and any violation results in an alert. If the alert persists beyond a certain delay specified in the situation, the monitor emits an event. Therefore, a monitoring event corresponds to one type of system alert and one monitoring situation configured in the IBM Tivoli monitoring system.

Each real event set is collected from one IT environment of an enterprise customer. The number of events and types are listed in Table 4.5. The $dataset1$ consists of a sequence of events including 104 distinct event types, which are collected within the time span of 32 days. There are 136 types of events in $dataset2$ and $1000k$ events occurring within 54 days. In both data sets, hundreds of types of events result in tens of thousands of pairs of event types. Since our algorithm takes a pair of events as the input, it would be time-consuming to consider all the pairs. In order to efficiently find the time lag of most possible dependent events, we filter out the types of events that appear less than 100 times in a corresponding data set.

Table 4.5: Real event data set.

Name	# of events	# of types	Time span
dataset1	$100k$	104	32 days
dataset2	$1000k$	136	54 days

We employ the *appLagEM* with $\xi = 0.001$ to mine the time lag of temporal dependency between two events. To increase the probability of getting the global optimal value, we run the algorithm in a batch of 50 rounds by feeding in random initial parameters every round. The snippet of some interesting time lags discovered is shown as Table 4.4. The metric signal-to-noise ratio [Sch99], a concept in signal processing, is used to measure the impact of noise relative to the expected time lag. Signal-to-noise is given as below:

$$SNR = \frac{\mu}{\sigma}.$$

The larger the SNR, the less relative impact of noise to the expected time lags.

$TEC\_Error \rightarrow_L Ticket\_Retry$ is a temporal dependency discovered from $dataset1$, where time lag L follows the normal distribution with $\mu = 0.34$ and the variance $\sigma^2 = 0.107178$. The small expected time lag μ less than 0.1 seconds indicates that the two events appear almost at the same time. And the small variance shows that most of time lags between the two event types are around the expected time lag μ. In fact, TEC_Error is caused whenever the monitoring system fails to generate an incident ticket to the ticket system. And $Ticket\_Retry$ is raised when the monitoring system tries to generate the ticket again.

$AIX\_HW\_Error \rightarrow_L AIX\_HW\_Error$ in $dataset1$ describes a pattern related to the event AIX_HW_Error. With the discovered μ and σ^2, the event AIX_HW_Error

happens with an expected period about 10 seconds with small variance less than 1 seconds. In a real production environment, the event AIX_HW_Error is raised when monitoring system polls an AIX server which is down. The failure to respond to the monitoring system leads to an event AIX_HW_Error almost every 10 seconds.

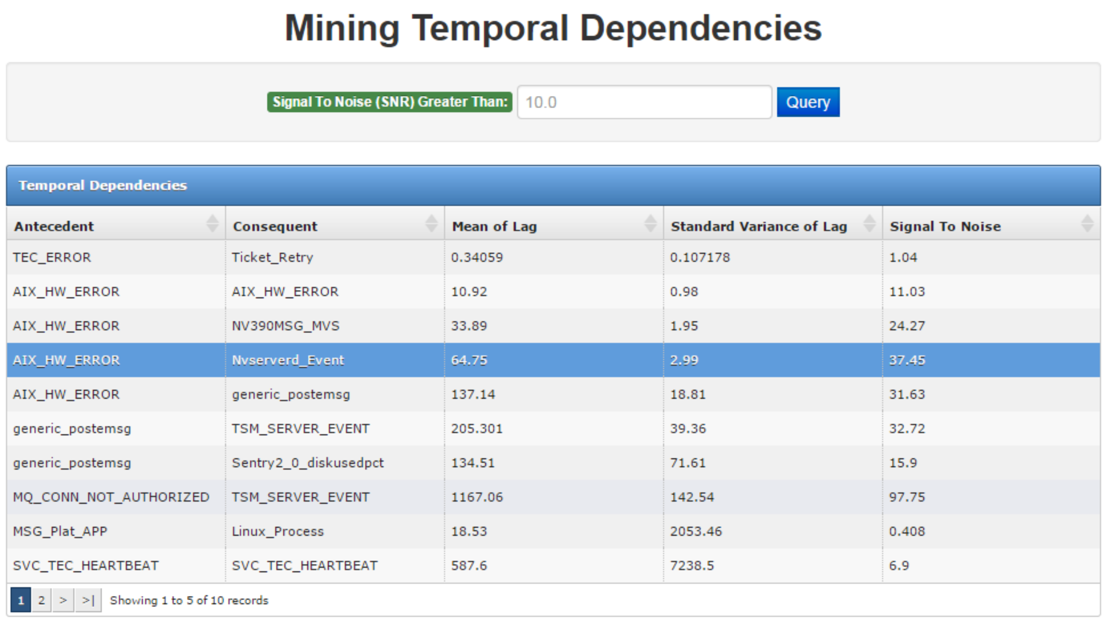
In *dataset*2, the expected time lag between MSG_Plat_APP and $Linux\_Process$ is 18.53 seconds. However, the variance of the time lags is quite large relative to the expected time lag with $SNR = 0.4$. It leads to a weak confidence in temporal dependency between these two events because the discovered time lags get involved in too much noise. In practice, MSG_Plat_APP is a periodic event which is the heartbeat signal sent by the applications. However, the event $Linux\_Process$ is related to the different processes running on the Linux. So it is reasonable to assume a weak dependency between them.

The event $SVC\_TEC\_HEARTBEAT$ is used to record the heartbeat signal for reporting the status of service instantly. The temporal dependency discovered from the *dataset*2 shows that $SVC\_TEC\_HEARTBEAT$ is a periodic event with an expected period of 10 minutes. Although the variance seems large, the standard deviation is relatively small compared with the expected period μ. Therefore, it still strongly indicates the periodic temporal dependency.

4.5.4 Time Lag Discovery Comparison

This section, the temporal patterns discovered from the real data are used for the time lag discovery comparison among some existing temporal pattern mining methods.

In [LM04], the inter-arrival pattern, known as TPattern as well, can also be employed to find the time lag between events such as $TEC\_Error \rightarrow_{[t-\delta,t+\delta]} Ticket\_Retry$ where t and δ is very small. However, it fails to find the temporal pattern such as $MQ\_CONN\_NOT\_AUTHORIZED \rightarrow_L TSM\_SERVER\_EVENT$ with a large


Mining Temporal Dependencies

Signal To Noise (SNR) Greater Than: 10.0 Query

Temporal Dependencies

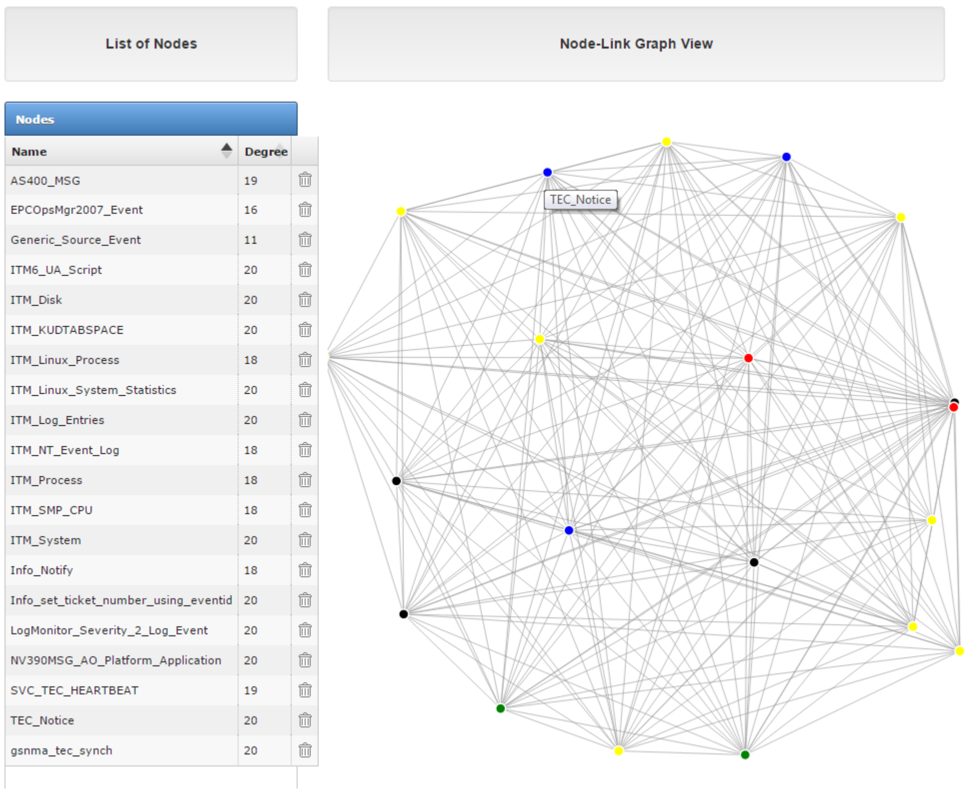
Antecedent	Consequent	Mean of Lag	Standard Variance of Lag	Signal To Noise
TEC_ERROR	Ticket_Retry	0.34059	0.107178	1.04
AIX_HW_ERROR	AIX_HW_ERROR	10.92	0.98	11.03
AIX_HW_ERROR	NV390MSG_MVS	33.89	1.95	24.27
AIX_HW_ERROR	Nvserverd_Event	64.75	2.99	37.45
AIX_HW_ERROR	generic_postemsg	137.14	18.81	31.63
generic_postemsg	TSM_SERVER_EVENT	205.301	39.36	32.72
generic_postemsg	Sentry2_0_diskusedpct	134.51	71.61	15.9
MQ_CONN_NOT_AUTHORIZED	TSM_SERVER_EVENT	1167.06	142.54	97.75
MSG_Plat_APP	Linux_Process	18.53	2053.46	0.408
SVC_TEC_HEARTBEAT	SVC_TEC_HEARTBEAT	587.6	7238.5	6.9

1 2 > >| Showing 1 to 5 of 10 records

Figure 4.13: Temporal dependencies is discovered by setting the SNR threshold and are displayed in a table. The number of temporal dependencies depends on the SNR setting.

expected time lag about of 20 minutes. The reason is that inter-arrival pattern is discovered by only considering the inter-arrival time lag, and the inter-arrival time lags are exactly the small time lags.

In [TLS12], Algorithm *STScan* based on the support and the χ^2 test is proposed to find the interleaved time lags between events. Algorithm *STScan* can find the temporal pattern such as $AIX\_HW\_Error \rightarrow_{[25,25]} AIX\_HW\_Error$ and $AIX\_HW\_Error \rightarrow_{[8,9]} AIX\_HW\_Error$ by setting the support threshold and the confidence level of χ^2 test. In our algorithm, we describe temporal patterns through expected time lag and its variance.



Name	Degree
AS400_MSG	19
EPCOpsMgr2007_Event	16
Generic_Source_Event	11
ITM6_UA_Script	20
ITM_Disk	20
ITM_KUDTABSPACE	20
ITM_Linux_Process	18
ITM_Linux_System_Statistics	20
ITM_Log_Entries	20
ITM_NT_Event_Log	18
ITM_Process	18
ITM_SMP_CPU	18
ITM_System	20
Info_Notify	18
Info_set_ticket_number_using_eventid	20
LogMonitor_Severity_2_Log_Event	20
NV390MSG_AO_Platform_Application	20
SVC_TEC_HEARTBEAT	19
TEC_Notice	20
gsnma_tec_synch	20

Figure 4.14: Temporal dependencies are shown in a graph where each node denotes an event and an edge between two nodes indicates the corresponding two events are dependent.

4.6 System Demonstration

With the help of the time lag mining method, TDMS is able to extract temporal dependencies between events. The representation of temporal dependencies acts an essential part to the users in pattern interpretation.

Several user interfaces for TDMS are demonstrated in Figure 4.13 and Figure 4.14, which are generated by running the appLagEM algorithm over two real data sets from some IT outsourcing centers by IBM Tivoli monitoring system. In Figure 4.13, the discovered temporal dependencies are displayed in a table, where each row corresponds to a temporal dependency rule. From each row, it can tell both the expected value and variance of the time lag between two dependent events. Text box at the top of Figure 4.13 allows to provide SNR, which is able to measure the strength of the temporal dependency.

The temporal dependency graph is constructed in Figure 4.14 according to the dependency rules. In the temporal dependency graph, the nodes represent the events, and the edge between two events indicates an existing temporal dependency. The number of edges in the graph decreases with the growth of SNR's threshold. For the convenience of exploration, all related events are provided in the table on the left.

4.7 Summary

In this chapter, we propose a novel parametric model to discover the distribution of interleaved time lags of the fluctuating events based on an EM-based algorithm. In order to find the distribution of time lag for a large scale event set, a near linear approximation algorithm is proposed. Extensive experiments conducted on both synthetic and real data show its efficiency and effectiveness.

CHAPTER 5

TEMPORAL DEPENDENCY INFERENCE FROM SYSTEM STATISTICS

Large-scale time series data are prevalent across diverse application domains including system management, biomedical informatics, social networks, finance, etc. Temporal dependency discovery performs an essential part to identify the hidden interactions among the observed time series and helps to gain more insight into the behavior of the applications. However, the time-varying sparsity of the interactions among time series often poses a big challenge to temporal dependency discovery in practice.

This chapter formulates the temporal dependency problem with a novel Bayesian model allowing for both the sparsity and evolution of the hidden interactions among the observed time series. Taking advantage of the Bayesian modeling, an online inference method is proposed for time-varying temporal dependency discovery. Extensive empirical studies on both the synthetic and real application time series data are conducted to demonstrate the effectiveness and the efficiency of the proposed method.

5.1 Introduction

Large-scale multivariate time series data are prevalent across diverse application domains including system management, biomedical informatics, social networks, finance, etc. Temporal dependency discovery from multivariate time series has been recognized as one of the key tasks in time series analysis. Taking system management as an example, the time series data (e.g., CPU utilization, memory usage) are collected by monitoring the internal components of a large-scale distributed information system, where a great variety of involved components work together in a highly complex and coordinated manner. Temporal dependency discovered from the monitoring time series reveals important dependency relationships among components and has established its significance in system anomaly detection [JZXL14], root cause analysis for system faults [ZTL$^+$14], etc.

Mining temporal dependency structure among time series has been extensively studied in the past decades. The inference of temporal dependencies can be broadly categorized into two different frameworks: dynamic Bayesian Network [Mur02][JYG$^+$03] and Granger Causality [Gra69][Gra80][ALA07]. An extensive comparison study between these two types of frameworks is presented in [ZF09]. The Granger Causality framework is famous for its simplicity, robustness and extendability, and becomes increasingly popular in practice [CBL14]. Taking these advantages into account, this chapter mainly focuses on the the Granger Causality framework.

The intuitive idea of Granger Causality is that if the time series A Granger causes the time series B, the future value prediction of B can be improved by giving the value of A. The prediction is typically attained by inferring the distribution of time series. Since modeling the distribution for multivariate time series is extremely difficult while linear regression model is a simple and robust approach, regression model has evolved to be one of the principal approaches for Granger Causality. Specifically, to predict the future value of B, one regression model built only on the past values of B should be statistically significantly less accurate than the regression model inferred by giving the past values of both A and B.

Based on the regression model, two major approaches have been developed to discover the Granger Causal relationship for multivariate time series. The first approach employs the statistical significance test to identify the possible interactions among time series, where the nonzero coefficients of the regression model have been verified by hypothesis test. The second method, named Lasso-Granger, determines the Granger Causality from the time series by inferring the regression model with

Lasso regularization. The main idea of Lasso-Granger is to impose a L_1 regularization penalty on the regression coefficients, so that it can effectively identify the sparse Granger Causality especially in high dimensions. It has been shown that both two approaches are consistent in low dimensions, while only Lasso-Granger is consistent in high dimensions [BL13]. Our work is mainly based on the Lasso-Granger approach.

Most existing works related to Lasso-Granger method have been developed for Granger Causality inference by assuming that the latent causal relationships for multivariate time series are fixed yet unknown. However, this assumption rarely holds in practice, since real-world problems often involve underlying processes that are dynamically evolving over time. A scenario of system management, shown in Figure 5.1, is taken as an example. In this example, multiple instances of memory intensive applications are running on a server. At the early stage, the memory of this server is sufficient for supporting running application instances. However, if the number of application instances keeps increasing and the required memory exceeds the capacity of the server, then the server has to take advantage of its virtual memory (the virtual memory is built on the disk storage) to support the running application instances. As a result, an dynamic dependency relationship exists between the number of running application instances and the disk I/O (the number of bytes read from or written to the disk): at the beginning, no obvious relationship occurs between them, while strong relationship is indicated after the number of running application instances increases beyond a threshold related to the memory capacity. It turns out to be critical if the dynamically changing behaviors of the temporal dependency for time series can be identified instantly.

In this chapter, to capture the dynamical change of casual relationships among the time series, we propose a time-varying temporal dependency model based on Lasso-Granger Casuality and develop effective online inference algorithms using particle

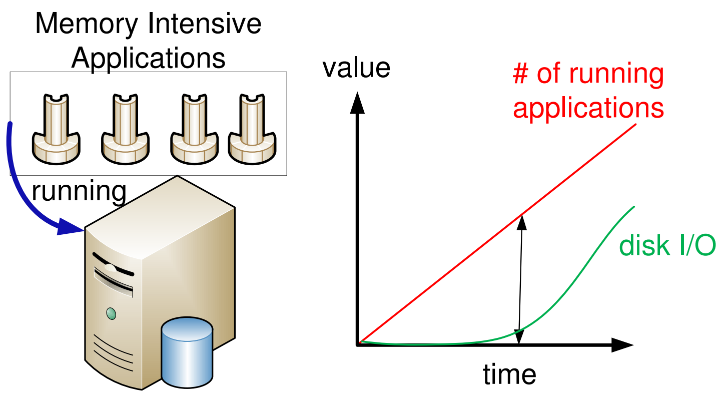


Figure 5.1: The correlation between the number of memory intensive applications and the disk I/O changes dynamically over time in the system management.

learning. The dynamical change behaviors of the temporal dependency is explicitly modeled as a set of random walk particles. The fully adaptive inference strategy of particle learning allows our model to effectively capture the varying dependency and learn the latent parameters simultaneously. We conduct empirical studies on both synthetic and real dataset. The experimental result demonstrate the effectiveness of our proposed approach.

The remainder of this chapter is organized as follows. We formulate the problem for identifying time-varying temporal dependency in Section 5.2. The solution based on particle learning for online model inference is presented in Section 5.3. Extensive empirical evaluation results are reported in Section 5.4. Finally, we conclude our work and the future work in Section 5.5.

5.2 Problem Formulation

In this section, we formally define the Granger Causality problem from a Bayesian perspective first, and then model the time-varying temporal dependency problem. Some important notations mentioned in this chapter are summarized in Table 5.1.

Table 5.1: Important Notations

Notation	**Description**
$\mathbf{Y}$	a set of time series.
K	the number of time series in $\mathbf{Y}$.
T	the length of time series.
$\mathbf{y}_i$	the i^{th} time series.
$\mathbf{y}_{j,t}$	the value of j^{th} time series at time t.
$\mathbf{y}_{\boldsymbol{\cdot},t}$	a column vector containing the values of all time series at time t.
$\mathbf{x}_t$	a column vector built from all time series with time lag L at time t.
$\mathcal{P}_{j,t}$	the set of particles for predicting $\mathbf{y}_{j,t}$ at time t and $\mathcal{P}_{j,t}^{(i)}$ is the i^{th} particle of $\mathcal{P}_{j,t}$.
$\mathbf{W}^l$	the coefficient matrix for time lag l in VAR model.
$\mathbf{w}_j$	the coefficient vector used to predict j^{th} time series value in Bayesian Lasso model.
$\mathbf{w}_{j,t}$	the coefficient vector used to predict j^{th} time series value at time t in time-varying Bayesian Lasso model.
$\mathbf{c}_{\mathbf{w}_j}$	the constant part of $\mathbf{w}_{j,t}$.
$\delta_{\mathbf{w}_{j,t}}$	the drifting part of $\mathbf{w}_{j,t}$.
$\eta_{j,t}$	the standard Gaussian random walk at time t, given $\eta_{j,t-1}$.
θ_j	the scale parameters used to compute $\delta_{\mathbf{w}_{j,t}}$.
σ_j^2	the variance of value prediction for the j^{th} time series.
α, β	the hyper parameters determine the distribution of σ_j^2.
$\mu_{\mathbf{w}}$	the hyper parameters determine the distribution of $\mathbf{w}_j$ in Bayesian Lasso model.
$\mu_{\mathbf{c}}$	the hyper parameters determine the distribution of $\mathbf{c}_{\mathbf{w}_j}$.
μ_θ	the hyper parameters determine the distribution of θ_j.
γ_p^2	the augmented random variable for $\mathbf{w}_j$, with λ.
$\gamma_{c,p}^2$	the augmented random variable for $\mathbf{c}_{\mathbf{w}_j}$, with λ_1.
$\gamma_{\theta,p}^2$	the augmented random variable for θ_j, with λ_2.
λ, λ_1, λ_2	the Lasso penalty parameters for $\mathbf{w}_j$, $\mathbf{c}_{\mathbf{w}_j}$ and θ_j, respectively.

5.2.1 Basic Concepts and Terminologies

Let $\mathbf{Y}$ be a set of time series, denoted as $\mathbf{Y} = \{\mathbf{y}_i | 1 \leq i \leq K\}$, where K is the number of time series in $\mathbf{Y}$ and $\mathbf{y}_i$ is the i^{th} time series. Assume $\mathbf{y}_{i,t} \in R$ to be the value of the i^{th} time series at time t, where $0 \leq t \leq T$. The time series $\mathbf{y}_j$ is supposed to be caused by another time series $\mathbf{y}_i$ in terms of Granger Causality, denoted as $\mathbf{y}_i \rightarrow_g \mathbf{y}_j$, if and only if the regression for $\mathbf{y}_j$ using the past values of both $\mathbf{y}_j$ and $\mathbf{y}_i$ gains statistically significant improvement in terms of accuracy comparing with doing so with past values of $\mathbf{y}_j$ only. The Granger causal relationship among the set of time series $\mathbf{Y}$ is formulated as a directed graph G, where each vertex of G corresponds to a time series, and an edge exists directed from $\mathbf{y}_i$ to $\mathbf{y}_j$ if $\mathbf{y}_i \rightarrow_g \mathbf{y}_j$.

In practice, the inference of Ganger causality is usually achieved by fitting the time series data $\mathbf{Y}$ with a Vector Auto-Regression (VAR) model. Let $\mathbf{y}_{\cdot,t} = (\mathbf{y}_{1,t}, ..., \mathbf{y}_{K,t})^\intercal$, a column vector containing the values of K time series at time t. Given the maximum time lag L, the VAR model is expressed as follows,

$$\mathbf{y}_{\cdot,t} = \sum_{l=1}^{L} (\mathbf{W}^l)^\intercal \mathbf{y}_{\cdot,t-l} + \epsilon, \tag{5.1}$$

where $\mathbf{W}^l$ is $K \times K$ coefficient matrix for time lag l, and ϵ is a $K \times 1$ vector, describing the random noise. The nonzero value of $\mathbf{W}^l_{ij}$ indicates $\mathbf{y}_i \rightarrow_g \mathbf{y}_j$. A statistics test [ALA07] method is applied to determine the nonzero values in $\mathbf{W}^l$, based on the VAR model shown in Equation 5.1. However, the combinational explosion for the statistics test on time series pairs brings about its inefficiency for Granger causality inference, especially analyzing time series data with high dimension.

Lasso-Granger provides a more efficient and consistent way to infer the Granger causal relation among time series, where L_1 regularization is imposed for addressing sparsity issue in high dimensional time series data [ALA07]. Specifically, the coefficient matrix $\mathbf{W}^l$ is obtained by minimizing the following objective function,

$$\min_{\{\mathbf{W}^l\}} \sum_{t=L+1}^{T} \| \mathbf{y}_{\cdot,t} - \sum_{l=1}^{L} (\mathbf{W}^l)^\intercal \mathbf{y}_{\cdot,t-l} \|_2^2 + \lambda \sum_{l=1}^{L} \| \mathbf{W}^l \|_1, \tag{5.2}$$

where λ is the penalty parameter, which determines the sparsity of the coefficient matrix $\mathbf{W}^l$.

In Equation 5.2, Lasso-Granger provides regression for K variables, where each variable is expressed as a linear function of its own past values and past values of all

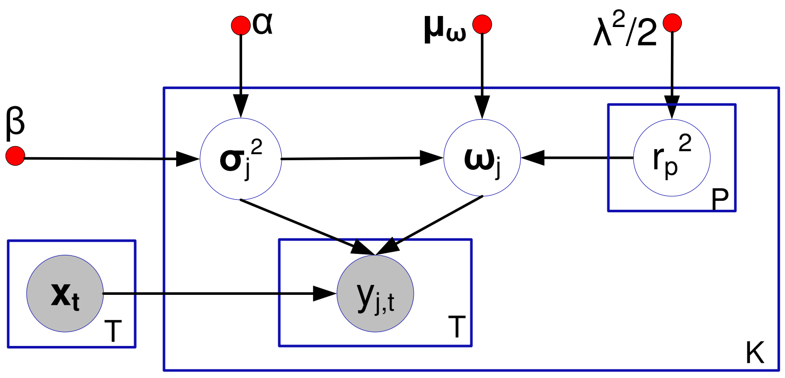


Figure 5.2: Bayesian Lasso model is expressed in Graphical representations for Granger Causality. Random variable is denoted as a circle. The circle with gray color filled means the corresponding random variable is observed. Red dot represents a hyper parameter.

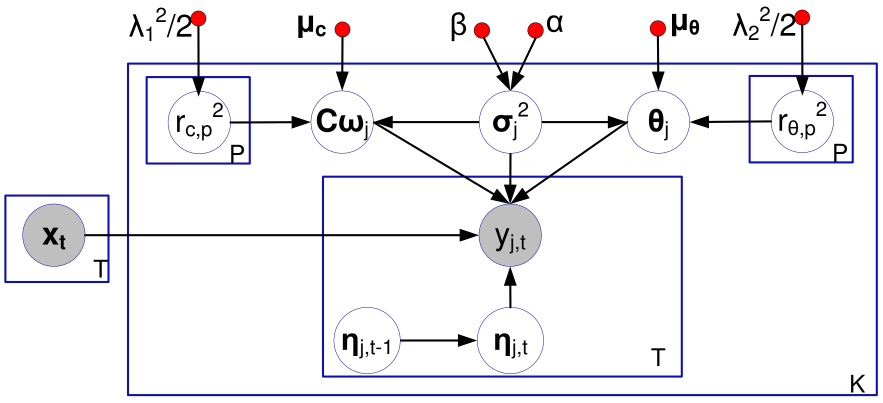


Figure 5.3: Time-varying Bayesian Lasso model is expressed in Graphical representations for Granger Causality. Random variable is denoted as a circle. The circle with gray color filled means the corresponding random variable is observed. Red dot represents a hyper parameter.

other variables with L_1 regularization. To be simplified, we focus on the regression for one arbitrarily given variable $\mathbf{y}_j$, and the regression of other variables can be derived in a similar way.

Let $\mathbf{x}_t = vec([\mathbf{y}_{\cdot,t-1}, \mathbf{y}_{\cdot,t-2}, ..., \mathbf{y}_{\cdot,t-L}])$, where $vec(\cdot)$ is an operator to convert a matrix into a vector by stacking column vectors. The Lasso regression for the variable

$\mathbf{y}_j$ is expressed as follows,

$$\min_{\mathbf{w}_j} \sum_{t=L+1}^{T} (\mathbf{y}_{j,t} - \mathbf{w}_j^{\intercal}\mathbf{x}_t)^2 + \lambda \parallel \mathbf{w}_j \parallel_1, \tag{5.3}$$

where $\mathbf{w}_j$ is the coefficient vector of the regression for the variable $\mathbf{y}_j$. Assuming $P = K * L$, both $\mathbf{x}_t$ and $\mathbf{w}_j$ are column vectors with the dimension $P \times 1$. However, Equation 5.3 tends to be addressed as an optimization problem, and it is not suitable for online inference.

5.2.2 Bayesian Modeling

In order to track the temporal dependencies among time series instantly, the problem described in Equation 5.3 is reformulated from a Bayesian perspective. Bayesian method provides a natural and principled way of combining prior information with data, within a solid decision theoretical framework. The past information about parameters can be incorporated and formed as prior knowledge for future analysis. When new observations become available at current time t, the previous posterior distribution of parameters at time $t-1$ can be used as a prior for current parameter inference. The parameter estimate for linear regression with Lasso penalty can be interpreted as a Bayesian posterior mode estimate when the priors on the regression parameters are independent Laplace distributions [PC08]. The regression for $\mathbf{y}_{j,t}$ is implemented by a linear combination of $\mathbf{x}_t$ with coefficient vector $\mathbf{w}_j$. From Bayesian perspective, given the coefficient vector (i.e., $\mathbf{w}_j$) and the variance of random observation noise (i.e., σ_j^2), it is assumed that $\mathbf{y}_{j,t}$ follows a Gaussian distribution as below,

$$\mathbf{y}_{j,t}|\mathbf{w}_j, \sigma_j^2 \sim \mathcal{N}(\mathbf{w}_j^{\intercal}\mathbf{x}_t, \sigma_j^2). \tag{5.4}$$

In this setting, a graphical representation for Bayesian Lasso model is illustrated in Figure 5.2, where the predicted value $\mathbf{y}_{j,t}$ depends on random variable $\mathbf{x}_t$, $\mathbf{w}_j$ and σ_j^2.

To obtain a Bayesian model equivalent to the Lasso regression in Equation 5.3 and simplify the computation, the conjugate prior distributions for all the coefficients in $\mathbf{w}_j$ are assumed as the independent Laplace distributions. Therefore,

$$\pi(\mathbf{w}_j|\sigma_j^2) = \prod_{p=1}^{P} \frac{\lambda}{2\sqrt{\sigma^2}} e^{-\lambda|\mathbf{w}_{j,p}|/\sqrt{\sigma^2}}, \tag{5.5}$$

where $\pi(\mathbf{.})$ denotes the probability density function. The distribution in Equation 5.5 can be equivalently expressed as a scale mixture of normals with an exponential mixing density. The augmented latent variables $\gamma_1^2, ..., \gamma_P^2$, following independent exponential distributions, are introduced to build the mixture of normals. The full Bayesian Lasso model is developed in the following hierarchical representation.

$$\begin{aligned}
&\mathbf{w}_j|\sigma_j^2, \gamma_1^2, ..., \gamma_P^2 \sim \mathcal{N}(\mu_{\mathbf{w}}, \sigma_j^2 \mathbf{R}_{\mathbf{w_j}}), \\
&\sigma_j^2 \sim \mathcal{IG}(\alpha, \beta), \\
&\gamma_p^2 \sim \mathit{Exp}(\lambda^2/2), \quad 1 \leq p \leq P,
\end{aligned} \tag{5.6}$$

where $\mathbf{R}_{\mathbf{w_j}} = \mathtt{diag}(\gamma_1^2, ..., \gamma_P^2)$. The prior of σ_j^2 follows Inverse Gamma (abbr., $\mathcal{IG}$) distribution with hyper parameters α and β. The prior of γ_p^2 is given by the exponential distribution (denoted as *Exp*) with the hyper parameter $\lambda^2/2$, where λ is the Lasso regularization parameter defined in Equation 5.3. Given σ_j^2 and $\gamma_1^2, ..., \gamma_P^2$, the prior of the coefficient vector $\mathbf{w}_j$ follows a Gaussian distribution with $\mu_{\mathbf{w}}$ and $\sigma_j^2 \mathbf{R}_{\mathbf{w_j}}$ as the mean and the variance, respectively. Typically, $\mu_{\mathbf{w}}$ is set to be $\mathbf{0}$.

The full hierarchical representation in Equation 5.6 can be reduced to the joint distribution of independent Laplace priors in Equation 5.5 after integrating out all the augmented latent variables $\gamma_1^2, ..., \gamma_P^2$. With the help of the Bayesian Lasso model, the temporal dependency in terms of Granger Causality can be determined by inferring the posterior distribution of $\mathbf{w}_j$ instantly.

5.2.3 Dynamic Causal Relationship Modeling

In real practice, the underlying causal relationship among time series tends to evolve over time. As illustrated in Figure 5.4, From the time $t-1$ to t, the dynamic changes of the causal relationship among time series consist of three types: new dependency occurring, dependency fading away, and the strength of dependency varying.

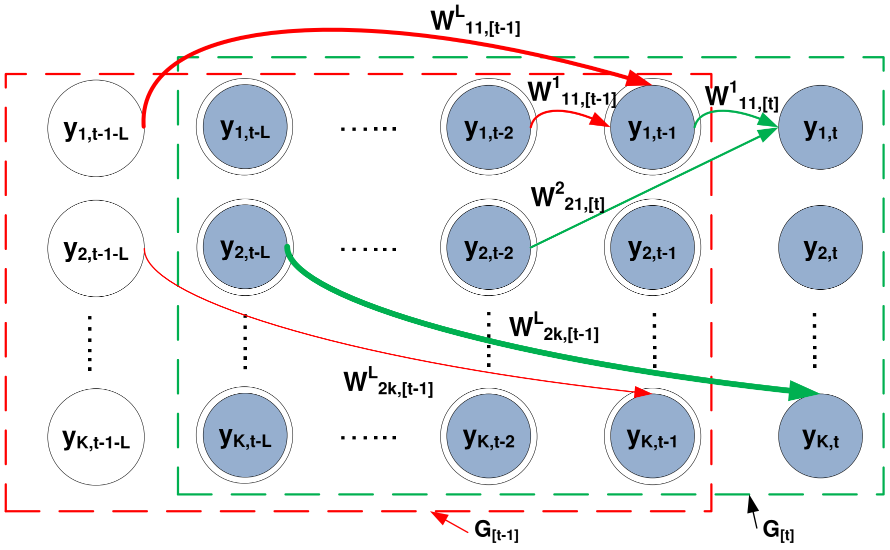


Figure 5.4: L is the maximum time lag for VAR model. Temporal dependency among time series changes from $G[t-1]$ at time $t-1$ to $G[t]$ at time t. The nonzero coefficients are indicated by the directed edges. Red lines is used to denote the temporal dependencies in $G[t-1]$, while the green lines represent the temporal dependencies in $G[t]$. The thicker lines mean stronger dependencies existing.

As shown in Equation 5.3, the value prediction for $\mathbf{y}_j$ at time t is conducted by a linear combination of its own past values and the past values of other variables, using coefficient vector $\mathbf{w}_j$ with L_1 regularization penalty. Each element in the coefficient vector $\mathbf{w}_j$ indicates the contribution of the past value of the corresponding variable for predicting $\mathbf{y}_{j,t}$. The aforementioned model is based on the assumption that $\mathbf{w}_j$ is unknown but fixed, which does not work well with the scenario where the temporal dependency dynamically changes over time. To account for the dynamics, our goal is to come up with a model having the capability of capturing the drift of $\mathbf{w}_j$ over time so as to track the time-varying temporal dependency among the time series instantly. Let $\mathbf{w}_{j,t}$ denote the coefficient vector for predicting $\mathbf{y}_{j,t}$ at time t. Taking the drift of $\mathbf{w}_j$ into account, $\mathbf{w}_{j,t}$ is formulated as follows:

$$\mathbf{w}_{j,t} = \mathbf{c}_{\mathbf{w}_j} + \delta_{\mathbf{w}_{j,t}}, \tag{5.7}$$

where $\mathbf{w}_{j,t}$ is decomposed into two components including both the stationary component $\mathbf{c}_{\mathbf{w}_j}$ and the drift component $\delta_{\mathbf{w}_{j,t}}$. Both components are P-dimensional vectors. Similar to modeling $\mathbf{w}_j$ in Figure 5.2, a conjugate prior distribution below is assumed to generate the stationary component $\mathbf{c}_{\mathbf{w}_j}$.

$$\mathbf{c}_{\mathbf{w}_j} \sim \mathcal{N}(\mu_c, \sigma_j^2 \mathbf{R}_{\mathbf{c_j}}), \tag{5.8}$$

where μ_c is the hyper parameter, and $\mathbf{R}_{\mathbf{c_j}} = \mathtt{diag}(\gamma_{c,1}^2, ..., \gamma_{c,P}^2)$. The latent variables $\gamma_{c,1}^2, ..., \gamma_{c,P}^2$ follow independent exponential distributions ruled by the hyper parameter $\lambda_1^2/2$, as shown in Figure 5.3.

However, it's not straightforward to model the drift component with a single function due to the diverse changing behaviors of the regression coefficients. First, some coefficients change frequently, while some coefficients keep relatively stable. Moreover, the coefficients for different variables can change with diverse scales. To simplify the inference , we assume that each element of $\delta_{\mathbf{w}_{j,t}}$ drifts independently. Due to the uncertainty of drifting, we formulate $\delta_{\mathbf{w}_{j,t}}$ by combining a standard Gaussian random walk $\eta_{j,t}$ and a scale variable θ_j using the following Equation:

$$\delta_{\mathbf{w}_{j,t}} = \theta_j \odot \eta_{j,t}, \tag{5.9}$$

where $\eta_{j,t} \in \mathcal{R}^P$ is the drift value at time t caused by the standard random walk and $\theta_j \in \mathcal{R}^P$ contains the changing scales for all the elements of $\delta_{\mathbf{w}_{j,t}}$. The operator $\odot$ is used to denote the element-wise product. The standard Gaussian random walk is defined with a Markov process as shown in Equation 5.10.

$$\eta_{j,t} = \eta_{j,t-1} + \mathbf{v}, \tag{5.10}$$

where $\mathbf{v}$ is a standard Gaussian random variable defined by $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathcal{I}_P)$, and $\mathcal{I}_P$ is a $P \times P$-dimensional identity matrix. It is equivalent that $\eta_{j,t}$ is sampled from the Gaussian distribution

$$\eta_{j,t} \sim \mathcal{N}(\eta_{j,t-1}, \mathcal{I}_P). \tag{5.11}$$

Similarly, the scale random variable θ_j is generated with a conjugate prior distribution

$$\theta_j \sim \mathcal{N}(\mu_\theta, \sigma_j^2 \mathbf{R}_{\theta_\mathrm{j}}), \tag{5.12}$$

where μ_θ is predefined hyper parameter, and $\mathbf{R}_{\theta_\mathrm{j}} = \mathtt{diag}(\gamma_{\theta,1}^2, ..., \gamma_{\theta,P}^2)$. The latent variables $\gamma_{\theta,1}^2, ..., \gamma_{\theta,P}^2$, following the independent exponential distributions governed by the hyper parameter $\lambda_2^2/2$, are used to construct $\mathbf{R}_{\theta_\mathrm{j}}$. The random variable σ_j^2 of the time-varying Bayesian Lasso model in Figure 5.3 is drawn from the Inverse Gamma distribution, which is the same as the one described in Equation 5.6.

Combining Equation 5.7 and Equation 5.9, we obtain:

$$\mathbf{w}_{j,t} = \mathbf{c}_{\mathbf{w}_j} + \theta_j \odot \eta_{j,t}, \tag{5.13}$$

Accordingly, the value $\mathbf{x}_j^t$ is modeled to be drawn from the following a Gaussian distribution as below,

$$\mathbf{y}_{j,t} | \mathbf{c}_{\mathbf{w}_j}, \theta_j, \eta_{j,t}, \sigma_j^2 \sim \mathcal{N}((\mathbf{c}_{\mathbf{w}_j} + \theta_j \odot \eta_{j,t})^\intercal \mathbf{x}_t, \sigma_j^2). \tag{5.14}$$

The time-varying Bayesian Lasso model is presented with a graphical model representation in Figure 5.3. Compared with the model in Figure 5.2, a standard Gaussian random walk $\eta_{j,t}$ and the corresponding scale θ_j for j^{th} time series are introduced in the new model. The new model explicitly formulates the coefficients in Lasso regression, considering the time-varying temporal dependency in real-world application. From the new model, each element value of $\mathbf{c}_{\mathbf{w}_j}$ indicates the contribution of the past values of each variable in predicting the value $\mathbf{y}_{j,t}$, while the element values of θ_j show the drift scales of their contributions to the prediction of $\mathbf{y}_{j,t}$. A large element value of θ_j signifies a great change occurring to the strength of the corresponding causal relationship over time.

Lemma 5.2.1 (Equivalent Optimization) *The time-varying Bayesian Lasso model is equivalent to the optimization problem as follows:*

$$\begin{aligned} \min_{\{\mathbf{w}_{j,t}\}} \sum_{t=L+1}^{T} (\mathbf{y}_{j,t} - (\mathbf{c}_{\mathbf{w}_j} + \theta_j \odot \eta_{j,t})^\intercal \mathbf{x}_t)^2 + \\ \lambda_1 \parallel \mathbf{c}_{\mathbf{w}_j} \parallel_1 + \lambda_2 \parallel \theta_j \parallel_1, \end{aligned} \tag{5.15}$$

where λ_1 and λ_2 are penalty parameters, determining the sparsity of both stationary component and drift component.

Based on the idea of Bayesian Lasso, Lemma 5.2.1 is straightforward. Thus, its proof is not provided.

According to Lemma 5.2.1, λ_1 is set to determine the sparsity of stationary component and λ_2 is used for controlling the variance of drift component. It is difficult to infer the coefficient vectors $\{\mathbf{w}_{j,t}\}$ instantly directly from Equation 5.15, since $\eta_{j,t}$ is the latent variables. We develop our solution to infer the time-varying Bayesian Lasso model from a Bayesian perspective and the solution is presented in the following section.

5.3 Methodology and Solution

In this section, we present the methodology for online inference of the time-varying Bayesian Lasso model.

The posterior distribution inference involves the latent random variables σ_j^2, $\mathbf{c}_{\mathbf{w_j}}$, θ_j, $\mathbf{R}_{\mathbf{c}_j}$, $\mathbf{R}_{\theta_j}$, and $\eta_{j,t}$. According to the graphical model in Figure 5.3, all the latent random variables are grouped into three categories: parameter random variable, augmented random variable and latent state random variable. σ_j^2, $\mathbf{c}_{\mathbf{w}_j}$, θ_j, are parameter random variables since they are assumed to be fixed and unknown, and their values do not depend on the time. $\mathbf{R}_{\mathbf{c}_j}$, $\mathbf{R}_{\theta_j}$ are regarded as augmented random variables where these variables are introduced for equivalent Lasso derivation but their specific values are not very interesting for the problem. Instead, $\eta_{j,t}$ is referred to as a latent state random variable since it is not observable and its value is time dependent according to Equation 5.10. On the other hand, $\mathbf{x}_t$ and $\mathbf{y}_{j,t}$ are referred to as observed random variables.

Our goal is to infer both latent parameters and latent state variables. However, since the inference partially depends on the random walk which generates the latent state variables, we use the sequential sampling based inference strategy that are widely used in sequential monte carlo sampling [DDFG01] [SDdFG13], particle filtering [DKZ+03], and particle learning [CJLP10] to learn the distribution of both parameters and the state random variables.

Since state $\eta_{j,t-1}$ changes over time with a standard Gaussian random walk, it follows a Gaussian distribution after accumulating $t-1$ standard Gaussian random walks. Assume $\eta_{j,t-1} \sim \mathcal{N}(\mu_{\eta_j}, \mathbf{\Sigma}_{\eta_j})$, a particle is defined as follows.

Definition 5.3.1 (Particle) *A particle for predicting $\mathbf{y}_{j,t}$ is a container which maintains the current status information for value prediction. The status information comprises of random variables such as σ_j^2, $\mathbf{c}_{\mathbf{w}_j}$, θ_j, $\mathbf{R}_{\mathbf{c}_j}$, $\mathbf{R}_{\theta_j}$, and $\eta_{j,t}$, and the hyper parameters of their corresponding distributions such as α and β, $\mu_{\mathbf{c}}$, μ_θ, λ_1, λ_2, μ_{η_k} and $\mathbf{\Sigma}_{\eta_k}$.*

5.3.1 Re-sample Particles with Weights

At time $t-1$, a fixed-size set of particles are maintained for the value prediction of the j^{th} time series, where the particle set is denoted as $\mathcal{P}_{j,t-1}$ and the number of particles in $\mathcal{P}_{j,t-1}$ is B. Let $\mathcal{P}_{j,t-1}^{(i)}$ be the i^{th} particle in the particle set $\mathcal{P}_{j,t-1}$ at time $t-1$, where $1 \le i \le B$. Each particle $\mathcal{P}_{j,t-1}^{(i)}$ has a weight, denoted as $\rho^{(i)}$, indicating its fitness for the new observed data at time t. Note that $\sum_{i=1}^{B} \rho^{(i)} = 1$. The fitness of each particle $\mathcal{P}_{j,t-1}^{(i)}$ is defined as the likelihood of the observed data $\mathbf{x}_t$ and $\mathbf{y}_{j,t}$. Therefore,

$$\rho^{(i)} \propto P(\mathbf{x}_t, \mathbf{y}_{j,t} | \mathcal{P}_{j,t-1}^{(i)}). \tag{5.16}$$

Further, according to Equation 5.14, the distribution of $\mathbf{y}_{j,t}$ is determined by the random variables $\mathbf{c}_{\mathbf{w}_j}$, θ_j, σ_j^2 and $\eta_{j,t}$.

Therefore, we can compute $\rho^{(i)}$ in proportional to the density value at $\mathbf{y}_{j,t}$. Thus,

$$\begin{aligned}\rho^{(i)} \propto \iint_{\eta_{j,t},\eta_{j,t-1}} & \{\mathcal{N}(\mathbf{y}_{j,t} | (\mathbf{c}_{\mathbf{w}_j} + \theta_j \odot \eta_{j,t})^\intercal \mathbf{x}_t, \sigma_j^2) \\ & \mathcal{N}(\eta_{j,t} | \eta_{j,t-1}, \mathcal{I}_P) \mathcal{N}(\eta_{j,t-1} | \mu_{\eta_j}, \mathbf{\Sigma}_{\eta_j})\} \\ & d\eta_{j,t}\, d\eta_{j,t-1},\end{aligned}$$

where state variables $\eta_{j,t}$ and $\eta_{j,t-1}$ are integrated out due to their change over time, and $\mathbf{c}_{\mathbf{w}_j}$, θ_j, σ_j^2 are from $\mathcal{P}_{j,t-1}^{(i)}$. Then we obtain

$$\rho^{(i)} \propto \mathcal{N}(\mathbf{m}_j, \mathbf{Q}_j), \tag{5.17}$$

where

$$\begin{aligned}\mathbf{m}_j &= (\mathbf{c}_{\mathbf{w}_j} + \theta_j \odot \eta_{j,t})^\intercal \mathbf{x}_t \\ \mathbf{Q}_j &= \sigma_j^2 + (\mathbf{x}_t \odot \theta_j)^\intercal (\mathcal{I}_P + \mathbf{\Sigma}_{\eta_\mathbf{j}})(\mathbf{x}_t \odot \theta_j).\end{aligned} \tag{5.18}$$

Before updating any parameters, a re-sampling process is conducted. We replace the particle set $\mathcal{P}_{j,t-1}$ with a new set $\mathcal{P}_{j,t}$, where $\mathcal{P}_{j,t}$ is generated from $\mathcal{P}_{j,t-1}$ using sampling with replacement based on the weights of particles. Then sequential parameter updating is based on $\mathcal{P}_{j,t}$.

5.3.2 Latent State Inference

At time $t-1$, the sufficient statistics for state $\eta_{j,t-1}$ are the mean (i.e., μ_{η_j}) and the covariance (i.e., $\mathbf{\Sigma}_{\eta_j}$). Provided with the new observation data $\mathbf{x}_t$ and $\mathbf{y}_{j,t}$ at time t, the sufficient statistics for state $\eta_{j,t}$ need to be re-computed. We apply the Kalman filtering [Har90] method to recursively update the sufficient statistics for $\eta_{j,t}$ based on the new observation and the sufficient statistics at time $t-1$. Let μ'_{η_j} and $\mathbf{\Sigma}'_{\eta_j}$ be the new sufficient statistics of state $\eta_{j,t}$ at time t. Then,

$$\begin{aligned} \mu'_{\eta_j} &= \mu_{\eta_j} + \underbrace{\mathbf{G}_j(\mathbf{y}_{j,t} - (\mathbf{c}_{\mathbf{w}_j} + \theta_j \odot \eta_{j,t})^\intercal \mathbf{x}_t))}_{\text{Correction by Kalman Gain}}, \\ \mathbf{\Sigma}'_{\eta_j} &= \mathbf{\Sigma}_{\eta_j} + \mathcal{I}_P - \underbrace{\mathbf{G}_j \mathbf{Q}_j \mathbf{G}_j^\intercal}_{\text{Correction by Kalman Gain}}, \end{aligned} \tag{5.19}$$

where $\mathbf{Q}_j$ is defined in Equation 5.18 and $\mathbf{G}_j$ is Kalman Gain [Har90] defined as

$$\mathbf{G}_j = (\mathcal{I}_P + \mathbf{\Sigma}_{\eta_j})(\mathbf{x}_t \odot \theta_j)\mathbf{Q}_j^{-1}.$$

As shown in Equation 5.19, both μ'_{η_j} and $\mathbf{\Sigma}'_{\eta_j}$ are estimated with a correction using Kalman Gain $\mathbf{G}_j$(i.e., the last term in both two formulas). With the help of the sufficient statistics for the state random variable, $\eta_{j,t}$ can be drawn from the Gaussian distribution

$$\eta_{j,t} \sim \mathcal{N}(\mu'_{\eta_j}, \mathbf{\Sigma}'_{\eta_j}). \tag{5.20}$$

5.3.3 Augmented Variable Inference

The augmented variables $\mathbf{R}_{\mathbf{c}_j}$ and $\mathbf{R}_{\theta_j}$ are diagonal matrices composed of independent random variables $\gamma^2_{c,1}, ..., \gamma^2_{c,P}$ and $\gamma^2_{\theta,1}, ..., \gamma^2_{\theta,P}$, respectively. The independent random variables are drawn from exponential distribution as follows,

$$\begin{aligned} \gamma_{c,p} &\sim Exp(\lambda_1^2/2), \\ \gamma_{\theta,p} &\sim Exp(\lambda_2^2/2), \end{aligned} \tag{5.21}$$

where $1 \leq p \leq P$. At each time stamp, those augmented random variables are sampled independently. Assume $\mathbf{R}_j = \begin{bmatrix} \mathbf{R_{c_j}} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{\theta_\mathbf{j}} \end{bmatrix}$, where $\mathbf{R}_j$ is a $2P \times 2P$ -dimensional matrix.

5.3.4 Parameter Inference

At time $t-1$, the sufficient statistics for the parameter random variables (σ_j^2, $\mathbf{c_{w_j}}$, θ_j) are (α, β, μ_c, μ_θ). Let $\mathbf{z}_t = (\mathbf{x}_t{}^\intercal, (\mathbf{x}_t \odot \eta_\mathbf{j,t})^\intercal)^\intercal$, $\mu_j = (\mu_\mathbf{c}{}^\intercal, \mu_\theta{}^\intercal)^\intercal$, and $\nu_j = (\mathbf{c_{w_j}}{}^\intercal, \theta_\mathbf{j}{}^\intercal)^\intercal$ where $\mathbf{z}_t$, μ_j, and ν_j are $2P$-dimensional vector.

Therefore, the inference of $\mathbf{c_{w_j}}$ and θ_j is equivalent to infer ν_j with its distribution $\nu_j \sim \mathcal{N}(\mu_j, \sigma_j^2 \mathbf{R}_j^{\frac{1}{2}} \mathbf{\Sigma}_j \mathbf{R}_j^{\frac{1}{2}})$, where $\mathbf{\Sigma}_j$ is initialized with an identity matrix time 0. Assume $\mathbf{\Sigma'}_j$, $\mu'_\mathbf{j}$, α', and β' be the sufficient statistics at time t which are updated based on the sufficient statistics at time $t-1$ and the new observation data. The sufficient statistics for parameters are updated as follows:

$$\begin{aligned}
\mathbf{\Sigma'}_j &= (\mathbf{\Sigma}_j^{-1} + \mathbf{R}_j^{\frac{1}{2}} \mathbf{z}_t \mathbf{z}_t^\intercal \mathbf{R}_j^{\frac{1}{2}})^{-1}, \\
\mu'_\mathbf{j} &= \mathbf{R}_j^{\frac{1}{2}} \mathbf{\Sigma'}_j \mathbf{R}_j^{\frac{1}{2}} \mathbf{z}_t \mathbf{y}_{j,t} + \mathbf{R}_j^{\frac{1}{2}} \mathbf{\Sigma'}_j \mathbf{\Sigma}_j \mathbf{R}_j^{\frac{1}{2}} \mu_j), \\
\alpha' &= \alpha + \frac{1}{2}, \\
\beta' &= \beta + \frac{1}{2}(\mu_j^\intercal \mathbf{R}_j^{-\frac{1}{2}} \mathbf{\Sigma}_j^{-1} \mathbf{R}_j^{-\frac{1}{2}} \mu_j + \mathbf{y}_{j,t}^2 - \mu_j'^\intercal \mathbf{R}_j^{-\frac{1}{2}} \mathbf{\Sigma'}_j^{-1} \mathbf{R}_j^{-\frac{1}{2}} \mu'_j).
\end{aligned} \tag{5.22}$$

At time t, the sampling process for σ_j^2 and ν_j is summarized as follows:

$$\begin{aligned}
\sigma_j^2 &\sim \mathcal{IG}(\alpha', \beta'), \\
\nu_j &\sim \mathcal{N}(\mu'_j, \sigma_j^2 \mathbf{R}_j^{\frac{1}{2}} \mathbf{\Sigma'}_j \mathbf{R}_j^{\frac{1}{2}}).
\end{aligned} \tag{5.23}$$

5.3.5 Algorithm

Putting all the aforementioned things together, an algorithm based on the proposed time-varying Bayesian Lasso model is provided below.

Online inference for time-varying Bayesian Lasso model starts with MAIN procedure, as presented in Algorithm 6. The parameters B, L, α, β, λ_1 and λ_2 are given as the input of MAIN procedure. The initialization is executed from line 2 to line 6. As new observation $\mathbf{y}_{\cdot,t}$ arrives at time t, $\mathbf{x}_t$ is built using the time lag, then $\mathbf{w}_{j,t}$ is inferred by calling UPDATE procedure. Especially in the UPDATE procedure, we use the *resample-propagate* strategy in particle learning [CJLP10] rather than the *propagate-resample* strategy in particle filtering [DKZ$^+$03]. With the *resample-propagate* strategy, the particles are re-sampled by taking $\rho^{(i)}$ as the i^{th} particle's weight, where the $\rho^{(i)}$ indicates the occurring probability of the observation at time t given the particle at time $t-1$. The *resample-propagate* strategy is considered as an optimal and fully adapted strategy, avoiding an importance sampling step.

5.4 Empirical Study

With the purpose of demonstrating the performance of the proposed algorithm, we conduct the experiments over both synthetic and real data sets, and illustrate a real case study from the system management. Before diving into the discussion of the evaluation in detail, we first outline the general implementation of the baseline algorithms for comparison, then verify the proposed algorithm using every data set one by one. The evaluation on each data set is started with a brief description of the data and the corresponding evaluation methods, and followed by the presentation of the comparative experimental results between the proposed algorithm and the baseline algorithms.

Algorithm 6 The algorithm for time-varying Bayesian Lasso model

```
1: procedure MAIN($B$, $L$, $\alpha$, $\beta$, $\lambda_1$, $\lambda_2$)    ▷ main entry
2:     Initialize $\mu_c = \mathbf{0}$, $\mu_\theta = \mathbf{0}$.
3:     for $j \leftarrow 1, K$ do
4:         Initialize regression for $\mathbf{y}_j$ with $B$ particles.
5:         Initialize $\Sigma_j$ with identity matrix.
6:     end for
7:     for $t \leftarrow 1, T$ do
8:         Get $\mathbf{x}_t$ using time lag $L$.
9:         for $j \leftarrow 1, K$ do
10:            UPDATE($\mathbf{x}_t$, $y_{j,t}$).
11:            Output $\mathbf{w}_{j,t}$ according to Eq. 5.13.
12:        end for
13:    end for
14: end procedure

15: procedure UPDATE($\mathbf{x}_t$, $y_{j,t}$)    ▷ update the inference.
16:     for $i \leftarrow 1, B$ do    ▷ Compute weights for each particle.
17:         Compute weight $\rho^{(i)}$ of particle $\mathcal{P}^{(i)}_{j,t-1}$ by Eq. 5.17.
18:     end for
19:     Re-sample $\mathcal{P}_{j,t}$ from $\mathcal{P}_{j,t-1}$ according to $\rho^{(i)}$s.
20:     for $i \leftarrow 1, B$ do    ▷ Update statistics for each particle.
21:         Update the sufficient statistics for $\eta_{j,t}$ by Eq. 5.19.
22:         Sample $\eta_{j,t}$ according to Eq. 5.20.
23:         Construct augmented variables $\mathbf{R}_j$ with Eq. 5.21.
24:         Update the statistics for $\sigma_j^2$, $\mathbf{c}_{\mathbf{w}_j}$, $\theta_j$ by Eq. 5.22.
25:         Sample $\sigma_j^2$, $\mathbf{c}_{\mathbf{w}_j}$, $\theta_j$ according to Eq. 5.23.
26:     end for
27: end procedure
```

5.4.1 Baseline Algorithms

In the empirical study, we demonstrate the performance of our method by comparing with the following baseline algorithms including:

- `BLR`(q_0): It infers the temporal dependencies among time series using Bayesian Linear Regression with prior distribution $\mathcal{N}(\mathbf{0}, q_0^{-1}\mathbf{I}_d)$. It has been shown that the setting of the penalty parameter λ in ridge regression can be achieved by tuning q_0 accordingly [B+06].

- `BLasso`(λ): It applies Bayesian Lasso to learn the temporal dependencies, where λ is the L_1 penalty parameter. It presents an online inference for Lasso regression from Bayesian perspective [PC08].
- `TVLR`(q): It makes use of the Time-Varying Linear Regression from Bayesian perspective, which is capable of capturing the dynamics of dependency without regularization. The parameter q specifies the prior distribution $\mathcal{N}(\mathbf{0}, q^{-1}\mathbf{I}_{2d})$ for both constant and varying components of the coefficients [ZWML16].

One the other hand, we denote our proposed method as `TVLasso`(λ_1,λ_2), where the Time-Varying Bayesian Lasso regression algorithm is used to infer the time-varying temporal dependency among time series. The penalty parameters λ_1 and λ_2 are presented in Equation 5.15, determining the sparsity of both stationary component and drift component, respectively. Note that the algorithms in [SKX09] and [LKJ09] are not included as baseline algorithms in our experiment, since both are off-line algorithms, while the work of this chapter mainly focuses on online inference of time-varying temporal dependency. During our experiments, we extract small subset of data with early time stamps and employ grid search to find the optimal parameters for all the algorithm. The parameter settings are verified by cross validation in terms of the prediction errors over the extracted data subset.

5.4.2 Evaluation Measures

`AUC` Score: In order to further verify the efficacy of the proposed method for temporal dependency identification, `AUC`, the Area Under the ROC [Bra97], is applied for performance evaluation due to its independence of priors, costs, and operating points [LD06]. The value of `AUC` is the probability that the algorithm will assign a higher value to a randomly chosen existing edge than a randomly chosen non-existing edge in the temporal dependency structure. As we have mentioned in Section 5.2.1, nonzero value of $\mathbf{W}_{ij}^{l}$ indicates $\mathbf{y}_i \rightarrow_g \mathbf{y}_j$. It is reasonable to suppose that a higher absolute value of $\mathbf{W}_{ij}^{l}$ implies a larger likelihood of existing a temporal dependency $\mathbf{y}_i \rightarrow_g \mathbf{y}_j$. At each time t, an `AUC` score of the algorithm is obtained by comparing its inferred temporal dependency structure with the ground truth at t.

Prediction Error: Let $\mathbf{W}_t$ be the true coefficient matrix and $\widehat{\mathbf{W}}_t$ be the estimated coefficient matrix. We define the prediction error at time t as $\Delta = ||\mathbf{W}_t - \widehat{\mathbf{W}}_t||_F$, where $|| \bullet ||_F$ is the Frobenius Norm [CDG00]. A smaller prediction error indicates a better inference of dynamic temporal structure.

In order to give a clear illustration, we segment the time line into time buckets with the same predefined size and illustrate the performance with an average value of the corresponding measure for every time bucket.

5.4.3 Synthetic Data

The main advantage of using synthetic data sets is that the detailed dependency structures are known and hence we can systematically evaluate the performance of our proposed method with different factors such as noise and sparsity levels and quantitatively compare with other alternative solutions using various performance measures.

Synthetic Data Generation: The synthetic data generation is governed by the parameters shown in Table 5.2. The time series data are generated with the VAR model, where the coefficient value $\mathbf{W}_{ij}^{l}$ indicates the strength of dependency $\mathbf{y}_i \rightarrow_g \mathbf{y}_j$. To simulate the time-varying temporal dependencies among time series, five types of dynamics are randomly injected into the VAR model, depicting the dynamic changes of the coefficients, including:

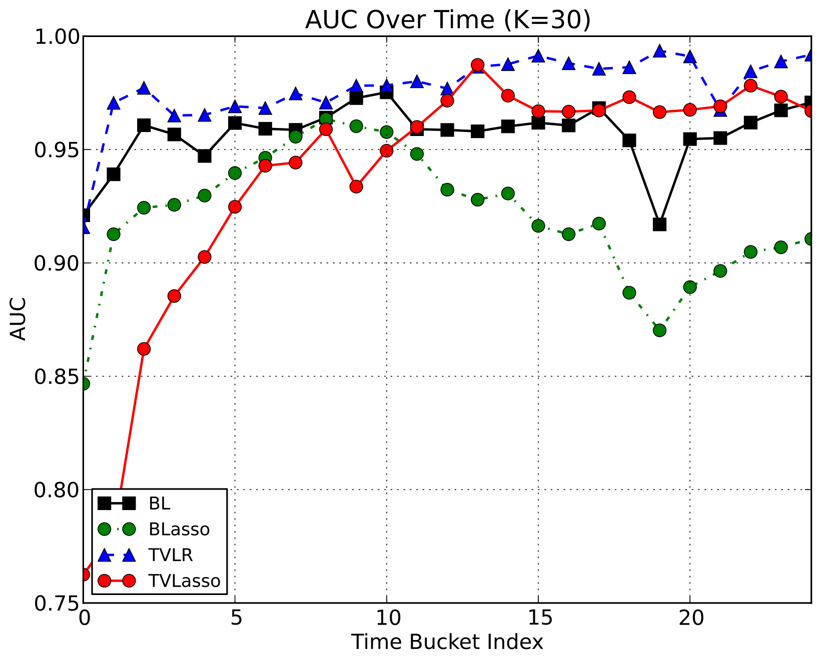


Figure 5.5: The temporal dependency identification performance is evaluated in terms of `AUC` by comparing algorithms such as `BLR`(1.0), `BLasso`(1k), `TVLR`(1.0), `TVLasso`(2k,2k). The bucket size is 200. The number of time series is 30.

(1) **Zero Value** The coefficient holds a zero value, indicating no temporal dependency existing. The number of coefficient with zero value is determined by the sparsity s.

(2) **Constant Value** The coefficient holds a constant nonzero value, which is randomly generated from the standard Gaussian distribution.

(3) **Piecewise Constant** The time line is randomly segmented into multiple intervals. The number of intervals is uniformly sampled in $(0, I]$. During each interval, the coefficient value is constant. The constant values are generated from the standard Gaussian distribution.

(4) **Periodic Change** The coefficient value varies periodically as time evolves, where the periodic change of the coefficient is simulated by sin curve whose period is uniformly sampled from the range $(0, T)$.

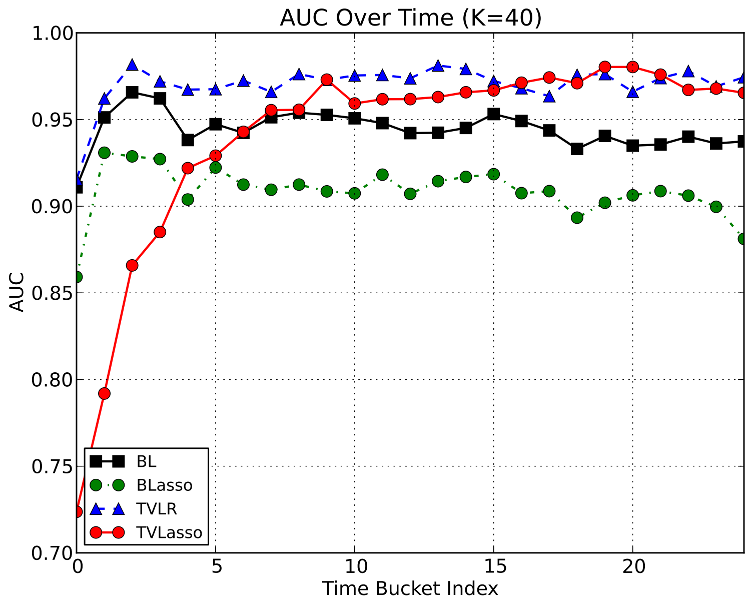


Figure 5.6: The temporal dependency identification performance is evaluated in terms of `AUC` by comparing algorithms such as `BLR`(1.0), `BLasso`(1k), `TVLR`(1.0), `TVLasso`(1k,2k). The bucket size is 200. The number of time series is 40.

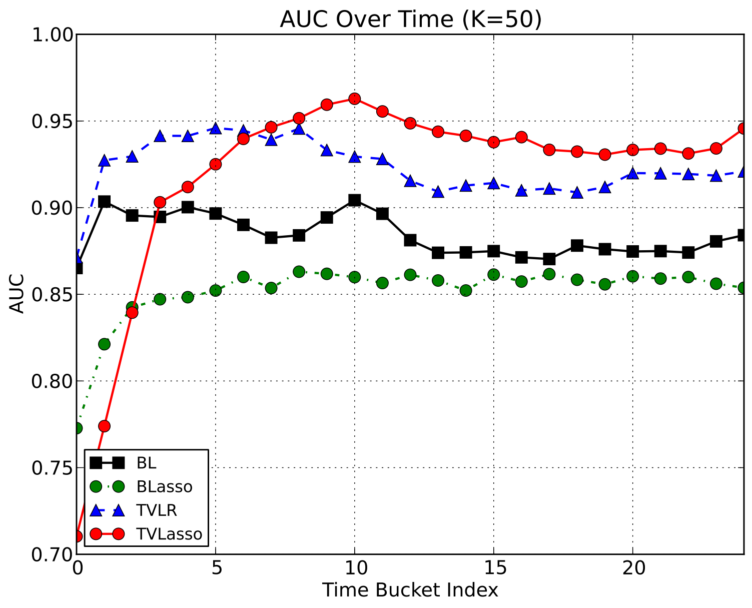


Figure 5.7: The temporal dependency identification performance is evaluated in terms of `AUC` by comparing algorithms such as `BLR`(1.0), `BLasso`(1k), `TVLR`(1.0), `TVLasso`(1k,2k). The bucket size is 200. The number of time series is 50.

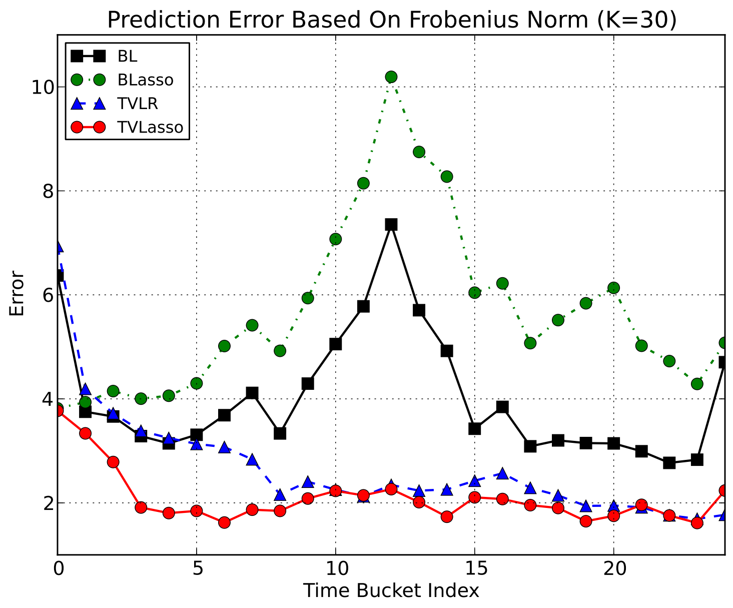


Figure 5.8: The temporal dependency identification performance is evaluated in terms of prediction error by comparing algorithms such as `BLR`(1.0), `BLasso`(1k), `TVLR`(1.0), `TVLasso`(2k,2k). The bucket size is 200. The number of time series is 30.

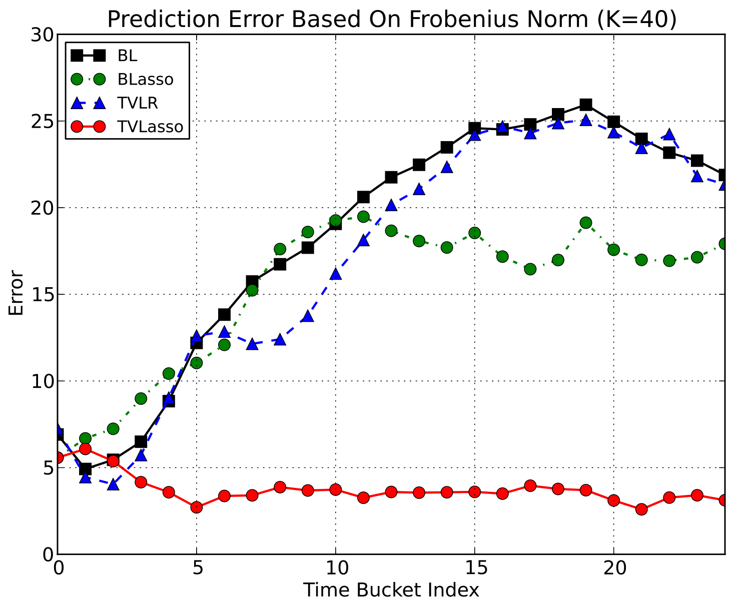


Figure 5.9: The temporal dependency identification performance is evaluated in terms of prediction error by comparing algorithms such as `BLR`(1.0), `BLasso`(1k), `TVLR`(1.0), `TVLasso`(1k,2k). The bucket size is 200. The number of time series is 40.

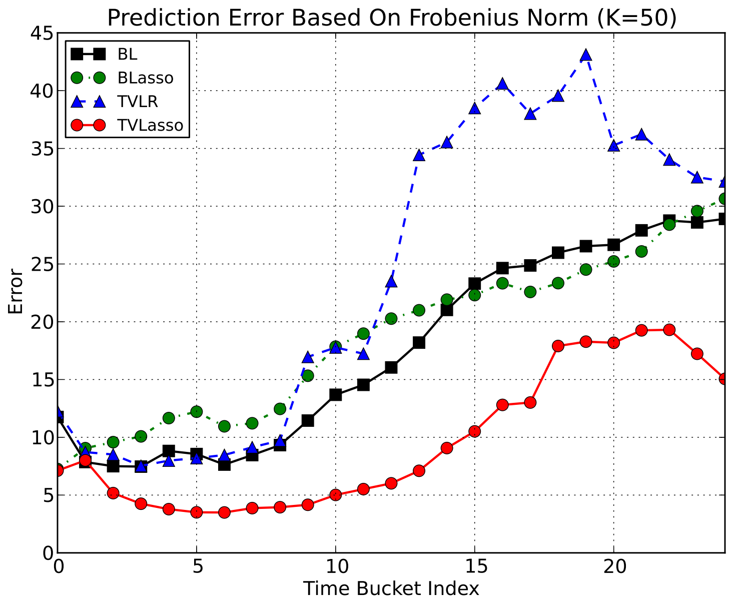


Figure 5.10: The temporal dependency identification performance is evaluated in terms of prediction error by comparing algorithms such as `BLR`(1.0), `BLasso`(1k), `TVLR`(1.0), `TVLasso`(1k,2k). The bucket size is 200. The number of time series is 50.

(5) **Random Walk** The coefficient value at time t is determined by a standard Gaussian random walk from the value at time $t-1$.

The sparsity of the temporal dependencies is regulated by s, indicating that a coefficient has the probability s to be generated by type (1). Accordingly, the other four types (2)-(5) uniformly share the probability $1-s$ for simulating the coefficient.

Dynamic Temporal Dependency Tracking: In order to show the capability in capturing the dynamic temporal dependency with a visualized straightforward example, we start with a simulation where $K = 20$, $T = 3000$, $L = 1$, $I = 10$, $s = 0.9$, $\mu = 0$ and $\sigma^2 = 1$. Both the baseline algorithms and our proposed algorithm infer the temporal dependency in an online mode. The performance of all the algorithms depends on the parameter setting. Therefore, we first conduct the performance comparison for each algorithm with diverse parameter settings. Then the one with best performance is selected for comparison study. Eight coefficients are selected and

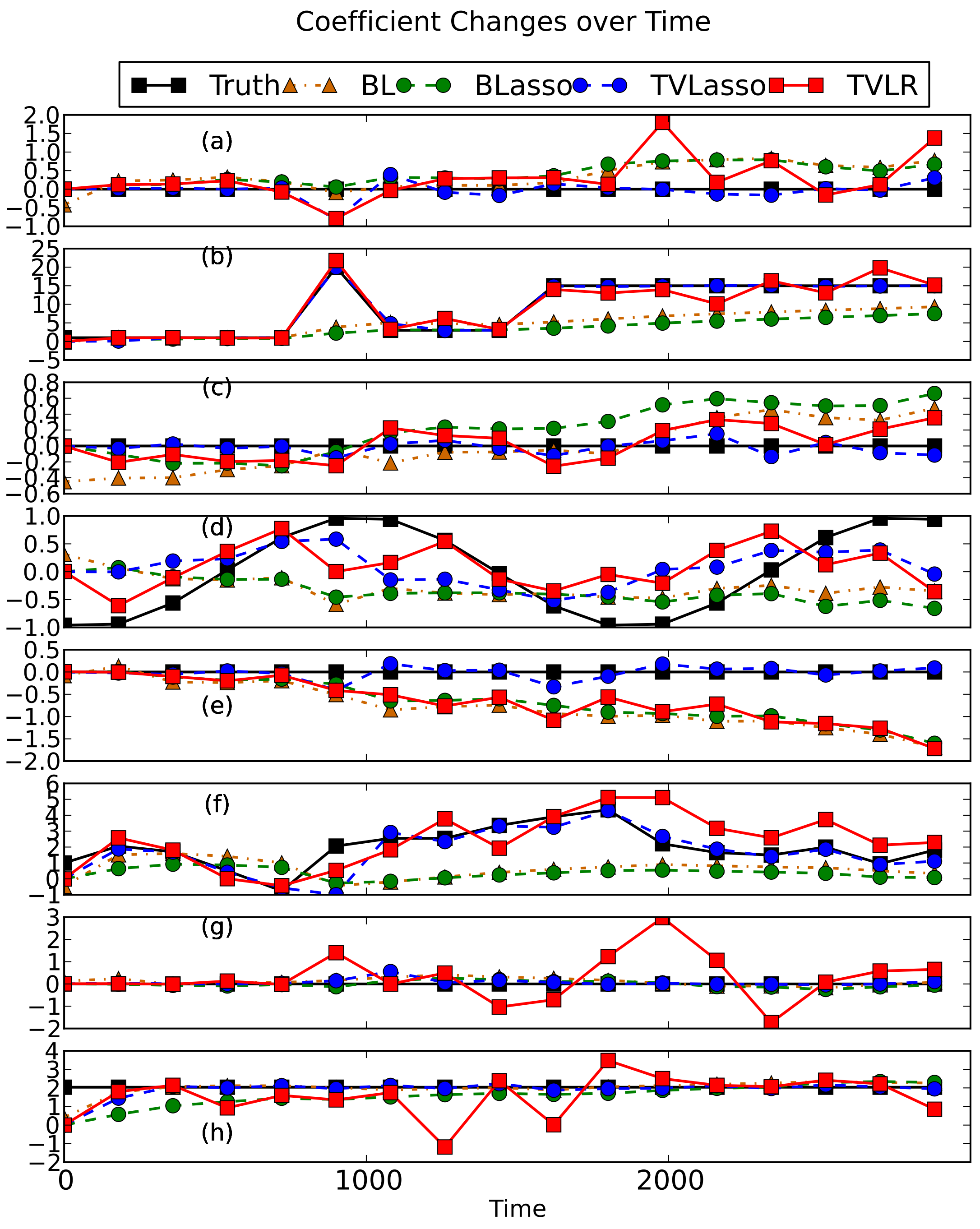


Figure 5.11: The temporal dependencies among 20 time series are leant and eight coefficients among all are selected for demonstration. Coefficients with zero values are displayed in (a),(c),(e) and (g). The coefficients with piecewise constant, periodic change, random walk and constant value are shown in (b),(d),(f) and (h), respectively.

Table 5.2: Parameters for Synthetic Data Generation

Name	Description
K	The number of time series.
T	The length of time series.
L	The maximum time lag for VAR model.
I	The maximum number of intervals used to segmented the time line.
s	The sparsity of the temporal dependency, denoted as the ratio of coefficients with zero value to K.
μ	The mean of the noise introduced during regression.
σ^2	The variance of the noise introduced during regression.

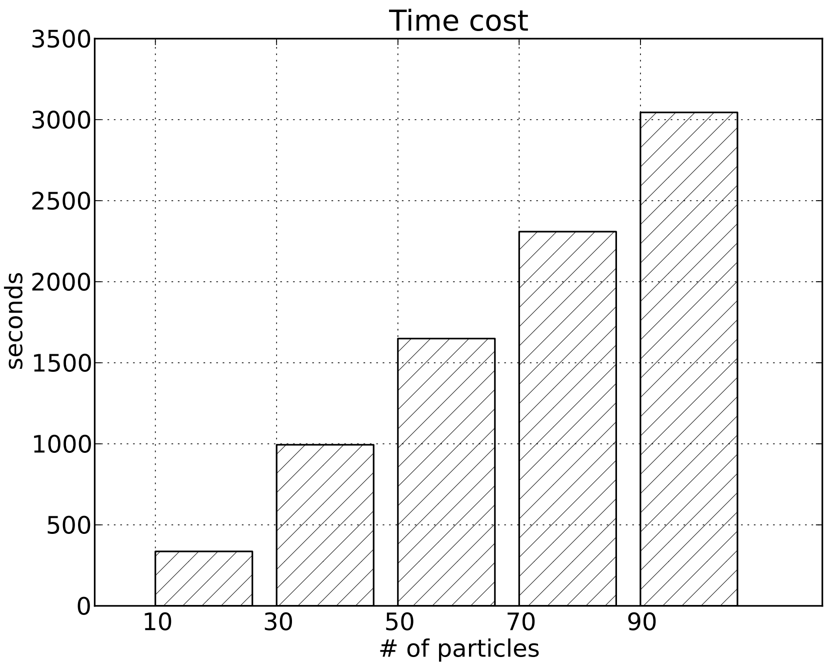


Figure 5.12: The time cost of `TVLasso` with different number of particles.

displayed in Figure 5.11. It shows our proposed algorithm `TVLasso` can effectively capture the time-varying temporal dependency with different types of dynamics. The

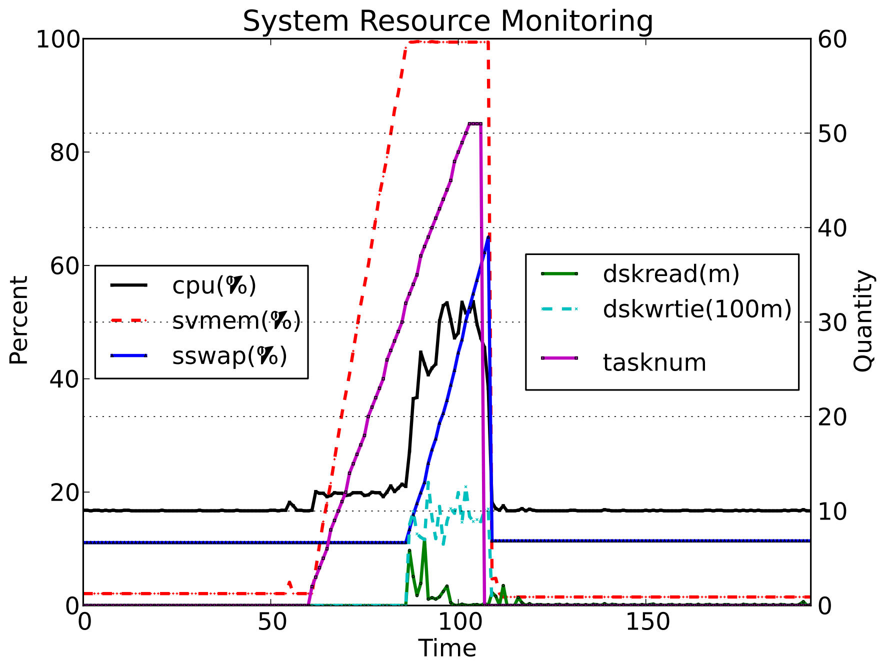


Figure 5.13: The system resource monitoring time series collected every 5 seconds.

`BLasso` algorithm shows more robustness than `BL` for zero-value coefficient inference, and is more suitable for inference with high sparsity. The algorithm `TVLR` captures the dynamic change of the coefficients better than both `BLasso` and `BL`, but it is less stable when comparing with `TVLasso`.

Performance Evaluation: We continue to conduct the evaluation in terms of `AUC` and prediction error over a simulation data set with higher dimension, where $K = (30, 40, 50)$, $T = 5000$, $L = 1$, $I = 10$, $s = 0.9$, $\mu = 0$ and $\sigma^2 = 1$. The evaluations with different Ks in terms of `AUC` are depicted in Figure 5.5, Figure 5.6 and Figure 5.7, respectively. The performance of `TVLasso` is comparable with `TVLR` in low dimensions, while `TVLasso` quickly catches up with other baseline algorithms at the beginning and keeps outperforming them in high dimensions. Comparing with other two baseline algorithms, `TVLR` shows a relatively good performance since it models the dynamic change explicitly.

In terms of prediction error, `TVLasso` incurs lowest prediction error consistently, shown in Figure 5.8, Figure 5.9 and Figure 5.10. When in high dimension, `TVLR` gets the highest prediction error even though it obtains relatively high `AUC` score, where the reason is that the `AUC` is computed based on the absolute value of the coefficient. The conclusion is that our proposed algorithm `TVLasso` is consistent in the coefficient prediction while `TVLR` may suffer coefficient prediction with opposite sign of the truth, especially in high dimensions.

Time Cost: The time cost increases linearly as the number of particles shown in Figure 5.12.

5.4.4 Case Study

System Management

We conduct the case study in a real system FIU-Miner [ZJZ$^+$13a], which is a fast, integrated and user-friendly system for data mining in distributed system. FIU-Miner composes every job as a workflow where a set of computing tasks are organized in a dependency graph. A job of FIU-Miner can be scheduled in different ways, such as one-time execution at a particular time, periodic execution every one predefined time interval. To help FIU-Miner make decisions on job scheduling, the system monitoring agents are deployed to all the computing nodes in the distributed environment, and periodically collect the information about both resource usage and running processes. The resource usage information includes CPU utilization, memory usage, disk I/O, networking I/O, etc. The running process information describes the status, the number of running instances aggregated by the program, running time, and so forth. An alert is raised if the predefined monitoring situation persists violated beyond a particular duration. We deploy our algorithm with FIU-Miner to instantly infer the causal dependency among the collected monitoring information.

To illustrate the efficacy of our method, we inspect an alert raised at the time stamp 2016-07-06 01:30:39,852, when a persistent high system load occurred. The process information is aggregated by the executable program. The number of instances for a matrix computation program is identified with strong dependencies between other system resource monitoring time series. The system monitoring time series as well as the number of instances for the identified program are displayed in Fig 5.13. Here `cpu(%)`, `svmem(%)`, `sswap(%)`, `dskread(m)`, `dskwrite(100m)` and `tasknum` represent the CPU utilization, virtual memory usage, swap memory usage, the number of bytes reading from the disk, the number of bytes writing to the disk, and the number of instances for a matrix computation program, respectively. `cpu(%)`, `svmem(%)` and `sswap(%)` share the Percent axis, and `dskread(m)`, `dskwrite(100m)` and `tasknum` share the Quantity axis. Each computing node in the distributed environment has $31G$ memory in total. The causal dependencies discovered by multiple algorithms are shown in Figure 5.14. The `tasknum` increases linearly to 52 at the beginning, and then decreases to 0 abruptly.

After meticulously inspecting the source code of the matrix computation program, each instance allocates $1G$ memory for holding the matrix data, but does not explicitly recycle the used memory after computation. FIU-Miner schedules the program periodically as a sub-process but does not reap the completed sub-processes until all the sub-processes have been scheduled. It ends in a number of zombie processes during scheduling and causes a resource leak.

As illustrated by the algorithm `TVLasso` in Figure 5.14, at the early stage, `tasknum` strongly infers `cpu`, `svmem`, and `sswap`. After the consumed memory exceeds the total available memory of the computing node, `tasknum` has strong causal relations with `dskread` and `dskwrite`. Finally, the temporal dependencies disappear after all the sub-processes are reaped by the schedule process of FIU-Miner. However, the

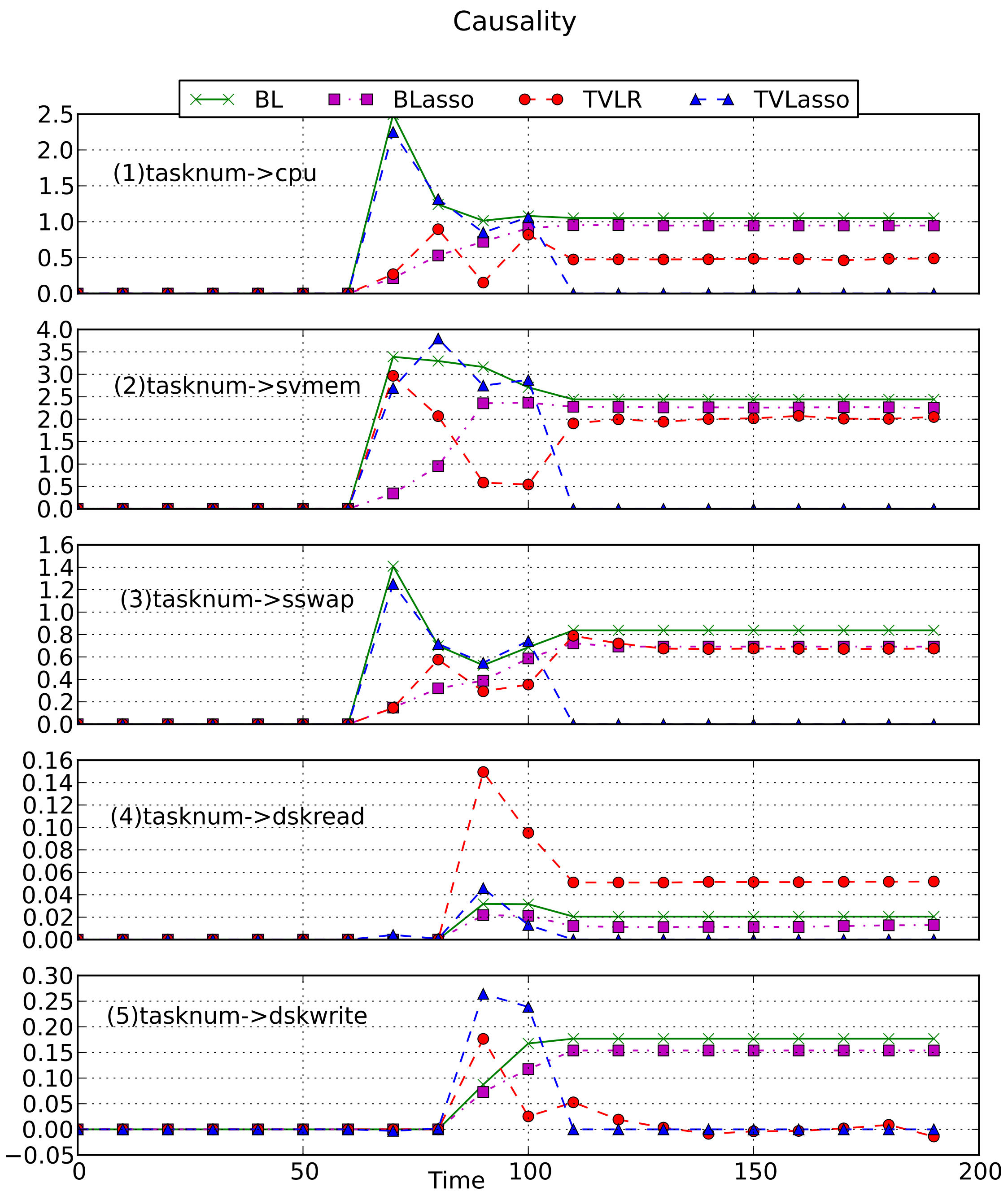


Figure 5.14: Temporal dependencies among system resource monitoring time series are discovered.

baseline algorithms can not effectively react with the dynamic changes of temporal dependencies.

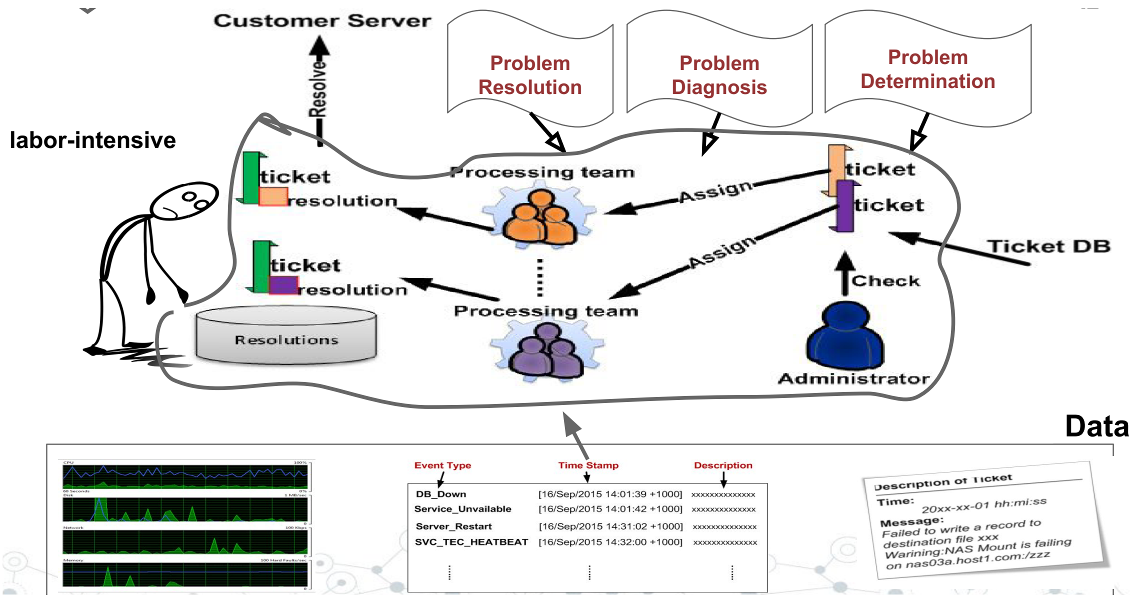
5.5 Summary

In this chapter, we take the dynamic change of the underlying temporal dependencies among time series into account and explicitly model the dynamic change as a random walk. In order conduct online inference which is often required especially in system management scenarios, we propose a method based on the particle learning to efficiently infer both parameters and latent variables simultaneously. The empirical study is conducted on both synthetic and real data to verify the efficiency and efficacy of our proposed method. The experimental result shows that our method can effectively track the time-varying temporal dependencies among time series and outperforms the existing methods especially when tackling the data with high dimension and sparsity.

CHAPTER 6

CONCLUSION AND FUTURE WORK

Optimizing the quality of service delivery improves customer satisfaction and sharpens the competitive edge of business. The routine IT maintenance procedure plays an essential part in IT service management optimization. However, as we discussed in previous chapters, the entire routine IT maintenance procedure gets involved with large amount of human efforts since the complete automation is not feasible (shown in Figure 6.1). Therefore, maximal automation of routine IT maintenance procedure is one of ultimate goals of IT service management optimization. After meticulously studying the entire IT maintenance procedure and inspecting the data (i.e., IT incident tickets, system monitoring events, time series for system performance statistics), three research directions are identified from data mining perspective, with the purpose of providing an integrated and intelligent solution to facilitate the IT activities such as problem determination, diagnosis and resolution.



```
DB_Down              [16/Sep/2015 14:01:39 +1000]  xxxxxxxxxxxxxx
Service_Unvailable   [16/Sep/2015 14:01:42 +1000]  xxxxxxxxxxxxxx
Server_Restart       [16/Sep/2015 14:31:02 +1000]  xxxxxxxxxxxxxx
SVC_TEC_HEATBEAT     [16/Sep/2015 14:32:00 +1000]  xxxxxxxxxxxxxx
      ⋮                         ⋮                         ⋮
```

Description of Ticket
Time:
20xx-xx-01 hh:mi:ss
Message:
Failed to write a record to destination file xxx
Warining:NAS Mount is failing on nas03a.host1.com:/zzz

Figure 6.1: The summary of my dissertation research on IT service management optimization.

The three research directions are highlighted as below:

1. automatically determining problem categories according to the symptom description in a ticket;
2. intelligently discovering interesting temporal patterns from system events;
3. instantly identify temporal dependencies among system performance statistics data.

To follow up with the work in my dissertation, some future work along the three directions are provided.

- We focus on tree-based hierarchy in the dissertation, which can be extended to DAG-based hierarchy in future work. In addition, more domain expert knowledge can be automatically incorporated into the framework to reduce the system administrators' involvement in the overall system proposed in this dissertation. Based on KILO algorithm, another direction is to propose an framework which is capable of refining the category determination interactively with further knowledge.
- In the future, we will extend our model to discover temporal patterns with more complicated distributions of time lags. This includes patterns with potential multiple time lags between two events that satisfy more complicated distribution laws. The next challenging step is to move from the discovery of pairwise dependencies to the discovery of multi-event dependencies. These realistic conditions will be considered in our future work.
- To discover the time-varying temporal dependency among time series, the choice of penalty parameters is very essential. One possible future work is to come up with online method to automatically identify the proper parameters. The time-varying temporal dependency discovery among time series unveils the dynamic

change of the system structure over time. Another possible direction is to apply the discovered time-varying temporal dependency for anomaly detection.

- The system administrators act as inevitable roles during the routine IT maintenance procedure of IT service management. The reason is that a great amount of domain expertise has been accumulated, which implicitly, efficiently and effectively facilitate the IT activities. Therefore, one of the future work is to come up with a way to explicitly model and store those domain knowledge, and integrate those domain knowledge with our works in the dissertation. It will further optimize the IT service management and alleviate the human efforts.

BIBLIOGRAPHY

[ABCM09] Michal Aharon, Gilad Barash, Ira Cohen, and Eli Mordechai. One graph is worth a thousand logs: Uncovering hidden structures in massive system event logs. In *Machine Learning and Knowledge Discovery in Databases*, pages 227–243. Springer, 2009.

[ABD$^+$07] Naga Ayachitula, Melissa Buco, Yixin Diao, Surendra Maheswaran, Raju Pavuluri, Larisa Shwartz, and Chris Ward. It service management automation-a hybrid methodology to integrate and orchestrate collaborative human centric and automation centric workflows. In *Services Computing, 2007. SCC 2007. IEEE International Conference on*, pages 574–581. IEEE, 2007.

[AC06] Raman Adaikkalavan and Sharma Chakravarthy. Snoopib: Interval-based event specification and detection for active databases. *Data & Knowledge Engineering*, 59(1):139–165, 2006.

[AFGY02a] Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yiu. Sequential pattern mining using a bitmap representation. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 429–435. ACM, 2002.

[AFGY02b] Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yiu. Sequential pattern mining using a bitmap representation. In *Proceedings of KDD*, pages 429–435, 2002.

[AIS93] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pages 207–216. ACM, 1993.

[ALA07] Andrew Arnold, Yan Liu, and Naoki Abe. Temporal causal modeling with graphical granger methods. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 66–75. ACM, 2007.

[ALR01] Algirdas Avizienis, Jean-Claude Laprie, and Brian Randell. *Fundamental concepts of dependability*. University of Newcastle upon Tyne, Computing Science, 2001.

[AS95] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, pages 3–14. IEEE, 1995.

[AZ12] Charu C Aggarwal and ChengXiang Zhai. A survey of text classification algorithms. In *Mining text data*, pages 163–222. Springer, 2012.

[B$^+$06] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 4. springer New York, 2006.

[BDH02] Paul N Bennett, Susan T Dumais, and Eric Horvitz. Probabilistic combination of text classifiers using reliability indicators: Models and results. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 207–214. ACM, 2002.

[BJ94] Richard G Baraniuk and Douglas L Jones. A signal-dependent time-frequency representation: Fast algorithm for optimal kernel design. *Signal Processing, IEEE Transactions on*, 42(1):134–146, 1994.

[BK11] Wei Bi and James T Kwok. Multi-label classification on tree-and dag-structured hierarchies. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 17–24, 2011.

[BK12] Wei Bi and James T Kwok. Mandatory leaf node prediction in hierarchical multilabel classification. In *Advances in Neural Information Processing Systems*, pages 153–161, 2012.

[BL12] Mohammad Taha Bahadori and Yan Liu. On causality inference in time series. In *AAAI Fall Symposium: Discovery Informatics*, 2012.

[BL13] Mohammad Taha Bahadori and Yan Liu. An examination of practical granger causality inference. In *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 2013.

[BLSB04] Matthew R Boutell, Jiebo Luo, Xipeng Shen, and Christopher M Brown. Learning multi-label scene classification. *Pattern recognition*, 37(9):1757–1771, 2004.

[BNJ03] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[BO07] Khellaf Bouandas and Aomar Osmani. Mining association rules in temporal sequences. In *Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on*, pages 610–615. IEEE, 2007.

[Bra97] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[BSS$^+$06] Hendrik Blockeel, Leander Schietgat, Jan Struyf, Sašo Džeroski, and Amanda Clare. *Decision trees for hierarchical multilabel classification: A case study in functional genomics.* Springer, 2006.

[BST06] Zafer Barutcuoglu, Robert E Schapire, and Olga G Troyanskaya. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836, 2006.

[CBGZ06a] Nicolò Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. Hierarchical classification: combining bayes with svm. In *Proceedings of the 23rd international conference on Machine learning*, pages 177–184. ACM, 2006.

[CBGZ06b] Nicolò Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. Incremental algorithms for hierarchical classification. *The Journal of Machine Learning Research*, 7:31–54, 2006.

[CBL14] Dehua Cheng, Mohammad Taha Bahadori, and Yan Liu. Fblg: A simple and effective approach for temporal dependence discovery from time series data. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 382–391. ACM, 2014.

[CBV10] Nicolo Cesa-Bianchi and Giorgio Valentini. Hierarchical cost-sensitive algorithms for genome-wide gene function prediction. 2010.

[CC05] Vitor R Carvalho and William W Cohen. On the collective classification of email speech acts. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 345–352. ACM, 2005.

[CCM04] William W Cohen, Vitor R Carvalho, and Tom M Mitchell. Learning to classify email into“speech acts”. In *EMNLP*, pages 309–316, 2004.

[CDAR97] Soumen Chakrabarti, Byron Dom, Rakesh Agrawal, and Prabhakar Raghavan. Using taxonomy, discriminants, and signatures for navigating in text databases. In *VLDB*, volume 97, pages 446–455, 1997.

[CDG00] Bruno Carpentieri, Iain S Duff, and Luc Giraud. Sparse pattern selection strategies for robust frobenius-norm minimization preconditioners in electromagnetism. *Numerical linear algebra with applications*, 7(7-8):667–685, 2000.

[CH98] William W Cohen and Haym Hirsh. Joins that generalize: Text classification using whirl. In *KDD*, pages 169–173, 1998.

[CH04] Lijuan Cai and Thomas Hofmann. Hierarchical document categorization with support vector machines. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, pages 78–87. ACM, 2004.

[CH07] Lijuan Cai and Thomas Hofmann. Exploiting known taxonomies in learning overlapping concepts. In *IJCAI*, volume 7, pages 708–713, 2007.

[CH09] Weiwei Cheng and Eyke Hüllermeier. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2-3):211–225, 2009.

[CJLP10] Carlos Carvalho, Michael S Johannes, Hedibert F Lopes, and Nick Polson. Particle learning and smoothing. *Statistical Science*, 25(1):88–106, 2010.

[CK01] Amanda Clare and Ross D King. Knowledge discovery in multi-label phenotype data. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 42–53. Springer, 2001.

[CK03] Amanda Clare and Ross D King. Predicting gene function in saccharomyces cerevisiae. *Bioinformatics*, 19(suppl 2):ii42–ii49, 2003.

[CM12] Gianpaolo Cugola and Alessandro Margara. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys (CSUR)*, 44(3):15, 2012.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[DC00] Susan Dumais and Hao Chen. Hierarchical classification of web content. In *Proceedings of the 23rd annual international ACM SIGIR conference* *on Research and development in information retrieval*, pages 256–263. ACM, 2000.

[DDF$^+$90] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990.

[DDFG01] Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001.

[DGA00] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and computing*, 10(3):197–208, 2000.

[DHS12] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[DJL09] Yixin Diao, Hani Jamjoom, and David Loewenstern. Rule-based problem classification in it service management. In *Cloud Computing, 2009. CLOUD'09. IEEE International Conference on*, pages 221–228. IEEE, 2009.

[DKFH12] Jeroen De Knijff, Flavius Frasincar, and Frederik Hogenboom. Domain taxonomy learning from text: The subsumption method versus hierarchical clustering. *Data & Knowledge Engineering*, 2012.

[DKH12] Krzysztof Dembczynski, Wojciech Kotlowski, and Eyke Hüllermeier. Consistent multilabel ranking through univariate losses. *arXiv preprint arXiv:1206.6401*, 2012.

[DKS04] Ofer Dekel, Joseph Keshet, and Yoram Singer. Large margin hierarchical classification. In *Proceedings of the twenty-first international conference on Machine learning*, page 27. ACM, 2004.

[DKS05] Ofer Dekel, Joseph Keshet, and Yoram Singer. An online algorithm for hierarchical phoneme classification. In *Machine Learning for Multimodal Interaction*, pages 146–158. Springer, 2005.

[DKZ$^+$03] Petar M Djuric, Jayesh H Kotecha, Jianqui Zhang, Yufei Huang, Tadesse Ghirmai, Mónica F Bugallo, and Joaquin Miguez. Particle filtering. *IEEE signal processing magazine*, 20(5):19–38, 2003.

[DWCH10] Krzysztof Dembczyński, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. Regret analysis for performance metrics in multi-label classification: the case of hamming and subset zero-one loss. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 280–295. Springer, 2010.

[DWCH12] Krzysztof Dembczyński, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. On label dependence and loss minimization in multi-label classification. *Machine Learning*, 88(1-2):5–45, 2012.

[Eic06] Michael Eichler. Graphical modelling of multivariate time series with latent variables. *Preprint, Universiteit Maastricht*, 2006.

[EW01] André Elisseeff and Jason Weston. A kernel method for multi-labelled classification. In *Advances in neural information processing systems*, pages 681–687, 2001.

[FHMB08] Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and K-laus Brinker. Multilabel classification via calibrated label ranking. *Machine learning*, 73(2):133–153, 2008.

[Gew82] John Geweke. Measurement of linear dependence and feedback between multiple time series. *Journal of the American statistical association*, 77(378):304–313, 1982.

[GM05] Nadia Ghamrawi and Andrew McCallum. Collective multi-label classification. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 195–200. ACM, 2005.

[Gra69] Clive WJ Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society*, pages 424–438, 1969.

[Gra80] Clive WJ Granger. Testing for causality: a personal viewpoint. *Journal of Economic Dynamics and control*, 2:329–352, 1980.

[Gra03] Michael Granitzer. *Hierarchical text classification using methods from machine learning*. Citeseer, 2003.

[GRS99] Minos N Garofalakis, Rajeev Rastogi, and Kyuseok Shim. Spirit: Sequential pattern mining with regular expression constraints. In *VLDB*, volume 99, pages 7–10, 1999.

[GS04] Shantanu Godbole and Sunita Sarawagi. Discriminative methods for multi-labeled classification. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 22–30. Springer, 2004.

[GS11] Yuhong Guo and Dale Schuurmans. Adaptive large margin training for multilabel classification. In *AAAI*, 2011.

[Hal62] John H Halton. Sequential monte carlo. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 58, pages 57–78. Cambridge Univ Press, 1962.

[Har90] Andrew C Harvey. *Forecasting, structural time series models and the Kalman filter*. Cambridge university press, 1990.

[HCC03] Thomas Hofmann, Lijuan Cai, and Massimiliano Ciaramita. Learning with taxonomies: Classifying documents and words. In *NIPS workshop on syntax, semantics, and statistics*, 2003.

[HCF95] K Houck, S Calo, and A Finkel. Towards a practical alarm correlation system. In *Integrated Network Management IV*, pages 226–237. Springer, 1995.

[Hec98] David Heckerman. A tutorial on learning with bayesian networks. In *Learning in graphical models*, pages 301–354. Springer, 1998.

[HHAI95] Manilla Heikki, Toivonen Hannu, and Verkamo A. Inkeri. Discovering frequent episodes in sequences. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 210–215, 1995.

[HJP03] Peg Howland, Moongu Jeon, and Haesun Park. Structure preserving dimension reduction for clustered text data based on the generalized singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 25(1):165–179, 2003.

[HKK01] Eui-Hong Sam Han, George Karypis, and Vipin Kumar. Text categorization using weight adjusted k-nearest neighbor classification. In

Pacific-asia conference on knowledge discovery and data mining, pages 53–65. Springer, 2001.

[HMP02] Joseph L. Hellerstein, Sheng Ma, and C-S Perng. Discovering actionable patterns in event data. *IBM Systems Journal*, 41(3):475–493, 2002.

[Hof99] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

[HP04] Peg Howland and Haesun Park. Generalizing discriminant analysis using the generalized singular value decomposition. *IEEE transactions on pattern analysis and machine intelligence*, 26(8):995–1006, 2004.

[HPMA$^+$00a] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Freespan: frequent pattern-projected sequential pattern mining. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 355–359. ACM, 2000.

[HPMA$^+$00b] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. Freespan: frequent pattern-projected sequential pattern mining. In *Proccedings of KDD*, pages 355–359, 2000.

[HPO] HPOpenView. HP OpenView : Network and Systems Management Products. `http://www8.hp.com/us/en/software/enterprise-software.html`.

[HPS96] David A Hull, Jan O Pedersen, and Hinrich Schütze. Method combination for document filtering. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 279–287. ACM, 1996.

[HZMVV10] Bharath Hariharan, Lihi Zelnik-Manor, Manik Varma, and Svn Viswanathan. Large scale max-margin multi-label classification with priors. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 423–430, 2010.

[IA12] Khan Irfan and Jain Anoop. A comprehensive survey on sequential pattern mining. *International Journal of Engineering Research and Technology*, 2012.

[IBM16] IBMTivoli. IBM Tivoli Monitoring. `https://www.ibm.com/software/tivoli`, 2016.

[Jen96] Finn V Jensen. *An introduction to Bayesian networks*, volume 210. UCL press London, 1996.

[JGB$^+$02] L Juhl Jensen, Ramneek Gupta, Nikolaj Blom, D Devos, J Tamames, Can Kesmir, Henrik Nielsen, Hans Henrik Stærfeldt, Krzysztof Rapacki, Christopher Workman, et al. Prediction of human protein function from post-translational modifications and localization features. *Journal of molecular biology*, 319(5):1257–1265, 2002.

[Joa98] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[Jol02] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[Jor02] A Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems*, 14:841, 2002.

[JPLC12] Yexi Jiang, Chang-Shing Perng, Tao Li, and Rong Chang. Intelligent cloud capacity management. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 502–505. IEEE, 2012.

[JTYY08] Shuiwang Ji, Lei Tang, Shipeng Yu, and Jieping Ye. Extracting shared subspace for multi-label classification. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 381–389. ACM, 2008.

[JYG$^+$03] Ronald Jansen, Haiyuan Yu, Dov Greenbaum, Yuval Kluger, Nevan J Krogan, Sambath Chung, Andrew Emili, Michael Snyder, Jack F Greenblatt, and Mark Gerstein. A bayesian networks approach for predicting protein-protein interactions from genomic data. *Science*, 302(5644):449–453, 2003.

[JZXL14] Yexi Jiang, Chunqiu Zeng, Jian Xu, and Tao Li. Real time contextual collective anomaly detection over multiple data streams. *Proceedings of the ODD*, pages 23–30, 2014.

[KL51] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, pages 79–86, 1951.

[Kle03] Jon Kleinberg. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*, 7(4):373–397, 2003.

[KWI$^+$11] Cristina Kadar, Dorothea Wiesmann, Jose Iria, Dirk Husemann, and Mario Lucic. Automatic classification of change requests for improved it service quality. In *SRII Global Conference (SRII), 2011 Annual*, pages 430–439. IEEE, 2011.

[LALR09] Aurélie C Lozano, Naoki Abe, Yan Liu, and Saharon Rosset. Grouped graphical granger modeling for gene expression regulatory networks discovery. *Bioinformatics*, 25(12):i110–i118, 2009.

[Lan95] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th international conference on machine learning*, pages 331–339, 1995.

[LC96] Leah S Larkey and W Bruce Croft. Combining classifiers in text categorization. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 289–297. ACM, 1996.

[LD06] Thomas Landgrebe and R Duin. A simplified extension of the area under the roc to the multiclass domain. In *Seventeenth annual symposium of the pattern recognition association of South Africa*, pages 241–245, 2006.

[LJ98] Yong H Li and Anil K Jain. Classification of text documents. *The Computer Journal*, 41(8):537–546, 1998.

[LK97] David D Lewis and Kimberly A Knowles. Threading electronic mail: A preliminary study. *Information processing & management*, 33(2):209–217, 1997.

[LKJ09] Yan Liu, Jayant R Kalagnanam, and Oivind Johnsen. Learning dynamic temporal graphs for oil-production equipment monitoring system.

In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234. ACM, 2009.

[LL99] Savio LY Lam and Dik Lun Lee. Feature reduction for neural network based text categorization. In *Database Systems for Advanced Applications, 1999. Proceedings., 6th International Conference on*, pages 195–202. IEEE, 1999.

[LL01] Wai Lam and Kwok-Yin Lai. A meta-learning approach for text categorization. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 303–309. ACM, 2001.

[LL13] Lei Li and Tao Li. An empirical study of ontology-based multi-document summarization in disaster management. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2013.

[LLL+14] Chen Luo, Jian-Guang Lou, Qingwei Lin, Qiang Fu, Rui Ding, Dongmei Zhang, and Zhe Wang. Correlating events with time series for incident diagnosis. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1583–1592. ACM, 2014.

[LLMP05] Tao Li, Feng Liang, Sheng Ma, and Wei Peng. An integrated framework on mining logs files for computing system management. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 776–781. ACM, 2005.

[LLNM+09] Aurelie C Lozano, Hongfei Li, Alexandru Niculescu-Mizil, Yan Liu, Claudia Perlich, Jonathan Hosking, and Naoki Abe. Spatial-temporal causal modeling for climate change attribution. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 587–596. ACM, 2009.

[LM04] Tao Li and Sheng Ma. Mining temporal patterns without predefined time windows. In *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*, pages 451–454. IEEE, 2004.

[LMH02] Feng Liang, Sheng Ma, and Joseph L Hellerstein. Discovering fully dependent patterns. In *SDM*, pages 511–527. SIAM, 2002.

[log16a] logicMonitor. LogicMonitor. `http://www.logicmonitor.com/`, 2016.

[Log16b] LogITIL. ITIL. `http://www.itil-officialsite.com`, 2016.

[LPG02] G di Lucca, M Di Penta, and Sara Gradara. An approach to classify software maintenance requests. In *Software Maintenance, 2002. Proceedings. International Conference on*, pages 93–102. IEEE, 2002.

[LPP$^+$10] Tao Li, Wei Peng, Charles Perng, Sheng Ma, and Haixun Wang. An integrated data-driven framework for computing system management. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 40(1):90–99, 2010.

[LSLW12] Xueqing Liu, Yangqiu Song, Shixia Liu, and Haixun Wang. Automatic taxonomy construction from keywords. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1433–1441. ACM, 2012.

[Luc08] David Luckham. *The power of events: An introduction to complex event processing in distributed enterprise systems.* Springer, 2008.

[LZ12] Bing Liu and Lei Zhang. A survey of opinion mining and sentiment analysis. In *Mining text data*, pages 415–463. Springer, 2012.

[Ma98] Bing Liu Wynne Hsu Yiming Ma. Integrating classification and association rule mining. In *Proceedings of the fourth international conference on knowledge discovery and data mining*, 1998.

[Man16] ManageEngine. ManageEngine. `https://www.manageengine.com/`, 2016.

[MBW14] Andrii Maksai, Jasmina Bogojeska, and Dorothea Wiesmann. Hierarchical incident ticket classification with minimal supervision. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 923–928. IEEE, 2014.

[MH01] Sheng Ma and Joseph L Hellerstein. Mining partially periodic event patterns with unknown periods. In *Data Engineering, 2001. Proceedings. 17th International Conference on*, pages 205–214. IEEE, 2001.

[MHPG02] Sheng Ma, Joseph L Hellerstein, Chang-shing Perng, and Genady Grabarnik. Progressive and interactive analysis of event data using event miner. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 661–664. IEEE, 2002.

[MRS08] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

[MSGL09] Patricia Marcu, Larisa Shwartz, Genady Grabarnik, and David Loewenstern. Managing faults in the service delivery process of service provider coalitions. In *Services Computing, 2009. SCC'09. IEEE International Conference on*, pages 65–72. IEEE, 2009.

[MTV97] Heikki Mannila, Hannu Toivonen, and A Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data mining and knowledge discovery*, 1(3):259–289, 1997.

[Mur02] Kevin Patrick Murphy. *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, University of California, Berkeley, 2002.

[NKT10] Gulisong Nasierding, Abbas Z Kouzani, and Grigorios Tsoumakas. A triple-random ensemble classification method for mining multi-label data. In *2010 IEEE International Conference on Data Mining Workshops*, pages 49–56. IEEE, 2010.

[P$^+$99] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 10(3):61–74, 1999.

[PC08] Trevor Park and George Casella. The bayesian lasso. *Journal of the American Statistical Association*, 103(482):681–686, 2008.

[PHMA$^+$01a] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 0215–0215. IEEE Computer Society, 2001.

[PHMA$^+$01b] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. Prefixspan: Mining sequential patterns by prefix-projected growth. In *Proceedings of EDBT*, pages 215–224, 2001.

[PHMAZ00] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, and Hua Zhu. Mining access patterns efficiently from web logs. In *Knowledge Discovery and*

Data Mining. Current Issues and New Applications, pages 396–407. Springer, 2000.

[PTG$^+$03] C.S. Perng, D. Thoenen, G. Grabarnik, S. Ma, and J. Hellerstein. Data-driven validation, completion and construction of event relationship networks. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pages 729–734. ACM, 2003.

[QHR$^+$07] Guo-Jun Qi, Xian-Sheng Hua, Yong Rui, Jinhui Tang, Tao Mei, and Hong-Jiang Zhang. Correlative multi-label video annotation. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 17–26. ACM, 2007.

[Qui86] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[RPH08] Jesse Read, Bernhard Pfahringer, and Geoff Holmes. Multi-label classification using ensembles of pruned sets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 995–1000. IEEE, 2008.

[RPHF09] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 254–269. Springer, 2009.

[RPHF11] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine learning*, 85(3):333–359, 2011.

[RS99] Miguel E Ruiz and Padmini Srinivasan. Hierarchical neural networks for text categorization (poster abstract). In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 281–282. ACM, 1999.

[RS02] Miguel E Ruiz and Padmini Srinivasan. Hierarchical text categorization using neural networks. *Information Retrieval*, 5(1):87–118, 2002.

[SA96a] Ramakrishnan Srikant and Rakesh Agrawal. *Mining sequential patterns: Generalizations and performance improvements.* Springer, 1996.

[SA96b] Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proceedings of EDBT*, pages 3–17, 1996.

[Sch99] D. J. Schroeder. *Astronomical optics*. Academic Press, 1999.

[SDdFG13] Adrian Smith, Arnaud Doucet, Nando de Freitas, and Neil Gordon. *Sequential Monte Carlo methods in practice*. Springer Science & Business Media, 2013.

[SDHH98] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 workshop*, volume 62, pages 98–105, 1998.

[SJF11] Carlos N Silla Jr and Alex A Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2):31–72, 2011.

[SKX09] Le Song, Mladen Kolar, and Eric P Xing. Time-varying dynamic bayesian networks. In *Advances in Neural Information Processing Systems*, pages 1732–1740, 2009.

[SL01] Aixin Sun and Ee-Peng Lim. Hierarchical text classification and evaluation. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 521–528. IEEE, 2001.

[SM86] Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.

[SS00] Robert E Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine learning*, 39(2-3):135–168, 2000.

[SVS$^+$10] Leander Schietgat, Celine Vens, Jan Struyf, Hendrik Blockeel, Dragi Kocev, and Sašo Džeroski. Predicting gene function using hierarchical multi-label decision tree ensembles. *BMC bioinformatics*, 11(1):1, 2010.

[TK06] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *Dept. of Informatics, Aristotle University of Thessaloniki, Greece*, 2006.

[TKV09] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Mining multi-label data. In *Data mining and knowledge discovery handbook*, pages 667–685. Springer, 2009.

[TLP+12] Liang Tang, Tao Li, Florian Pinel, Larisa Shwartz, and Genady Grabarnik. Optimizing system monitoring configurations for non-actionable alerts. In *Proceedings of IEEE/IFIP Network Operations and Management Symposium*, 2012.

[TLS12] Liang Tang, Tao Li, and Larisa Shwartz. Discovering lag intervals for temporal dependencies. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 633–641. ACM, 2012.

[TLS+13] Liang Tang, Tao Li, Larisa Shwartz, Florian Pinel, and Genady Ya Grabarnik. An integrated framework for optimizing automatic monitoring systems in large IT infrastructures. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1249–1257. ACM, 2013.

[TLSG13] Liang Tang, Tao Li, Larisa Shwartz, and Genady Grabarnik. Recommending Resolutions for Problems Identified by Monitoring. pages 134–142, 2013.

[TV07] Grigorios Tsoumakas and Ioannis Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *European Conference on Machine Learning*, pages 406–417. Springer, 2007.

[urla] IBM Tivoli: Integrated Service Management software. `http://www-01.ibm.com/software/tivoli/`.

[urlb] ITIL. `http://www.itil-officialsite.com/`.

[url15] libSVM, 2015. `https://www.csie.ntu.edu.tw/~cjlin/libsvm/`.

[US02] Naonori Ueda and Kazumi Saito. Parametric mixture models for multi-labeled text. In *Advances in neural information processing systems*, pages 721–728, 2002.

[Vap13] Vladimir Vapnik. *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[Vat12] Peerapon Vateekul. Hierarchical multi-label classification: Going beyond generalization trees. 2012.

[VM02] Ricardo Vilalta and Sheng Ma. Predicting rare events in temporal domains. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 474–481. IEEE, 2002.

[VSS$^+$08] Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. Decision trees for hierarchical multi-label classification. *Machine Learning*, 73(2):185–214, 2008.

[WDH10] Hua Wang, Chris HQ Ding, and Heng Huang. Multi-label classification: Inconsistency and class balanced k-nearest neighbor. In *AAAI*, 2010.

[WDR06] Eugene Wu, Yanlei Diao, and Shariq Rizvi. High-performance complex event processing over streams. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 407–418. ACM, 2006.

[WIK16] WIKI, 2016. https://en.wikipedia.org/wiki/Network-attached_storage.

[WJ12] Bi Wei and T. Kwok James. Hierarchical multilabel classification with minimum bayes risk. In *Proceedings of the 12th International Conference on Data Mining*. IEEE, 2012.

[WL04] Wagner Rodrigo Weinert and Heitor Silvério Lopes. Neural networks for protein classification. *Applied Bioinformatics*, 3(1):41–48, 2004.

[WWP99] Andreas S Weigend, Erik D Wiener, and Jan O Pedersen. Exploiting hierarchy in text categorization. *Information Retrieval*, 1(3):193–216, 1999.

[YADS11] Zoulficar Younes, Fahed Abdallah, Thierry Denoeux, and Hichem Snoussi. A dependent multilabel classification method derived from the k-nearest neighbor rule. *EURASIP Journal on Advances in Signal Processing*, 2011(1):1–14, 2011.

[Yan95] Yiming Yang. Noise reduction in a statistical approach to text categorization. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 256–263. ACM, 1995.

[YAP00] Yiming Yang, Tom Ault, and Thomas Pierce. Combining multiple learning strategies for effective cross validation. In *ICML*, pages 1167–1174, 2000.

[YC94] Yiming Yang and Christopher G Chute. An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems (TOIS)*, 12(3):252–277, 1994.

[YL99] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49. ACM, 1999.

[YP97] Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *ICML*, volume 97, pages 412–420, 1997.

[YTS07] Rong Yan, Jelena Tesic, and John R Smith. Model-shared subspace boosting for multi-label classification. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 834–843. ACM, 2007.

[Zak01] Mohammed J Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine learning*, 42(1-2):31–60, 2001.

[zen16] zenoss. zenoss. `http://ownit.zenoss.com/`, 2016.

[ZF09] Cunlu Zou and Jianfeng Feng. Granger causality vs. dynamic bayesian network inference: a comparative study. *BMC bioinformatics*, 10(1):1, 2009.

[ZJXG05] Shenghuo Zhu, Xiang Ji, Wei Xu, and Yihong Gong. Multi-labelled classification using maximum entropy method. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 274–281. ACM, 2005.

[ZJZ$^+$13a] Chunqiu Zeng, Yexi Jiang, Li Zheng, Jingxuan Li, Lei Li, Hongtai Li, Chao Shen, Wubai Zhou, Tao Li, Bing Duan, et al. Fiu-miner: a fast, integrated, and user-friendly system for data mining in distributed environment. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1506–1509. ACM, 2013.

[ZJZ$^+$13b] Chunqiu Zeng, Yexi Jiang, Li Zheng, Jingxuan Li, Lei Li, Hongtai Li, Chao Shen, Wubai Zhou, Tao Li, Bing Duan, Ming Lei, and Pengnian Wang. FIU-Miner: A Fast, Integrated, and User-Friendly System for Data Mining in Distributed Environment. In *Proceedings of the Nineteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2013.

[ZL15] Chunqiu Zeng and Tao Li. Event pattern mining. In Tao Li, editor, *Event Mining: Algorithms and Applications*, pages 71–121. Chapman and Hall/CRC, 2015.

[ZLSG14a] Chunqiu Zeng, Tao Li, Larisa Shwartz, and Genady Ya Grabarnik. Hierarchical multi-label classification over ticket data using contextual loss. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*, pages 1–8. IEEE, 2014.

[ZLSG14b] Chunqiu Zeng, Tao Li, Larisa Shwartz, and Genady Ya Grabarnik. Hierarchical multi-label classification over ticket data using contextual loss. In *Proceedings of IEEE/IFIP Network Operations and Management Symposium*, pages 1–8. IEEE, 2014.

[ZTL$^+$14] Chunqiu Zeng, Liang Tang, Tao Li, Larisa Shwartz, and Genady Ya Grabarnik. Mining temporal lag from fluctuating events for correlation and root cause analysis. In *Network and Service Management (CNSM), 2014 10th International Conference on*, pages 19–27. IEEE, 2014.

[ZU99] Detlef Zimmer and Rainer Unland. On the semantics of complex events in active database management systems. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 392–399. IEEE, 1999.

[ZWML16] Chunqiu Zeng, Qing Wang, Shekoofeh Mokhtari, and Tao Li. Online context-aware recommendation with time varying multi-armed bandit. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 2025–2034, New York, NY, USA, 2016. ACM.

[ZZ06] Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006.

[ZZ07] Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048, 2007.

[ZZ14] Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837, 2014.

VITA

CHUNQIU ZENG

2012-Present	Ph.D., Computer Science
	Florida International University, Miami, Florida
2009-2012	Senior Data Engineer
	Alibaba Company, Hangzhou, P.R. China
2009	M.S. and B.S., Computer Science
	Sichuan University, Chengdu, P.R. China

PUBLICATIONS

Chunqiu Zeng, Qing Wang, Wentao Wang, Tao Li, Larisa Shwartz, *Online Inference for Time-Varying Temporal Dependency Discovery from Time Series*, in *Proceedings of the 4th annual IEEE International Conference on Big Data(IEEE Big Data)*, 2016.

Chunqiu Zeng, Liang Tang, Wubai Zhou, Tao Li, Larisa Shwartz, Genady Ya.Grabarnik, *An Integrated Framework for Mining Temporal Logs from Fluctuating Events*, in *IEEE Transactions on Services Computing(TSC)*, 2016.

Tao Li, Wubai Zhou, Chunqiu Zeng, Qing Wang, Qifeng Zhou, Dingding Wang, Jia X-u, Yue Huang, Wentao Wang, Minjing Zhang, Steve Luis, Shu-Ching Chen, Naphtali Rishe, *DI-DAP: An Efficient Disaster Information Delivery and Analysis Platform in Disaster Management*, in *Proceedings of the 25th ACM Conference on Information and Knowledge Management (CIKM)*, Indianapolis, US, Oct.2016.

Wubai Zhou , Liang Tang, Chunqiu Zeng, Tao Li, Larisa Shwartz, Genady Ya.Grabarnik, *Resolution Recommendation for Event Tickets in Service Management*, in *IEEE Transactions on Network and Service Management(TNSM)*, 2016.

Chunqiu Zeng, Qing Wang, Shekoofeh Mokhtari, Tao Li, *Online Context-Aware Recommendation with Time Varying Multi-Armed Bandit*, in *Proceedings of the 22nd annual ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.

Tao Li, Ning Xie, Chunqiu Zeng, Wubai Zhou, Li Zheng, Yexi Jiang, Yimin Yang, Hsin-Yu Ha, Wei Xue, Chao Shen, Liang Tang, Lei Li, Shu-ching Chen, Jai Navlakha, and S.S. Iyengar, *Data-driven Techniques in Disaster Information Management*, in *ACM Computing Surveys*, 2016.

Tao Li, Chunqiu Zeng, Li Zheng, Lei Li, Yue Huang, Zheng Liu, Qifeng Zhou, Wubai Zhou, and Wei Xue, *FIU-Miner (A Fast, Integrated, and User-Friendly System for Data Mining) and Its Applications*, in *Knowledge and Information Systems(KAIS)*, 2016.

Jian Xu, Liang Tang, Chunqiu Zeng, Tao Li, *Pattern Discovery via Constraint Programming.* in *Knowledge-Based Systems*, vol.99, pages 23-32, 2016.

Jian Xu, Yexi Jiang, Chunqiu Zeng, Tao Li, *Node Anomaly Detection for HomoGeneous Distributed Environments*, in *Expert Systems with Applications (ESWA)*. 2015.

Liang Tang, Yexi Jiang, Lei Li, Chunqiu Zeng, Tao Li, *Personalized Recommendation via Parameter-Free Contextual Bandits*, in *Proceedings of the 38th Annual International ACM SIGIR Conference (SIGIR)*. 2015.

Chunqiu Zeng, Liang Tang, Tao Li, Larisa Shwartz, Genady Ya. Graharnik, *Mining Temporal Lag from Fluctuating Events for Correlation and Root Cause Analysis*, in *Proceedings of the 10th International Conference on Network and Service Management (CNSM)*. 2014.

Chunqiu Zeng, Hongtai Li, Huibo Wang, Yudong Guang, Chang Liu, Tao Li, Mingjin Zhang, Shu-Ching Chen, Naphtali Rishe, *Optimizing Online Spatial Data Analysis with Sequential Query Patterns*, in *Proceedings of the 15th IEEE international Conference on Information Integration and Reuse(IRI)*. 2014.

Lei Li, Chao Shen, Long Wang, Li Zheng, Yexi Jiang, Liang Tang, Hongtai Li, Longhui Zhang, Chunqiu Zeng, Tao Li, *iMiner: Mining Inventory Data for Intelligent Management*, in *Proceedings of the 23rd ACM Conference on Information and Knowledge Management (CIKM)*. 2014.

Yexi Jiang, Chunqiu Zeng, Jian Xu, Tao Li, *Real Time Contextual Collective Anomaly Detection over Multiple Data Streams*, in *ACM SIGKDD Workshop Outliner Detection & Description under Data Diverity(SIGKDD WORKSHOP ODD^2)*. 2014.

Li Zheng, Chunqiu Zeng, Lei Li, Yexi Jiang, Wei Xue, Jingxuan Li, Chao Shen, Wubai Zhou, Hongtai Li, Liang Tang, Tao Li, Bing Duan, Ming Lei, and Pengnian Wang, *Applying Data Mining Techniques to Address Critical Process Optimization Needs in Advanced Manufacturing*,in *Proceedings of the 20th ACM Conference on Knowledge Discovery and Data Mining (KDD)*. 2014.

Chunqiu Zeng, Tao Li, Larisa Shwartz, Genady Ya. Graharnik, *Hierarchical Multi-Label Classification over Ticket Data using Contextual Loss*, in *Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS)*. 2014.

Li Zheng, Chao Shen, Liang Tang, Chunqiu Zeng, Tao Li, Steve Luis, Shu-Ching Chen, *Data Mining Meets the Needs of Disaster Information Management*, in *IEEE Transactions on Human-Machine Systems (THMS)*. 2013.

Chunqiu Zeng, Yexi Jiang, Li Zheng, Jingxuan Li, Lei Li, Hongtai Li, Chao Shen, Wubai Zhou, Tao Li, Bing Duan, Ming Lei, Pengnian Wang, *FIU-Miner: A Fast, Integrated, User-Friendly System for Data Mining in Distributed Environment*, in *Proceedings of the 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. 2013.

Li Zheng, Chao Shen, Liang Tang, Chunqiu Zeng, Tao Li, Steve Luis, Shu-Ching Chen and Jainendra K. Navlakha, *Disaster SitRep - A Vertical Search Engine and Information Analysis Tool In Disaster Management Domain*, in *Proceedings of the 13th IEEE international Conference on Inofrmation Integration and Reuse(IRI)*. 2012.