BOOLEAN MATRIX DECOMPOSITION AND EXTENSION WITH APPLICATIONS

by
Haibing Lu

A dissertation submitted to the
Graduate school-Newark
Rutgers, The State University of New Jersey
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
Graduate Program in Management
Information Technology Major

Written under the direction of
Dr. Vijayalakshmi Atluri
Dr. Jaideep Vaidya
and approved by

_____

_____

_____

_____

_____

Newark, New Jersey
October 2011

DISSERTATION ABSTRACT

BOOLEAN MATRIX DECOMPOSITION AND EXTENSION WITH APPLICATIONS

By Haibing Lu

Dissertation Directors: Dr. Vijayalakshmi Atluri and Dr. Jaideep Vaidya

Boolean matrix decomposition (BMD) refers to decomposing of an input Boolean matrix into a product of two Boolean matrices, where the first matrix represents a set of meaningful concepts, and the second describes how the observed data can be expressed as combinations of those concepts. As opposed to standard matrix factorization, BMD focuses on Boolean data and employs Boolean matrix product instead of standard matrix product. The key advantage of BMD is that BMD solutions provide much more interpretability, which enable BMD to have wide applications in multiple domains, including role mining, text mining, discrete pattern mining, and many others.

There are three main challenges in the research of BMD. First, real applications carry varying expectations and constraints on BMD solutions, which make the task of searching for a good BMD solution nontrivial. Second, BMD by itself has the issue of insufficiency in modeling some real data semantics, as only the set union operation is employed in combination. Third, BMD variants are generally

NP-hard in nature, which makes practitioners reluctant to apply the BMD model to large scale data analysis.

All of the three challenges are addressed in this dissertation. First, a unified framework, which is based on integer linear programming, is presented to encompass all BMD variants. Such a framework allows us to directly adopt fruitful research results in the optimization field to solve our own problems. It also provides researchers across different domains with a new perspective to view their problems and enables them to share their research results. Second, a novel extended Boolean matrix decomposition (EBMD) model is proposed. It allows describing an observed record as an inclusion of some concepts with an exclusion of some other concepts. Thus EBMD is effective to meet the needs of modeling some complex data semantics. Third, rank-one BMD is studied. Rank-one BMD is to decompose a Boolean matrix into the product of two Boolean vectors, which can be interpreted as a dominant pattern vector and a presence vector. By recursively applying rank-one BMD, a Boolean matrix is partitioned into clusters and discrete patterns of the observed data are thus discovered. Rank-one BMD can serve many functions of regular BMD, while rank-one BMD is relatively easy to solve compared to regular BMD. In addition, efficient 2-approximation algorithms are found for some special cases of rank-one BMD.

PREFACE

Dissertation Committee Members

- Dr. Nabil Adam, Rutgers University
- Dr. Vijayalakshmi Atluri, Rutgers University
- Dr. Pierangela Samarati, University of Milan
- Dr. Jaideep Vaidya, Rutgers University
- Dr. Hui Xiong, Rutgers University

ACKNOWLEDGEMENTS

The writing of a dissertation for me is more like reading my autobiography. All memories in the past five years are surfacing. Looking back at my work, I am deeply grateful for all helps and supports I have received throughout the course of my doctorate study.

I would like to express my deepest gratitude to my esteemed advisors, Dr. Vijay Atluri and Dr. Jaideep Vaidya. Without their expert guidance and help, this dissertation would not have been possible and I would not have gone so far in academic research.

My thanks go out to Dr. Nabil Adam, Dr. Hui Xiong, and Dr. Pierangela Samarati for their perceptive comments on improving my dissertation work.

I also wish to thank all my friends and student-colleagues at Rutgers Business School for enriching my life and giving me joy.

Finally, I want to thank my wife and my parents who have always supported me and stood by me with their patience and encouragement. I will pay you for the rest of my life.

Thank you all!

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

Many kinds of real data sets can be represented by Boolean matrices, such as market basket data, document data, Web click-stream data and user-to-permission assignment data in an organization. For instance, market basket data contains customer transaction records, where each record can be represented as a Boolean vector where each element indicates whether or not the corresponding item/product is purchased. A document can be described by a Boolean vector where each element indicates whether or not a corresponding word/term is present.

Interestingly, many important data analysis tasks on Boolean data can be transformed as Boolean matrix decomposition (BMD) problems. BMD is to decompose a Boolean matrix $A_{m\times n}$ into two matrices $X_{m\times k}$ and $C_{k\times n}$, such that $A = X \bigotimes C$. In which, $\bigotimes$ is called Boolean matrix product and significantly different from the ordinary matrix product. $\bigotimes$ is built on the logical arithmetic operations, $\vee$ and $\wedge$. If $A = X \bigotimes C$, we have

$$a_{ij} = \bigvee_{l=1}^{k} (X_{it} \wedge C_{tj}) \ . \tag{1.1}$$

An input Boolean matrix $A_{m\times n}$ can be viewed as m data records with n attributes $\{1, 2, ..., n\}$. The ith row vector corresponds to an attribute subset A_i such

that A_i contains the attribute j if $A_{ij} = 1$.

$C_{k\times n}$ can be viewed as as a collection of k concepts, where each concept is a subset of attributes $\{1, 2, ..., n\}$. Concept i consists of attribute j if $C_{ij} = 1$.

$X_{m\times k}$ can be interpreted as a combination matrix, showing how each observed data record is represented as a union of a subset of concepts.

Then a BMD solution $A = X \otimes C$ can be interpreted as follows:

$$A_i = \bigcup_{x_{ij}=1} C_j, \forall j \ . \tag{1.2}$$

Such interpretability enables BMD to model many real data semantics, which cannot be found in an ordinary matrix factorization.

Take the role mining problem (RMP) as an example. RMP comes from the implementation of Role-based access control (RBAC). RBAC is a widely accepted access control model, which greatly simplifies administration by assigning each user a few roles instead of a large number of individual permissions, where a role is a subset of permissions. To take the advantages of RBAC, organizations need to define a good set of roles, and then assign them to users appropriately, the work of which is called role engineering. To automate the process of role engineering, Vaidya et al. [62] propose to mine roles from existing user-to-permission assignment data, which is the origin of RMP. Look at the example of user-to-permission assignment data represented by a bipartite graph as shown in Figure 1.1. With m permissions and n users, that bipartite graph can be represented by a Boolean matrix $A_{m\times n}$, where $a_{ij} = 1$ if the ith permission is assigned to the jth user,

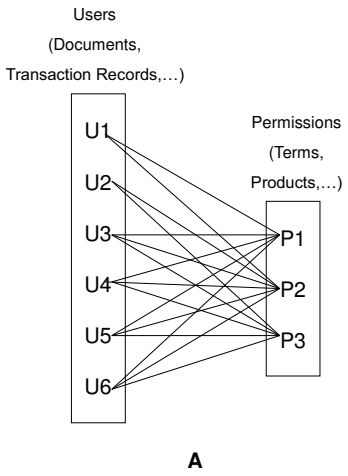


Figure 1.1. Bipartite Graph

otherwise 0. As we know, a role is nothing, but a subset of permissions. The basic objective of RMP is to find a set of roles and user-to-role assignments, such that every user has exactly the same permissions as what they had. To meet that requirement, the union of permissions contained by roles assigned to each user has to be the permission set that was assigned to that user. A feasible solution for the example of Figure 1.1 is as shown in Figure 1.2. In which, permission-to-role assignments and user-to-role assignments return two Boolean matrices C and X. Hence, that graph is corresponding to a BMD solution as $A = X \bigotimes C$. As we can see, role mining is essentially to finding a BMD solution with existing user-to-permission assignments as the input Boolean matrix. Besides RBAC, BMD can be applied to many other domains. For example, if the bipartite graph of Figure 1.1 is representing document-to-term data, the third layer of the tripartite graph of Figure 1.2 gives a set of topics extracted from existing document-to-term data . If the bipartite graph of Figure 1.1 describes a market basket data set, a BMD solution implied from Figure 1.2 generates a set of product itemsets, which

might be beneficial for supermarkets to design promotion strategies. There are prevalent applications of BMD in many domains involving Boolean data analysis tasks. However, there are three prominent problems existing in the research field of BMD as follows.

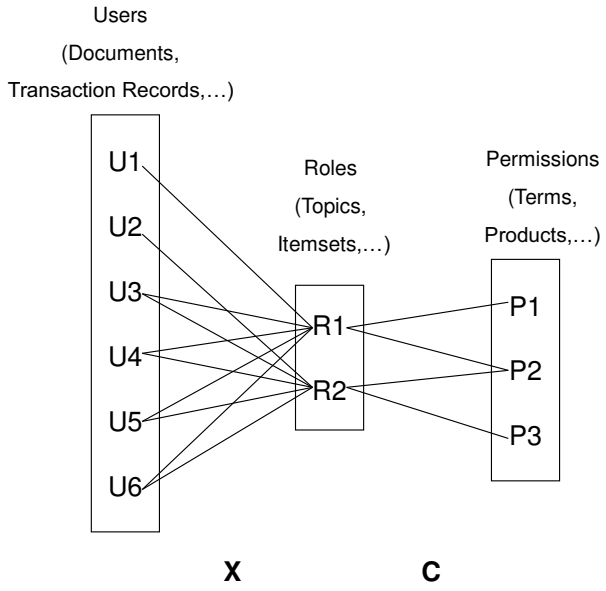


Figure 1.2. Tripartite Graph

1. **Lack of a General Framework.**

   Most Boolean data analysis tasks cannot be simply modeled as finding a feasible BMD solution. Usually specific objectives and constraints are attached. For example, to maximize the benefits of adopting the RBAC scheme, one possible way is to find a minimum set of roles. In the language of BMD, it is to find a feasible BMD solution, such that C is of the least rows. While, to minimize the RBAC administrative work, one needs to find a set of roles, which brings the least assignments. In other words, the decomposed matrices of the input user-to-permission assignment ma-

trix contain the least 1's cells. Besides various objectives, some constraints may be applied. For example, each user is limited to have up to k roles, or no pair of roles overlap more than t permissions. As we can see, even the role ming problem alone generates many variants of BMD. However, there are many other research problems that can be modeled through BMD, such as document summarization in text mining, tiling databases, market basket data analysis, Boolean data compressing, feature selection, dimensionality reduction, etc. We observe that despite in different domains, many problems are equivalent from the perspective of BMD. For instance, the basic RMP problem in role mining [60] and the minimum tiling problem in tiling databases [19] are the same. Unfortunately, these problems used to be studied in their own disciplinary. So to fully exploit the benefits of BMD and help people recognize its importance, we need a general framework, which enables to classify, model, and solve problems from different domains. So people across different domains can share their intelligence and collaborate closely.

2. **Insufficiency of BMD in Modeling Real Data Semantics.**

   The reason why BMD has broad applications is that its decomposition solutions provide such interpretabilites that each observed Boolean record is expressed as a union of a subset of concepts. A BMD solution not only identifies concepts, but also shows how to reconstruct observed data from those concepts. However, we notice that the Boolean matrix product only considers the union operation. In other words, a successful BMD gives a set of concepts and shows how every column of the input data can be expressed

as a union of some subset of those concepts. This way of modeling incompletely represents some real data semantics. Essentially, it ignores a critical component – the set difference operation: a column can be expressed as the combination of union of certain concepts as well as the exclusion of other topics.

To explain this explicitly, we look at a simple text mining example. One of main research topics in text mining is given a large number of documents to generate a few topics to summarize them, where each document can be represented by a Boolean matrix $A_{m\times n}$ with $a_{ij} = 1$ if the ith document contains the jth term, otherwise 0. Then a row A_i is corresponding to a subset of terms. Suppose a presidential speech covers all topics except "EDUCATION". With BMD, to describe that speech, we have to list all mentioned topics. However, if we create a topic called "ALL-TOPICS", that speech can simply be expressed as $ALL - TOPICS \backslash EDUCATION$. Another advantage of introducing the set difference operation is that we may be able to use fewer topics to describe the same documents. To take these advantages, we need to extend the conventional BMD model and come up with a more general model, which can represent both the set union operation and the set difference operation.

3. **Inability of Rank-One BMD to Impose Personal Preferences.**

   Mining discrete patterns in binary data is important for many data analysis tasks, such as data sampling, compression, and clustering. An example is that replacing individual records with their patterns would greatly reduce data size and simplify subsequent data analysis tasks. As a straightforward

approach, rank-one binary matrix decomposition has been actively studied recently for mining discrete patterns from binary data. It factorizes a binary matrix into the multiplication of one binary pattern vector and one binary presence vector, while minimizing mismatching entries. However, this approach suffers from two serious problems. First, if all records are replaced with their respective patterns, the noise could make as much as 50% in the resulting approximate data. This is because the approach simply assumes that a pattern is present in a record as long as their matching entries are more than their mismatching entries. Second, two error types, 1-becoming-0 and 0-becoming-1, are treated evenly, while in many application domains they are discriminated.

1.1 Problem Statement

The objective of this dissertation is to investigate methodologies to facilitate certain types of Boolean data analysis tasks, which can be viewed as variants of matrix decomposition. Specifically, we will address the following three research issues.

1. **Boolean Matrix Decomposition.**

   As Boolean matrix decomposition provides solutions of good interpretabilities, it has attracted increasing attention recently. However, as it is a relatively new topic, it has not received enough attention, despite its great potentiality of being applied to many research domains. The subsequence is that the BMD model is not well studied, and its importance and potential

applications are not recognized. To improve this fact, we will address the following sub-problems:

- reviewing important research problems that can be modeled through BMD, and identify their corresponding BMD variants, some typical problems of which will be collected for further extensive study;
- building a unified framework to encompass all BMD variants;
- studying computational complexity of those identified typical BMD variants;
- designing effective and efficient algorithms for those important BMD variants.

2. **Extended Boolean Matrix Decomposition.**

   Although BMD has lot of potential applications, it lacks a critical component in combination, the set difference operation. It makes BMD not sufficient to model certain semantics. To address it, we propose a new notion, extended Boolean matrix decomposition, which allows each observed data record to be expressed as an inclusion of a subset of concepts with an exclusion of another subset of concepts. The introduction of the set difference operation will not only correct the deficiency of BMD in modeling, but also enable us to describe observed Boolean data with fewer concepts in a more succinct way. For example, suppose a document-to-term data set is given as the matrix on the left side of Equation 1.3. With BMD, the least number of topics needed to describe those five documents is 3. However, by introducing the set difference operation, only two topics as shown in Figure 1.3

are necessary. Those five topics are consequently expressed as $D1 = T1$, $D2 = T2$, $D3 = T1 \cup T2$, $D4 = T1 \backslash T2$, and $D5 = T2 \backslash T1$. Those relationships can also be recorded in a way of Boolean matrix multiplication as Figure 1.3, where $\odot$ is called the extended boolean matrix product operator, and the cell of the combination matrix at $\{ij\}$ is 1, if the document D_i includes the topic T_j, otherwise 0.

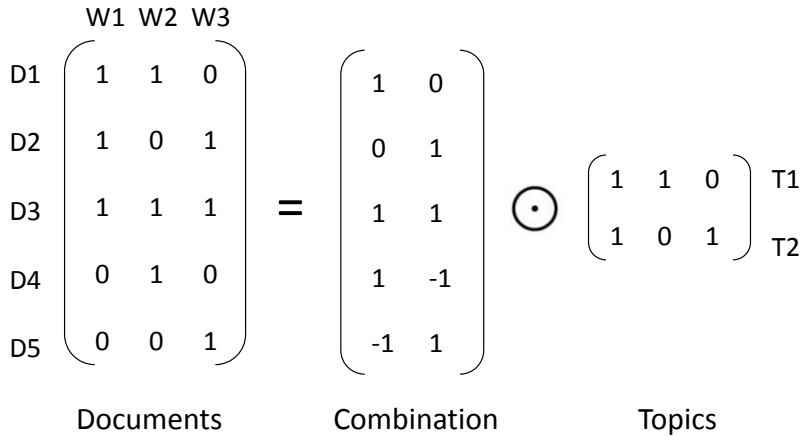


Figure 1.3. Illustration of Extended Boolean Matrix Factorization

EBMD also has a lot of implications in data compressing, role mining, text mining, knowledge discovery, etc. Different circumstances may carry varying objectives and requirements. To perform an extensive investigation on the EBMD model, we will address the following sub-problems:

- giving formal definition of EBMD and its operation rule;
- identifying applications that can benefit from EBMD and summarizing various EBMD variants;
- building a unified framework to encompass all EBMD variants;
- studying computational complexity of some typical EBMD variants;

- designing good algorithms for those EBMD variants.

3. **Weighted Rank-One Boolean Matrix Decomposition.**

Example 1.1 *Given a matrix A, a rank-one approximation is computed as follows:*

$$A = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix} \approx \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \end{pmatrix} = XY^T. \quad (1.3)$$

Rank-one BMD is a special variant of BMD. It is to decompose a Boolean matrix, where each row is an observed record, into the product of two Boolean vectors as illustrated in Equation 1.3. The decomposed Boolean row vector $(1, 1, 1, 1, 0)$ can be viewed as a dominant pattern of the observed data and the Boolean column vector $(1, 1, 0, 1)^T$, called a presence vector, shows which observed records belong to this dominant pattern. By its component values, the presence vector divides observed data records into two parts. By recursively applying rank-one BMD on every part, observed Boolean vectors can be divided into indivisible clusters. Dominant patterns for those clusters constitute discrete patterns of original data.

However, users are not able to impose their personal preferences on error type distribution of discovered patterns and the number of patterns. Look at the same example. What if one expects to discover more discrete patterns to improve the accuracy of patterns in approximating observed data? What if one wants to discover a set of patterns which can describe observed

data without introducing any 1-becoming-0 errors, because there are only a few 1's cells in the original data? To address these two issue, we propose weighted rank-one BMD, which is at the basis of conventional BMD to introduce different weights on 0-becoming-1 errors and 1-becoming-0 errors. To determine wether a data record belongs to a discrete pattern, three pieces of information need to be considered: (i) components of 1 in both the pattern and the data record, (ii) components which are 1 in the pattern and 0 in the data record; and (iii) components which are 0 in the pattern and 1 in the data record. By applying different weights to the three parts, instead of the same value as in conventional rank-one binary matrix approximation, users can effectively control the level of accuracy in the final approximate matrix and impose their preferences on the distribution of error types. Thus We call the problem weighted rank-one BMD.

1.2 Research Challenges

BMD is a relatively new research topic, but has received much attention recently from many research fields because of its good adaptability to real semantics. However, there are three main challenges remaining.

First, there lacks a general framework covering all BMD variants. It makes literature work not beneficial for solving other similar problems with only minor modifications. One of main reasons might be that people across different research fields have not seen their problems from the perspective of BMD. Actually, if they do, they would realize that their problems have much commonality. Hence, the first problem this dissertation deals with is to review all problems, which can be

modeled through BMD, and categorizes them into five main groups. To exploit the commonality of those problems, we will formulate them through mathematical programming. Once done that, for new problems with minor modifications from any of formulated problems, we simply change corresponding constraints or the objective function.

The second challenge is that the conventional BMD model does not consider the set difference operation, which makes it not be able to model the exclusion relationship, and limits the interpretabilities of decomposition solution. To address it, we propose EBMD, which allows both the set union operation and the set difference operation. Though it has broad applications, people may not realize it. Thus we will discuss all possible applications of EBMD and categorize them into groups as well. Furthermore, to exploit the commonality of those problems, we formulate them through integer programming (IP), which allows us to take advantages of available IP software packages and algorithms. As EBMD is a new notion, the computational complexity of its variants has never been studied. We will look at them also int this dissertation. Additionally, efficient heuristics will be designed for each EBMD variant.

The third challenge is that all problems to be studied are combinatorial problems in nature, which are usually hard to solve. The main application domains of our research are data mining and information security. Research problems occurring in those two domains usually involve large scale data, which requires us to give effective and efficient algorithms for the formulated combinational problems.

1.3 Contributions

There are four main contributions in this dissertation. First, the BMD problem is extensively investigated. Important BMD variants, which have pragmatic implications in reality, and their relations are identified and studied. Second, we propose the EBMD model, which addresses the inability of the BMD model to describe the set difference relationship. The proposed EBMD model also has the ability to discover some underlying data semantics along with the data mining process. Third, the weighted rank-one BMD model is proposed. It allows users to effectively impose their preferences on the number of mined discrete patterns and the error type distribution of resultant approximate data. Fourth, for each proposed problem, the computational complexity result is given, integer programming formulation is provided, and effective and efficient solutions, such as approximation algorithms and heuristics, are presented.

1.4 Outline of the Dissertation

The organization of the rest paper is as follows. Chapter 2 reviews the literature work related to Boolean matrix applications. Chapter 3 gives some background knowledge on computational complexity, mathematical programming, approximation algorithms, and heuristics. Chapters 4-6 study Boolean matrix decomposition, extended Boolean matrix decomposition, and weighted rank-one Boolean matrix decomposition respectively. Chapter 7 concludes our dissertation

CHAPTER 2
RELATED WORK

2.1 Role Engineering

Role engineering arises from implementing the RBAC system. RBAC is an access control system that users are assigned to roles instead to permissions. As the number of required roles is usually much less than the number of permissions, RBAC has the advantage of administrative efficiency over the conventional permission-based access control. Due to its advantage, many organizations want to transfer from their old access control systems to the RBAC system. To realize the full potential of RBAC, one must first define a complete and correct set of roles. According to a study by NIST, this task has been identified as the costliest component in realizing RBAC. The concept of role engineering was first presented by Coyne [11]. It refers to the systematic work for determining roles. Conceptually, there are two types of approaches towards role engineering. They are top-down and bottom-up. Top-down is by analyzing business processes to deduce roles [1, 2, 11]. However, all those approaches share one common weakness that it ignores existing user-to-permission assignments and calls for the cooperation among various authorities from different disciplines. The bottom-up approach generates roles purely from the existing user-to-permission assignments. It allows the automation of role generation without knowing the semantics of busi-

ness. Kuhlmann, Shohat, and Schmipf [47] present a bottom-up approach using clustering technique similar to the well known k-means clustering. Schlegelmilch and Steffens [58] have proposed an agglomerative clustering based role mining approach, known as ORCA. More recently, Vaidya et al. [62] proposed an approach based on subset enumeration, called RoleMiner. This approach not only eases the task of role engineering, but also helps in providing the security administrators an insight into user-to-role assignments. However, it does require an expert review of the results to choose which of the discovered roles are most advantageous to implement. To find the optimal role set matching the interestingness measures , Vaidya et al. [60] took a step forward to propose the role mining problem. Lu et al. [42] further connect the role mining problem with the Boolean matrix decomposition problem.

2.2 Text Mining

Text mining, roughly equivalent to text analytic, refers generally to the process of deriving high-quality information from text. High-quality information is typically derived through the deriving of patterns and trends through means such as statistical pattern learning. Text mining usually involves the process of structuring the input text, deriving patterns within the structured data, and finally evaluation and interpretation of the output. High quality in text mining usually refers to some combination of relevance, novelty, and interestingness. Typical text mining tasks include text categorization, text clustering, concept/entity extraction, production of granular taxonomies, sentiment analysis, document summarization, and entity relation modeling (i.e., learning relations between named entities) [30]. For in-

stance, data summarization is becoming an very important research topic, because with the widespread of internet technology we are facing the dramatically increasing amount of document data. Its basic task is to categorize a large amount of documents into some topics, where each topic could simply be a subset of words or a representative document. A good summarization scheme not only reduces data storage space, but also facilitates the task of information retrieval [5].

2.3 Ordinary Matrix Factorization

Ordinary matrix decomposition is a well-studied problem that has been the focus of significant research. Indeed, one of the earliest motivations of matrix decomposition came from the problem of solving linear equations. It is known that if a matrix A can be decomposed into the product of a lower triangular matrix L and a upper triangular matrix U, solving the systems $L(UX) = b$ and $UX = L^{-1}U$ is much easier than $AX = b$ [13]. In recent years, a big motivation for matrix decomposition is for data analysis and data processing. One of best known methods is perhaps the Singular Value Decomposition, $X = U \sum V$, where U and V are orthogonal real-valued matrices containing the left and right singular vectors of A, and $\sum$ is a diagonal matrix containing the singular values of A [13]. One classic application of this method is to get the optimal rank-k factorization of A by setting all but the top k singular values in $\sum$ to 0. In this sense, matrix decomposition can be used for compressing data. The underlying reason is that if we find $A_{m\times n} = X_{m\times k} \cdot C_{k\times n}$ and k is much less than m and n, storing X and C instead of A will save great space [29].

2.4 Nonnegative Matrix Factorization

While the SVD is optimal in terms of the Frobenius norm, recently, people have realized that it does not have sufficient interpretability. To address this problem, multiple new methods were proposed, like Probabilistic Latent Semantic Indexing [27], Latent Dirichlet Allocation [9] and Nonnegative Matrix Factorization (NMF) [31]. NMF is also an old problem that has been extensively studied in [55]. In NMF, the added restriction is that all the matrices should be non-negative. This can help cluster data, find centroids and even describe the probabilistic relationships between individual points and centroids. Ding et al. [15] show the equivalence of NMF, spectral clustering and K-means clustering. The work of Lee and Seung [38, 39] also helped bring much attention from machine learning and data mining research communities to NMF.

2.5 Boolean Matrix Factorization

Since many real applications involve Boolean data, such as document-to-term data, web click-stream data (users vs websites), DNA microarray expression profiles and protein-protein complex interaction network [63], Boolean data have obtained a special and important space in the domain of data analysis [40]. It is natural to represent Boolean data by Boolean matrices, which are a special case of non-negative matrices. Many research problems involved in Boolean data analysis can be reduced to Boolean matrix factorization. Geerts et al. [19] propose the tiling databases problem which aims to find a set of tiles to cover a 0-1 database. Since a tile can be represented by a Boolean vector, the tiling databases problem is reduced to finding a factorization of $A = C \bigotimes X$ by limiting each column of

C to be a subset of one column of A, because each tile can only cover cells of 1. Miettinen et al. [52] consider C as a discrete basis of A, from which A can be reconstructed. Given A, how to find a good basis is their core problem. This work is further developed by Miettinen [48] where C is limited to be the subset of columns of A. This limitation gives increased interpretability since each column of C can be seen as a centroid of A from the perspective of clustering. [48] also allows the factorization $C \bigotimes X$ to cover cells containing zeros in A as long as the amount of error is within a tolerable threshold. Haibing et al. [42] looks at the Boolean matrix factorization problem in the context of role based access control (RBAC). The first and most difficult step of implementing RBAC is mining roles given the user-permission assignment. By representing the user-permission assignment, the user-role assignment and the mined role set by Boolean matrices A, C and X, we have $A = C \bigotimes A$. Therefore, the role mining problem is to find a Boolean matrix factorization of A.

2.6 Probabilistic Matrix Factorization

Probabilistic matrix factorization is viewing the input matrix from the statistic perspective. The most famous work might be principle component analysis (PCA) [3]. It transforms a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components. PCA involves the calculation of the eigenvalue decomposition of a data covariance matrix or singular value decomposition of a data matrix. Some work were proposed to use matrix factorization to model user rating profiles for collaborative filtering [45, 46]. Their core idea is to find a good factorization of the input matrix to reconstruct the missing

cells in the input data. In [8, 14, 27], they use the concept of matrix factorization to find latent factors by which to index documents. It can be applied in clustering as well. Some work even tried clustering two attributes of the input data simultaneously, by factoring a matrix into the product of three matrices [17, 44, 51].

CHAPTER 3
BACKGROUND

3.1 Access Control

In computer system security, role-based access control (RBAC) is an approach to restricting system access to authorized users. Due to its advantages in administrative efficiency and flexibility, it has been used by the majority of enterprisers and is a newer alternative approach to mandatory access control (MAC) and discretionary access control (DAC).

Sandu R. et al. [57] is one of the most cited papers in the field of RBAC. It defined $RBAC_0$, the basic model, $RBAC_1$ which introduces role hierarchies, $RBAC_2$ which introduces constraints at the basis of $RBAC_0$, and $RBAC_3$ which includes both role hierarchies and constrains. Their detailed descriptions are given as follows.

Definition 3.1 *($RBAC_0$)*

- *U, R, P and S (users, roles, permissions and sessions, respectively);*
- *$PA \subseteq P \times R$, a many-to-many permian-to-role assignment relation;*
- *$UA \subseteq U \times R$, a many-to-many user-to-role assignment relation;*

- *user :* $S \rightarrow U$*, a function mapping each session* s_i *to the single user user(*s_i*) (constant for the session's lifetime);*

- *roles :* $S \rightarrow 2^R$*, a function mapping each session* s_i *to a set of roes* $roles(s_i) \subseteq \{r|(user(s_i), r) \in UA\}$ *(which can change with time) and session* s_i *has the permissions* $U_r \in roles(s_i)\{p|(p, r) \in PA\}$.

The base model consists of everything except role hierarchies and constraints.

Definition 3.2 *The* $RBAC_1$ *model has the following components:*

- *U, R, P, S, PA, UA, and user are unchanged from* $RBAC_0$*;*

- $RH \subseteq P \times R$ *is a partial order on* R *called the role hierarchy or role dominance relation, also written as* $\geq$*; and;*

- $roles : S \rightarrow 2^n$ *is modified from* $RBAC_0$ *to require* $roles(s_i) \subseteq \{r|(\exists r' \geq r)[(usr(s_i), r') \in UA\}$ *(which can change with time) and session* s_i *has the permissions* $\cup_{r \in roles(s_i)}\{p|(\exists r'' \leq r)[(p, r'') \in PA]\}$.

Definition 3.3 $RBAC_2$ *is unchanged from* $RBAC_0$ *except for requiring that there be constraints to determine the acceptability of various components of* $RBAC_0$*. Only acceptable values will be permitted.*

Common access control constraints include mutually exclusive roles, cardinality, and prerequisite roles.

Mutually exclusive roles. The most common RBAC constraint could be mutually exclusive roles. The same user can be assigned to at most one role in a

mutually exclusive set. This supports separation of duties, which is further ensured by a mutual exclusion constraint on permission assignments.

Cardinality. Another user assignment constraint is a maximum number of members in a role. Only one person can fill the role of department chair; Similarly, the number of roles an individual user can belong to could also be limited. These are cardinality constraints, which can be correspondingly applied to permission assignments to control the distribution of powerful permissions. Minimum cardinality constraint, on the other hand, may be difficult to implement. For example, if a role requires a minimum number of members, it would be difficult for the system to know if one of the members disappeared and to respond appropriately.

Prerequisite roles. The concept of prerequisite roles is based on competency and appropriateness, whereby a user can be assigned to role A only if the user already is assigned to role B. For example, only users who are already assigned to the project role can be assigned to the testing role in that project. The prerequisite (project) role is junior to the new (test) role. In practice, prerequisites between incomparable roles are less likely to occur.

Other Constraints Constraints also apply to sessions and other user and role functions associated with a session. A user may belong to two roles but cannot be active in both at the same time. Other session constraints limit the number of sessions a user can have active at the same time. Correspondingly, the number of sessions to which a permission is assigned can be limited.

Definition 3.4 *$RBAC_3$ provides both role hierarchies and constraints as it combines $RBAC_1$ and $RBAC_2$.*

3.2 Computational Complexity

Computational complexity measures how difficult it is to solve a problem. In this dissertation, we are dealing with many discrete optimization problems. To gain an insight into the difficulty of those problems, performing computational complexity analysis is necessary. The book [18] provides a perfect guide to the computational complexity theory.

Definition 3.5 *P, also known as PTIME or DTIME, is one of the most fundamental complexity classes. It contains all decision problems which can be solved by a deterministic Turing machine using a polynomial amount of computation time, or polynomial time.*

People commonly think that P is the class of computational problems which are ”efficiently solvable” or ”tractable”. In practice, some problems not known to be in P have practical solutions, and some that are in P do not. But this is a useful rule of thumb.

Definition 3.6 *NP is the set of decision problems where the ”yes”-instance can be recognized in polynomial time by a non-deterministic Turing machine.*

Intuitively, NP is the set of all decision problems for which the instances where the answer is ”yes” have efficiently verifiable proofs of the fact that the answer is indeed ”yes”. The complexity class P is contained in NP, but NP contains many important problems, the hardest of which are called NP-complete problems, for which no polynomial-time algorithms are known.

Definition 3.7 *The complexity class NP-complete (abbreviated NP-C or NPC) is a class of decision problems. A problem L is NP-complete if it has two properties:*

- *It is in the set of NP (nondeterministic polynomial time) problems: Any given solution to L can be verified quickly (in polynomial time).*
- *It is also in the set of NP-hard problems: Any NP problem can be converted into L by a transformation of the inputs in polynomial time.*

NP-complete is a subset of NP, the set of all decision problems whose solutions can be verified in polynomial time; NP may be equivalently defined as the set of decision problems that can be solved in polynomial time on a nondeterministic Turing machine.

If a problem is NP-complete, it implies that most likely there is no polynomial algorithm. Then it is better to resort to other solutions, such as approximation algorithms and heuristics. Many problems have been proven to be NP-complete. A typical routine to prove a problem $\prod$ is NP-complete consists of the following steps:

- showing that $\prod$ is in NP;
- selecting a known NP-complete problem $\prod'$;
- constructing a transformation f from $\prod'$ to $\prod$, and
- proving that f is a polynomial transformation.

Definition 3.8 *A problem H is NP-hard if and only if there is an NP-complete problem L that is polynomial time Turing-reducible to H.*

Informally, NP-hard means ”at least as hard as the hardest problems in NP”. NP-hard problems may be of any type: decision problems, search problems, or optimization problems.

3.3 Approximation Algorithm

In computer science and operations research, approximation algorithms are algorithms used to find approximate solutions to optimization problems. Approximation algorithms are often associated with NP-hard problems; since it is unlikely that there can ever be efficient polynomial-time exact algorithms solving NP-hard problems, one settles for polynomial time sub-optimal solutions. Unlike heuristics, which usually only find reasonably good solutions reasonably fast, one wants provable solution quality and provable run time bounds. Ideally, the approximation is optimal up to a small constant factor. Approximation algorithms are increasingly being used for problems where exact polynomial-time algorithms are known but are too expensive due to the input size.

Before giving the formal definition of approximation algorithm, we define combinatorial optimization problem. A combinatorial optimization problem $\prod$ is either a minimization problem or a maximization problem and consists of the following three parts:

- a set $D_{\prod}$ of instances;
- for each instance $I \in D_{\prod}$, a finite set $S_{\prod}(I)$ of candidate solutions for I; and
- a function $m_{\prod}$ that assigns to each instance $I \in D_{\prod}$ and each candidate

solution $\sigma \in S_{\prod}(I)$ a positive rational number $m_{\prod}(I, \sigma)$, called the solution value for σ.

If $\prod$ is a minimization (problem) problem, then an optimal solution for an instance $I \in D_{prod}$ is a candidate solution $\sigma^* \in S_{\prod}(I)$ such that, for all $\sigma \in S_{\prod}(I)$, $m_{\prod}(I, \sigma^*) \leq m_{\prod}(I, \sigma)$ ($m_{\prod}(I, \sigma^*) \geq m_{\prod}(I, \sigma)$).

Definition 3.9 *An algorithm $\mathcal{A}$ is a ρ-approximation algorithm of the optimization problem $\prod$ if the value $f(x)$ of the approximation solution $\mathcal{A}(x)$ to any instance x of $\prod$, is not more than a factor ρ times the value, OPT, of an optimum solution.*

$$\begin{cases} OPT \leq f(x) \leq \rho OPT, & if \rho > 1 \\ \rho OPT \leq f(x) \leq OPT, & if \rho < 1. \end{cases} \tag{3.1}$$

Definition 3.10 *A family of approximation algorithms for a problem $\mathcal{P}$, $\{\mathcal{A}_\epsilon\}_\epsilon$, is called a polynomial approximation scheme or PAS, if algorithm $\mathcal{A}_\epsilon$ is a $(1+\epsilon)$-approximation algorithm and its running time is polynomial in the size of the input for a fixed ϵ.*

Definition 3.11 *A family of approximation algorithms for a problem $\mathcal{P}$, $\{\mathcal{A}_\epsilon\}_\epsilon$, is called a fully polynomial approximation scheme or FPAS, if algorithm $\mathcal{A}_\epsilon$ is a $(1+\epsilon)$-approximation algorithm and its running time is polynomial in the size of the input and $1/\epsilon$.*

When a FPAS is a family of randomized algorithms, it will be called fully polynomial randomized approximation scheme or FPRAS.

3.4 Mathematical Programming

Mathematical programming or optimization refers to choosing the best element from some set of available alternatives.

In the simplest case, this means solving problems in which one seeks to minimize or maximize a real function by systematically choosing the values of real or integer variables from within an allowed set. This formulation, using a scalar, real-valued objective function, is probably the simplest example; the generalization of optimization theory and techniques to other formulations comprises a large area of applied mathematics. More generally, it means finding ”best available” values of some objective function given a defined domain, including a variety of different types of objective functions and different types of domains.

Linear programming (LP), is a special type of convex programming, studies the case in which the objective function is linear and the set of constraints is specified using only linear equalities and inequalities. Such a set is called a polyhedron or a polytope if it is bounded. Linear programs can be expressed in the following form.

$$maximize(minimize)\ \ c^T x \tag{3.2}$$

$$subject\ to\ \ Ax \leq b \tag{3.3}$$

There are several good algorithms for linear program. The simplex algorithm [12], developed by George Dantzig in 1947, solves LP problems by constructing a feasible solution at a vertex of the polytope and then walking along

a path on the edges of the polytope to vertices with non-decreasing values of the objective function until an optimum is reached. In practice, the simplex algorithm is quite efficient and can be guaranteed to find the global optimum if certain precautions against cycling are taken. However, the simplex algorithm has poor worst-case behavior. Leonid Khachiyan in 1979 introduced the ellipsoid method [12], the first worst-case polynomial-time algorithm for linear programming. Khachiyan's algorithm was of landmark importance for establishing the polynomial-time solvability of linear programs. It also inspired new lines of research in linear programming with the development of interior point methods, which can be implemented as a practical tool. In contrast to the simplex algorithm, which finds the optimal solution by progressing along points on the boundary of a polytopal set, interior point methods move through the interior of the feasible region.

Integer programming studies linear programs in which some or all variables are constrained to take on integer values. This is not convex, and in general much more difficult than regular linear programming. Integer programming problems are in many practical situations (those with bounded variables) NP-hard. 0-1 integer programming or binary integer programming (BIP) is the special case of integer programming where variables are required to be 0 or 1. This problem is also classified as NP-hard, and in fact the decision version was one of Karp's 21 NP-complete problems. Advanced algorithms for solving integer linear programs include: cutting-plane method, branch and bound, branch and cut, and branch and price.

3.5 Heuristics

In computer science, heuristic designates a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. Heuristics make few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. However, heuristics do not guarantee an optimal solution is ever found. Many heuristics implement some form of stochastic optimization.

Heuristics are used for combinatorial optimization in which an optimal solution is sought over a discrete search space. Popular heuristics for combinatorial problems include simulated annealing by Kirkpatrick et al. [33], genetic algorithms by Holland et al. [28], ant colony optimization by Dorigo,[9] and tabu search by Glover [20].

CHAPTER 4
BOOLEAN MATRIX DECOMPOSITION

4.1 BMD Variants with Applications

4.1.1 Basic BMD

The basic BMD problem is to decompose an input Boolean matrix into two Boolean matrices with the minimum size. In other words, it is to find the most succinct representation of a Boolean matrix in the form of Boolean matrix decomposition. According to the rule of BMD, a Boolean matrix with the size of $m \times n$ can only be decomposed into two Boolean matrices with the sizes of $m \times k$ and $k \times n$. So to minimize the size of decomposed matrices is to minimize k, which gives the definition of the basic BMD problem as the following.

Problem 4.1 (Basic BMD) *Given a matrix $A \in \{0,1\}^{m \times n}$, find matrices $X \in \{0,1\}^{m \times k}$ and $C \in \{0,1\}^{k \times n}$, such that $A = X \bigotimes C$ and k is minimized.*

The basic BMD problem has pragmatic implications in many application domains, including role mining, tiling databases, graph theory and set theory. Many problems in those application domains can be formulated as the basic BMD problem with appropriate transformations. In the following, we will introduce some of them.

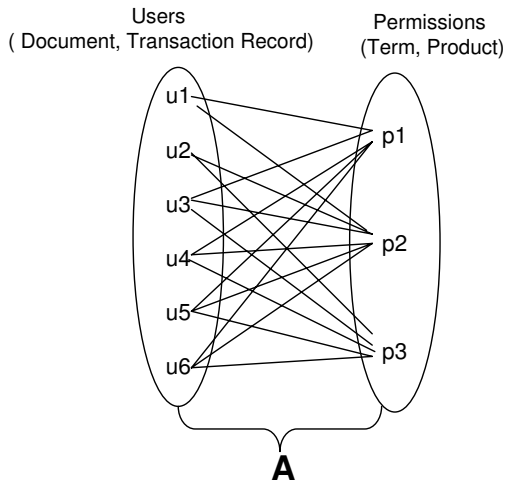


Figure 4.1. Bipartite Graph

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}. \tag{4.1}$$

Role Mining. The basic BMD problem can find its important application in role mining. As introduced in Chapter 2, the role mining problem arises from the implementation of a role-based access control system. It is to discover a good role set from user-to-permission assignments existing in an organization and assign these roles to users appropriately such that each user gets the same permissions as original. Role-based access control is usually administratively efficient compared to the traditional permission-based access control, especially for large-scale systems. It is due to the fact that the number of required roles is usually much less than the number of permissions.

The basic RMP problem, the fundamental role mining variant, attempts to find a minimum set of roles, which would maximize the benefits of RBAC. This prob-

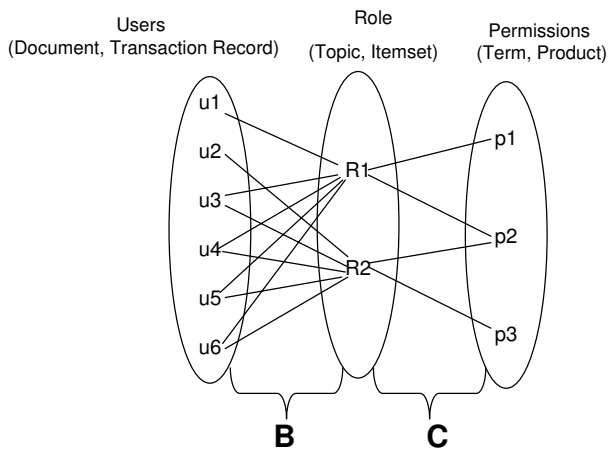


Figure 4.2. Tripartite Graph

lem is a basic BMD problem. Look at Figure 6.1a, an illustrative example of user-to-permission assignments. An edge means that its associated user is assigned its associated permission. Such user-to-permission assignments can be completely recorded by the Boolean matrix on the left to the equal sigh in Equation (4.1), where each row corresponds to a user, each column corresponds to a permission, and the value of 1 represents its corresponding user has its corresponding permission; otherwise, not. Similarly, the role mining solution as shown in Figure 5.5a can be mapped to the two Boolean matrices on the right to the equal sign in Equation (4.1). The first Boolean matrix on the right to the equal sign gives the same role assignments as shown in Figure 5.5a and the second Boolean matrix denotes two roles. So the basic RMP problem is equivalent to decomposing the Boolean matrix $A_{m\times}$ representing the user-to-permission assignments into two Boolean matrices $X_{m\times k}$ and $C_{k\times n}$, while minimizing k.

Market Basket Analysis. The basic BMD problem can be applied on market basket analysis as well. Market basket data contains the transactions on product

items, and is one of the main data types studied in the data mining research field. Market basket data are considered containing much information on customer behaviors and product values. The underlying patterns on market basket data are crucial to designing good commercializing strategies.

Market basket data can be represented in the form of a Boolean matrix with the cell at the position of $\{ij\}$ indicating whether or not the transaction i includes the item j. It can also be expressed as a bipartite graph as illustrated in Figure 6.1a, where nodes on the left side are transactions, nodes on the right side are items, and an edge means the transaction includes the item.

An important market basket data analysis task is to determine what items typically appear together, e.g., which items customers typically buy together in a database of supermarket transactions. This in turn gives insight into questions such as how to group them in store layout or product packages, how to market these products more effectively, or which items to offer on sale to increase the sale of other items.

Conventional solutions for determining such itemsets is based on the metric of support. The support of an itemset X is the ratio of transactions in which an itemset appears to the total number of transactions. Given a support threshold value, any itemset with a greater support value is considered to be frequent and selected.

However, this way suffers from some limitations. For a large-scale database a low support threshold would generate overwhelming itemsets, which are not of practical use for data owners. It is true that a high support threshold would reduce

selected itemsets significantly. However, such selected itemsets do not guarantee to cover the whole database or to describe all customer behaviors. In fact, in reality some itemsets might be less frequent, but quite important. A good itemset group is then expected to completely cover the whole database.

The basic BMD can be effectively utilized to resolve the itemset overwhelming issue. Instead of discovering frequent itemsets, we discover minimal itemsets to cover the whole transaction database. Such a solution can also be depicted by a tripartite graph as shown in 5.5a. The nodes in the middle denote itemsets. Edges between itemsets and items show the components of each itemset, while edges between transactions and itemsets gives the description for each transaction as a union of some itemsets. A tripartite graph is corresponding to two Boolean matrices. Hence, the task of minimizing the number of itemsets to describe a transaction database is a basic BMD problem.

Topic Identification

With the fast development of internet and database technologies, a huge amount of text data are generated and collected everyday, which creates needs for automated analysis. Given a collection of documents, a basic problem is: what topics are frequently discussed in the collection? Its answer would assist human understanding of the essence within documents and help in archiving and retrieving documents.

Given a collection of documents, a set of key words can be discovered. Each document then can be structured as a subset of keywords, which further can be represented as a Boolean row vector with each dimension corresponding to a keyword

and each component value indicating whether or not the keyword is included. As a result, a collection of documents can be represented as a Boolean matrix. It can be depicted as a bipartite graph as shown in Figure 4.5a as well.

The topic identification problem can be performed as follows: Discover minimal topics, each of which is a subset of keywords, to describe the given collection of documents. The solution would give a limited number of topics, which is a complete description of the whole collection of documents.

Such a topic identification problem is a basic BMD problem. Again, look at Figure 5.5a. The relation between documents, topics, and keywords is depicted as a tripartite graph, which decomposes the Boolean matrix of the document collection into two Boolean matrices.

4.1.2 Cost BMD

The cost BMD problem is to find a BMD solution minimizing the complexity of decomposition solution, which is the number of 1's cells in the decomposed matrices, instead of the number of roles.

Problem 4.2 (Cost BMD) *Given a Boolean matrix $A \in m \times n$, find Boolean matrices $X \in \{m \times k\}$ and $C \in \{k \times n\}$, such that $A = X \bigotimes C$ and $||X||_1 + ||C||_1$ is minimized.*

Cost BMD has an important application in role mining. The edge-RMP problem [60] is searching for the role set corresponding to the minimum administrative cost, which is quantified by the total number of 1's cells in the role-to-permission

assignment matrix C and the user-to-role assignment matrix X, as the management information system makes assignments only based on 1's cells.

Cost BMD can also be applied in Boolean data compression. As $A = X \bigotimes C$, storing A can be replaced by storing X and C, if the total size of X and C is much smaller than the size of A. To store a sparse Boolean matrix, only positions of 1's cells need to be maintained. Therefore, the size of a Boolean matrix is the number of its 1's cells. The problem of searching for the best compressing strategy for sparse matrices becomes a cost BMD problem.

4.1.3 Approximate BMD and Its Variants

The approximate BMD problem is to find a BMD solution without the restriction of exactness and is described as follows.

Problem 4.3 (Approximate BMD) *Given a Boolean matrix $A \in m \times n$ and a threshold value δ, find Boolean matrices $X \in \{m \times k\}$ and $C \in \{k \times n\}$, such that $||A - X \bigotimes C||_1 \leq \delta$ and k is minimized.*

An important motivation of approximate BMD is that in many cases a large number of concepts are required to exactly describe the observed Boolean data, while only a few concepts are necessary if a certain amount of inexactness is allowed. For those cases, if the exactness issue is not fatal, people tend to reduce the number of necessary concepts by introducing a limited amount of errors.

This model can be well applied to the text mining scenario, as for a large document-word data set, restricting each document be represented exactly by an

union of a subset of topics tends to cause the over-fitting problem. Hence it is better to allow some level of inexactness. It will facilitate the subsequent information retrieval task as well since less topics make indexing work easier. Approximate BMD can also be used to model the approximate RMP problem [62], which is a variant of basic RMP.

However, certain applications may have specific requirements on the characteristics of errors. It leads to two variants, 1-0 error free BMD and 0-1 error BMD, defined as follows.

Problem 4.4 (1-0 Error Free BMD) *Given a Boolean matrix $A \in m \times n$ and a threshold value δ, find Boolean matrices $X \in \{m \times k\}$ and $C \in \{k \times n\}$, such that $\sum_{ij} |a_{ij} - (X \otimes C)_{ij}|$ is minimized and $(X \otimes C)_{ij} = 1$ if $a_{ij} = 1$.*

Problem 4.5 (0-1 Error Free BMD) *Given a Boolean matrix $A \in m \times n$ and a threshold value δ, find Boolean matrices $C \in \{m \times k\}$ and $X \in \{k \times n\}$, such that $\sum_{ij} |a_{ij} - (X \otimes C)_{ij}|$ is minimized and $(X \otimes C)_{ij} = 0$ if $a_{ij} = 0$.*

For a sparse Boolean matrix with very few 1's cells, 1-0 errors would cause much information loss in the resultant Boolean matrix reconstructed from its approximate BMD solution. Hence, it is preferred to avoid 1-0 errors when decomposing sparse Boolean matrices, which gives rise to 1-0 error free BMD.

In the setting of role-based access control, 0-1 errors mean over-assignments. Over-assignments would cause serious security and safety problems because users may misuse permissions that are not supposed to be granted to them. Under-

assignments, which are 1-0 errors, are relatively more tolerable. Therefore, it is preferred to have 0-1 error free in the approximate RMP setting.

4.1.4 Partial BMD

Partial BMD is given concepts to describe observed data as combinations of concepts, formally defined as follows.

Problem 4.6 (Partial BMD) *Given matrices $A \in m \times n$ and $C \in \{k \times n\}$, find a Boolean matrix $X \in \{m \times k\}$, such that $\sum_{ij} |a_{ij} - (X \bigotimes C)_{ij}|$ is minimized.*

Partial BMD can be viewed as a subproblem of other BMD variants. For example, to solve the basic BMD problem, one two-phase approach is: (i) generate a candidate concept set; (2) examine how well the concept set describes the observed data. The second phase is a partial BMD problem.

Partial BMD can also rises on its own. The basis usage problem [52], which is given a set of Boolean basis vectors to describe an observed Boolean vector, is a partial BMD problem.

In addition to four BMD variants introduced above, there could be many other variants occurring in reality. For example, basic BMD may have the constraint that the combination of each observed vector is limited up to k concepts. But this dissertation focuses on those four typical BMD variants.

4.2 Theoretical Study

This section will give computational complexity results for basic BMD, approximate BMD, and partial BMD. The decision problem of basic BMD is a NP-

complete problem, which can be proven by a reduction to the set basis problem known to be NP-compete.

Definition 4.1 (Set Basis Problem) *INSTANCE: A finite set $\mathcal{U}$, a family $\mathcal{S} = \{S_1, ..., S_N\}$ of subsets of $\mathcal{U}$ and a positive integer k. QUESTION: Does there exist a set basis of size at most k for $\mathcal{S}$?*

Theorem 4.1 *The decision problem of basic BMD is NP-complete.*

Proof. The decision problem of basic BMD can be phrased as follows. INSTANCE: A Boolean matrix A and a positive integer k. QUESTION: Does there exist a BMD solution $X_{m\times k}$ and $C_{k\times n}$ of A?

For any set basis instance, we can find a basic BMD instance, which is true if and only if the set basis instance is true. For a set basis instance $\{\mathcal{U}, \mathcal{S}, k\}$, we create a vector sCce with dimensions of $|\mathcal{U}|$, which denotes the number of elements in $|\mathcal{U}|$, and each dimension corresponding to an element in $\{\mathcal{U}\}$. We also construct row vectors $\{A_1, ..., A_N\}$, such that $A_i(j) = 1$ if S_i contains element j. So far, we have created a basic BMD instance $\{A, k\}$. It is not difficult to see that the constructed instance is true if and only if the set basis instance is true. □

Basic BMD essentially is a special case of approximate BMD with the error threshold of 0. Hence, the decision problem of approximate BMD is NP-complete as well.

The decision problem of partial BMD is also NP-complete, which can be proven by a reduction to a known NP-complete problem, ±PSC [49] described as follows.

Problem 4.7 (±PSC) *INSTANCE: Two disjoint sets P and N of positive and negative elements, respectively, a collection $\mathcal{S}$ of subsets of $P \bigcup N$, and a positive integer t. QUESTION: Does there exist a subcollection $\mathcal{C} \in \mathcal{S}$, such that $|P \backslash (\cup \mathcal{C}|) + |N \cap (\cup \mathcal{C}|) \leq t$.*

Theorem 4.2 *The decision problem of partial BMD is NP-complete.*

Proof. The decision problem of partial BMD can be phrased as follows. INSTANCE: two Boolean matrices $A_{m \times n}$ and $C_{k \times n}$, and a positive number t. QUESTION: Does there exist a Boolean matrix X such that $\sum_{ij} |a_{ij} - (X \otimes C)_{ij}| \leq t$.

Given an instance of the decision problem of partial BMD, it is not difficult to check if it is true. So the decision problem of partial BMD belongs to NP.

For any ±PSC instance $\{P, N, \mathcal{S}, t\}$, we can construct a decision partial BMD instance as follows, which is true if and only if the ±PSC instance is true. We let A to a $1 \times (|P| + |N|)$ Boolean vector, with the first $|P|$ components being 1 and the others being 0. In addition, for the $1 \times (|P| + |N|)$ vector sCce, we let the first P dimensions correspond to positive elements in P respectively and the last N dimensions correspond to negative elements in N respectively. For each element subset s_i in $\mathcal{S}$, we create a Boolean row vector C_i of C such that: if s_i contains some element, the component of C_i which corresponds to that Crticular element is 1; otherwise 0. The constructed decision partial BMD instance $\{A_{1 \times (|P|+|N|)}, C_{|\mathcal{S}| \times (|P|+|N|)}, t\}$ is equivalent to the ±PSC instance. □

4.3 Mathematical Programming Formulation

BMD variants share much commonality. For example, the only difference between basic BMD and cost BMD is in the objective functions. The difference between approximate BMD and basic BMD is that approximate BMD allows inexactness. A natural arising thought is that if we can build a unified framework for all BMD variants, we then do not need to deal with each problem individually. Additionally when new BMD variants appear, system engineers do not need to start from scratch and can take advantage of algorithms that have been developed for existing BMD variants. As all BMD variants are essentially optimization problems, we propose to formulate them through integer linear programming.

There are many benefits by connecting BMD variants with integer linear programming. First, optimization has been studied for more than half century. There are quite a few good exact optimization algorithms, even for integer linear programming, such as branch-and-bound [37]. In addition, successful optimization software Packages are easily obtainable, such as Matlab and the Neos server [1]. Even though those BMD variants are proven to be hard to solve, small or medium size problems can still be solved through traditional optimization techniques. In addition to exact algorithms, approximation algorithms may be developed through LP-based techniques, such as dual-fitting [41], randomized rounding [23], and primal-dual schema [36]. The linear programming framework we will propose in fact can not only incorporate BMD variants, but also problems in other application domains, such as tiling database problems [19] and discrete basis problems [50].

[1] http://www-neos.mcs.anl.gov/

For ease of explaining, in this section, we will discuss BMD variants in the role mining context, where a BMD solution $\{X, C\}$ of user-to-permission assignments A gives roles C and user-to-role assignments X.

4.3.1 Partial BMD

In the context of role mining, partial BMD is given user-to-permission assignments A and roles C to assign roles appropriately to users such that the resultant user-to-permission assignment errors $\sum_{ij} |a_{ij} - (X \otimes C)_{ij}|$ is minimized. Then the partial BMD problem can be roughly represented as follows:

$$minimize\ ||A - X \otimes C||_1.$$

To formulate it as an explicit integer linear programming problem, we first formulate $X \otimes C = 0$ and then relax it by tolerating errors.

To do so, we let C_i denote role i and A_i denote permissions assigned to user i. $A - X \otimes C = 0$ means every user's permission set should be represented as a union of some candidate roles. This can be phrased as $A_i = \bigcup_{t \in s_i} C_t$, where s_i denotes the role subset assigned to user i. $\{s_i\}$ can convert to a Boolean matrix X such that $X_{it} = 1$ if role t belongs to s_i, otherwise $X_{it} = 0$.

The constraint essentially says that if some user has a particular permission, at least one role having that permission has to be assigned to that user. In turn, if that user does not have some permission, none of the roles having that permission can be assigned to it. So $X \otimes C = A$ can be transformed to the following equation set.

$$\begin{cases} \sum_{t=1}^{q} X_{it} C_{tj} \geq 1, & if\ A_{ij} = 1 \\ \sum_{t=1}^{q} X_{it} C_{tj} = 0, & if\ A_{ij} = 0 \end{cases} \tag{4.2}$$

To formulate inexactness, we introduce a non-negative slack variable $\{V_{ij}\}$ to each constraint and have the following modified constraints.

$$\begin{cases} \sum_{t=1}^{q} X_{it}C_{tj} + V_{ij} \geq 1, & if\ A_{ij} = 1 \\ \sum_{t=1}^{q} X_{it}C_{tj} - V_{ij} = 0, & if\ A_{ij} = 0 \end{cases} \tag{4.3}$$

For $A_{ij} = 1$, $\sum_{t=1}^{q} X_{it}C_{tj} + V_{ij} \geq 1$ allows $\sum_{t=1}^{q} X_{it}C_{tj}$ to be 0. For $A_{ij} = 0$, $\sum_{t=1}^{q} X_{it}C_{tj} - V_{ij} = 0$ allows $\sum_{t=1}^{q} X_{it}C_{tj}$ to be greater than 1. In other words, if $V_{ij} > 0$, the constraint enforcing whether $A_{ij} = 1$ or $A_{ij} = 0$ is not satisfied. The objective function $||A - X \otimes C||_1$ is then the count of unsatisfied constraints for $A = X \otimes C$. To formulate the objective function, we need to count the positive variables $\{V_{ij}\}$. To do so, we introduce another Boolean variable set $\{U_{it}\}$ and enforce the following constraints:

$$\begin{cases} MU_{ij} \geq V_{ij} \\ U_{ij} \leq V_{ij} \end{cases}.$$

In which, the big M is a large constant greater than q. The above two inequalities ensure that if $V_{ij} \geq 1$, $U_{ij} = 1$ and if $V_{ij} = 0$, $U_{ij} = 0$. Thus, the count of positive $\{V_{ij}\}$ is $\sum_{ij} U_{ij}$. Therefore, the integer linear programming formulation for partial BMD is as the following

$$minimize \sum_{ij} U_{ij}$$
$$\begin{cases} \sum_{t=1}^{q} X_{it}R_{tj} + V_{ij} \geq 1, if\ UC_{ij} = 1 \\ \sum_{t=1}^{q} X_{it}R_{tj} - V_{ij} = 0, if\ UC_{ij} = 0 \\ MU_{ij} - V_{ij} \geq 0, \forall i, j \\ U_{ij} \leq V_{ij}, \forall i, j \\ X_{it}, U_{ij} \in \{0, 1\}, V_{ij} \geq 0 \end{cases}. \tag{4.4}$$

4.3.2 Basic BMD

With the integer linear program formulation for partial BMD, it is easy to formulate other BMD variants. Consider basic BMD. In the role mining setting, it means given user-to-permission assignments $A_{m\times n}$ to find user-to-role assignments $X_{m\times k}$ and permission-to-role assignments $C_{k\times n}$. It can be succinctly put as an optimization problem as follows:

$$minimize\ k$$

$$s.t.\ X_{m\times k} \otimes C_{k\times n} = A_{m\times n}.$$

For simplicity, we break up the basic BMD into two subproblems: (i) find a candidate role set $\{R_1, R_2, ..., R_q\}$; (ii) locate a minimum candidate role subset to form A.

A role is nothing, but a permission subset. Given n permissions, there are 2^n possible roles. But most of them can be easily eliminated. For example, if none of users had both permissions i and j, any permission subset containing both permissions i and j can never be a candidate role. We will explicitly discuss the generation of candidate roles when we conduct experimental study on BMD variants later. Here assume we have already had a candidate role set $\{R_1, R_2, ..., R_q\}$. Then consider the second subproblem, which is similar to partial BMD.

Every user's permission set should be able to be represented as a union of some candidate roles. This can be phrased as the following: $A_i = \bigcup_{t\in s_i} R_t$, where s_i denotes the candidate role subset assigned to user i and A_i denotes the permission subset assigned to user i. $\{s_i\}$ can convert to a Boolean matrix X such that $X_{it} = 1$ if candidate role t belongs to s_i, otherwise $X_{it} = 0$. As both X and R

are Boolean matrices, their relation can be represented by the following Boolean matrix multiplication, $X_{n\times q} \otimes R_{q\times m} = A_{n\times n}$, where R and A are given and X is unknown. As X_{it} indicates whether R_j is assigned to user i, so if $\forall t\ X_{it} = 0$, it means that R_t is never employed. So the basic BMD is essentially to minimize the number of non-zero columns in X.

Same as the partial BMD, the constraint $X \otimes R = A$ can be enforced by

$$\begin{cases} \sum_{t=1}^{q} X_{it}R_{tj} \geq 1, & if\ A_{ij} = 1 \\ \sum_{t=1}^{q} X_{it}R_{tj} = 0, & if\ A_{ij} = 0 \end{cases}. \tag{4.5}$$

Now we need to count the number of non-zero columns in X. To do so, we introduce a set of Boolean slack variables $\{d_1, ..., d_q\}$, where $d_t = 1$ indicates role t is present, otherwise not. Then $|Roles| = \sum_{t=1}^{q} d_t$. Since d_t indicates the presence of a role, d_t should only be 1 when at least one user is assigned role t. Thus, $d_t = 1$ when at least one of $\{X_{1t}, ..., X_{mt}\}$ is 1. We can formulate this by adding in the constraints $\{d_t \geq X_{it},\ \forall i, t\}$.

Finally, putting every thing together, the basic BMD is formulated as the following

$$minimize \sum_{t} d_t$$
$$\begin{cases} \sum_{t=1}^{q} X_{it}R_{tj} \geq 1, if\ A_{ij} = 1 \\ \sum_{t=1}^{q} X_{it}R_{tj} = 0, if\ A_{ij} = 0 \\ d_t \geq X_{ij}, \qquad \forall i, j \\ d_t, X_{ij} \in \{0, 1\} \end{cases}. \tag{4.6}$$

4.3.3 Approximate BMD

In the role mining setting, approximate BMD problem means to minimize the number of required roles within a tolerable amount of errors. In Equation 4.4, the error amount is formulated as $\sum_{ij} U_{ij}$. In Equation 4.6, the number of roles is formulated as $\sum_t d_t$. By combining these two equation systems together, we easily obtain the formulation for the approximate BMD problem as Equation 4.7.

$$minimize \sum_t d_t$$
$$\begin{cases} \sum_{t=1}^{q} X_{it}R_{tj} + V_{ij} \geq 1, if\ A_{ij} = 1 \\ \sum_{t=1}^{q} X_{it}R_{tj} - V_{ij} = 0, if\ A_{ij} = 0 \\ MU_{ij} - V_{ij} \geq 0, \forall i, j \\ U_{ij} \leq V_{ij}, \forall i, j \\ d_t \geq X_{it}, \qquad \forall i, t \\ \sum_i \sum_j U_{ij} \leq \delta \\ d_j, X_{it}, U_{ij} \in \{0, 1\}, V_{ij} \geq 0 \end{cases} . \tag{4.7}$$

In which, $\sum_{i=1}^{n} \sum_{j=1}^{m} U_{ij} \leq \delta$ ensures the total deviating cells less than δ and the objective function $\sum_{t=1}^{q} d_t$ is the number of mined roles.

4.3.4 Cost BMD

Different from basic BMD, cost BMD is to minimize the administrative cost. As a RBAC system only needs to maintain explicit user-to-role assignments and explicit role-to-permission assignments, the administrative cost is hence evaluated by positive cells in the decomposed matrices X and C. By employing the same notations in the integer linear program formulation for the basic BMD as in Equation 4.6, the administrative cost can be formulated as the following

$$||X||_1 + ||C||_1 = \sum_i \sum_t X_{it} + \sum_t (d_t \sum_j R_{tj}). \tag{4.8}$$

$\sum_i \sum_t X_{it}$ is the count of positive cells in X and $\sum_t (d_t \sum_j R_{tj})$ counts positive cells in selected candidate roles, which is $||C||_1$. Thus, the integer linear program formulation for cost BMD can be obtained by replacing the objective function in Equation 4.6 with the formula in Equation 4.8.

4.3.5 Discussion

From the integer linear program formulating processes for those role mining variants, we observe that once one role mining variant is successfully formulated, it is easy to formulate other variants. This fact illustrates the benefit of building a unified integer linear program framework of the role mining problem. Such an integer linear program framework is broad and flexible to incorporate new variants, which may appear in practice. For instance, a role mining engineer may not want to see 0-becoming-1 errors in an approximate BMD solution, because 0-becoming-1 errors mean that a user obtains unauthorized permissions, which may harm the system security severely. To reflect this concern in the integer linear program formulation , we could simply replace the constraint for $A_{ij} = 0$ of $\sum_{t=1}^{q} X_{it} R_{tj} - V_{ij} = 0$ in Equation 4.7 by $\sum_{t=1}^{q} X_{it} R_{tj} = 0$. Consider another instance that all roles are expected to be highly repetitively used. To realize this expectation, we could add a constraint that $\sum_i X_{it} \geq LB$, where $\sum_i X_{it}$ is the number of users granted role t and LB is a specified frequency lower bound. So every picked role is employed more than LB times.

4.4 Algorithm Design for BMD Variants

The integer linear program formulation for the partial BMD takes mn variables and for other BMD variants takes about mq variables, where m is the number of users, n is the number of permissions, and q is the number of candidate roles. The state-of-art integer linear program software Packages can deal with up to millions variables. So for problems of sizes with m and q greater than 1000, we have to resort to efficient heuristics. In this section, we will propose efficient heuristics for BMD variants. They are easy to implement and run fast. Their effectiveness will be validated in the next section. Like the above integer linear program formulations that require candidate roles to be given, our heuristics need candidate concepts generated beforehand as well. So before presenting our heuristics, we will first discuss ways of generating candidate concepts. For ease of explaining, we will still discuss it in the role mining setting.

4.4.1 Candidate Role Set Generating

We present three ways of generating candidate roles. All of them were somehow mentioned in other peoples' work before either in the role mining context or in the discrete basis context.

Intersection. We call the first method *Intersection*, which is proposed by Vaidya et. in [62]. This method includes every unique user's permission set as a candidate role. In addition the intersections of every two user permission sets are included as candidate roles as well. This method is based on two observations. First in order to make a permission subset be a role, it must be employed and assigned to some user. In other words, a candidate role must be a subset of

some user's permission set. The other observation is that to make a RBAC system succinct and efficient, roles should be repetitively employed. So a role is desired to be assigned to multiple users. Hence, a candidate role is expected to be the intersection of two or multiple user's permission sets. Given m users, candidate roles generated by the *Intersection* method is $o(m^2)$.

Association. This method is to exploit the correlations between the columns of existing user-to-permission assignments A by employing the association rule mining concept in [52]. It was presented as a part of the *ASSO* algorithm proposed for the discrete basis problem. The concrete generation process is as follows. Suppose user-to-permission assignments $A_{m\times n}$ are given. Let $A(:,i)$ denote the ith column. Then n candidate roles will be generated, represented by a Boolean matrix $C_{n\times n}$. In which, $C_{ij} = 1$ if $\langle A(:,i), A(:,j)\rangle \backslash \langle A(:,i), A(:,i)\rangle \geq \tau$; otherwise 0. The operator $\langle ., .\rangle$ is the vector inner product operation, and τ is a predetermined threshold controlling the level of correlation.

Itself. This method is to simply treat unique user's permission sets from A as candidate roles. It was studied in [48], but in the scenario of the discrete basis problem. The problem is to find a discrete basis for an input Boolean matrix, such that the discrete basis is a subset of columns (or rows) of the input Boolean matrix. In the role mining context, if employing this method to generate candidate roles, it is assumed that for every role there must be one user who is granted this role and only this role.

4.4.2 Partial BMD

As we mentioned earlier, partial BMD is equivalent to the basis usage problem. A heuristic called Loc& IterX was proposed for the basis usage problem in [48]. Here we propose two more effective heuristics, one greedy approach and one simulated annealing approach.

Without loss of generality, we assume that there is only one user. Hence A is a $1 \times n$ row vector. A role set $C_{r\times n}$ is given. partial BMD becomes to find a role subset, such that the union of permissions contained in those roles is closest to A.

Greedy. A greedy algorithm is any algorithm that makes the locally optimal choice at each stage with the hope of finding the global optimum. It has many successful applications, such as the traveling salesman problem and the knapsack problem. Our greedy approach consists of a preliminary step. It is to assign all roles in C which are *subordinate* to A to the user. A role C_1 is considered to be subordinate to a role C_2 if all permissions contained by C_1 belong to C_2. After that, iteratively pick one remaining role which can reduce the reconstruction error by the most and assign it to the user, till the reconstruction error cannot be reduced any more.

Simulated Annealing Simulated annealing is a generic probabilistic heuristic for the global optimization problem. It locates a good approximation to the global optimum of a given function in a large search space. Different from greedy heuristics, simulated annealing is a generalization of a Markov Chain Monte Carlo method, which has a solid theoretical foundation. We present a simulated annealing heuristic for the partial BMD problem as the following. First, we let $(0, ..., 0)$

Algorithm 4.1 Greedy Algorithm for Partial BMD

Input: $A_{1\times n}$ and $C_{r\times n}$
Output: $X_{1\times r}$

```
1: X_{1×r} ← (0, ..., 0);
2: for i ← 1 : r do
3:    if C_i ⊆ A then
4:       X[i] = 1;
5:    end if
6: end for
7: loop
8:    C_i=the best remaining role, j=the largest improvement;
9:    if j > 0 then
10:      X[i] = 1;
11:   else
12:      break
13:   end if
14: end loop
```

be the starting state of $X_{1\times r}$. Then at each stage, randomly select its neighboring value by randomly picking one element of X and flipping its value from 0 to 1 or from 1 to 0. If the new X better reconstructs A, the next state is the new X. If not, with a certain probability less than 1, the next state is the new X . In other words, with certain probability, it remains its original state. This property reduces the chance of being stuck at a local optimum. The procedure described above allows a solution state to move to another solution state and hence produces a Markov Chain. Denote the nth state be x and the randomly selected neighboring value be y. If the next state is y with probability $min\{1, \frac{exp\{\lambda V(x)/N(x)\}}{exp\{\lambda V(y)/N(y)\}}\}$] or it remains x, where λ is a constant, $V(t)$ is the reconstruction error with the solution t, and $N(t)$ is the number of neighboring values of t. Such a Markov Chain has a limiting probability of 1 for arriving at optimal minimization solutions when $\lambda \to \infty$ [56]. But it has been found to be more useful or efficient to allow the

Algorithm 4.2 Simulated Annealing Algorithm for Partial BMD

Input: $A_{1\times n}$ and $C_{r\times n}$
Output: $X_{1\times r}$
1: $X, x \leftarrow (0, ..., 0)$; $n \leftarrow 1$;
2: **while** $n \leq limit$ & $V(X) > 0$ **do**
3: y= a random neighboring value of x;
4: **if** $V(y) < V(X)$ **then**
5: $X \leftarrow y$;
6: **end if**
7: $x = y$ with probability
$min\{1, \frac{exp\{log(1+n)V(x)\}}{exp\{Clog(1+n)V(y)\}}\}$;
8: $n \leftarrow n + 1$;
9: **end while**

value of λ to change with time. Simulated annealing is a popular variation of the preceding. Here, we adopt the formula proposed by Besag et al. [54] and let the transition probability be $min\{1, \frac{exp\{\lambda_n V(x)/N(x)\}}{exp\{\lambda_n V(y)/N(y)\}}\}$ where $\lambda_n = log(1 + n)$. In our case, $N(x)$ and $N(y)$ are equivalent and are canceled out in the formula. As computing time is limited, we terminate the algorithm after a certain number of iterations regardless of whether or not the global optimum is reached. Our complete simulated annealing algorithm is described as in Algorithm 4.2.

4.4.3 Basic BMD

The heuristic proposed for the basic BMD also runs in a greedy fashion. Look at the integer linear program formulation for the basic BMD as Equation 4.6. There are two main types of constraints, one for $\{A_{ij} = 1\}$ and the other for $\{A_{ij} = 0\}$. In order to satisfy the constraint set $\{\sum_{t=1}^{q} X_{it}R_{tj} = 0, if\ A_{it} = 0\}$, if $R_{jt} = 1$, X_{it} must be equal to 0. Therefore, many variables X_{it} can be determined directly in this way. Then the constraint set $\{\sum_{t=1}^{q} X_{it}R_{tj} = 0, if\ A_{it} = 0\}$ is all satisfied.

Next consider $\{\sum_{t=1}^{q} X_{it}R_{tj} \geq 1, if\ A_{it} = 1\}$. To satisfy it, if the particular

cells $R_{t_1 j}, R_{t_2,j}, ...R_{t_k,j}$ are 1, one of the associated cells $X_{it_1}, X_{it_2}, ...X_{it_k}$ has to be 1.

Now, consider the objective function $min \sum_{t=1}^{q} d_t$. As d_t is determined by the constraint $d_t \geq X_{it}$. Hence once one of the cells $\{X_{it}, \forall t\}$ is 1, d_t is forced to be 1. In other words, no matter how many variables in $\{X_{it}, \forall t\}$ take the value of 1, the objective function value is always increased by 1. Our greedy algorithm utilizes this property. At each step, choose a candidate role R_t such that by setting $\{X_{it}, \forall t\}$, except those predetermined, be 1, the most remaining constraints $\{\sum_{t=1}^{q} X_{it} R_{tj} = 1, if\ A_{ij} = 1\}$ are satisfied. We call the count of such satisfied remaining constraints the basic-key.

Definition 4.2 (Basic-Key) *For each column of the variable matrix $\{X_{it}\}$, the count of the constraints $\{\sum_{t=1}^{q} X_{ij} R_{tj} = 1,\ if\ A_{ij} = 1\}$ being satisfied by letting the cells of such a variable column be 1 except the cells predetermined, is called its basic-key.*

If there are multiple columns with the same greatest basic-key, we simply choose the column with the least index. Once a column is chosen, it means that the associated candidate role is chosen and the associated d_t is set to be 1. Then delete the satisfied constraints and perform the same procedure repetitively till all the constraints are satisfied. The full procedure of this greedy algorithm is given in Algorithm 4.3.

Algorithm 4.3 Greedy Algorithm for Basic BMD

Input: Unique user-to-permission assignments $A_{m\times n}$ and candidate roles $\{R_1, ..., R_q\}$.

Output: X and C

1: $C = \emptyset$;
2: Investigate the constraints $\{\sum_{t=1}^{q} X_{it}R_{tj} = 0\}$ in Equation 4.6 and determine some of X_{it} to be 0;
3: Select the candidate role R_t with the greatest basic-key value and include it in C.
4: Let undetermined values in $\{X_{it}\}$ be 1 and deleter hence satisfied constraints in $\{\sum_{t=1}^{q} X_{it}R_{tj} > 1\}$;
5: Go back to step 3 till all constraints of $\{\sum_{t=1}^{q} X_{it}R_{tj} = 0\}$ and $\{\sum_{t=1}^{q} X_{it}R_{tj} > 0\}$ are satisfied.
6: Set the remaining variables in X to be 0.
7: X is the subset of rows in X corresponding to selected roles in C.

4.4.4 Approximate BMD

From the integer linear program formulation perspective, approximate BMD seems much more complicated than basic BMD. However an efficient greedy heuristic for the approximate BMD can be easily developed by modifying the greedy heuristic for the basic BMD. As the δ-approximate BMD tolerates δ amount of errors, we can terminate Algorithm 4.3 early once the remaining uncovered 1's cells are less than δ.

4.4.5 Cost BMD

Cost BMD is to minimize the administrative cost $||X||_1+||C||_1$, while basic BMD only aims to minimize the number of roles. So a basic BMD optimal solution is not necessarily a cost BMD solution. Cost BMD was studied in [61], where a greedy heuristic was proposed. To distinguish it, we call it *Edge-Key*. Here we propose a new heuristic, called *Two-Stage*.

Cost BMD objective consists of two parts $||X||_1$ and $||C||_1$. Intuitively less number of roles leads to a less value of $||C||$. In this sense, an optimal solution of basic BMD can be a not-bad solution for cost BMD. Our greedy heuristic for basic BMD is able to produce a good C. However, X produced by the greedy heuristic is just a byproduct and not in its optimal form. We can start a second phase and reduce user-role assignments by reassigning obtained roles of C to each user. It is essentially another basic BMD problem that assigning the minimum roles from C to exactly cover all permissions for a user. So here we can adopt our greedy heuristic again. It is a common fact that if a role contains permissions not originally possessed by a user, the role can never be assigned to the user. With this rational, we can simplify the second-phase task by filtering out unlikely roles from C in advance. The complete algorithm is as described in Algorithm 4.4.

Algorithm 4.4 Two-Stage Algorithm for Edge-BMD

Input: user-to-permission assignments $A_{m\times n}$ and candidate roles $\{R_1, ..., R_q\}$.
Output: $||X||_1 + ||C||_1$

1: Run Algorithm 4.3 with $\{R_1, ..., R_q\}$ to obtain C.
2: **for** each A_i **do**
3: Remove roles that contain permissions not belonging to A_i from C.
4: Run Algorithm 4.3 with remaining roles in C to obtain X_i.
5: **end for**

4.5 Experimental Study

In this section, we conduct extensive numerical experiments on both synthetic data sets and real data sets to evaluate the performance of our heuristics.

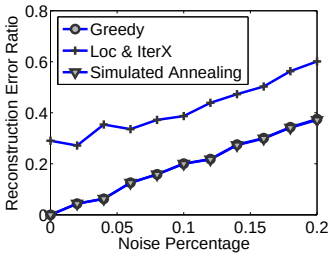
m	# of users
n	# of permissions
r	# of roles
ρ_1	density of C
ρ_2	density of X
$noise$	noise percentage
$limit$	maximum number of iterations

Figure 4.3. Notations

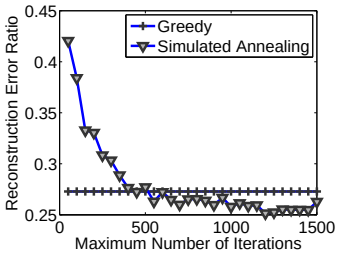
4.5.1 Synthetic Data

The synthetic data sets are generated as follows. Firstly, generate a set of unique roles $C_{r\times n}$ and user-to-permission assignments $X_{m\times r}$ in a random fashion. A is the Boolean product of C and X. Two parameters ρ_1 and ρ_2 are employed to control the density of 1's cells in C and X respectively, which then determine the density of 1's cells in A. Precisely, to create a role, generate a random number $Poissrnd(\rho_1)$ from a poisson distribution with the mean of $\rho_1 \times n$. If the generated number is greater than n, perform it again. Then randomly generate a Boolean vector with $Poissrnd(\rho_1)$ 1's elements. We generate a user's role assignment in a similar way, except replacing ρ_1 with ρ_2. To reflect real data sets, we also add in noise by flipping the values for a portio of cells. $noise$ is the noise percentage parameter. For convenience of reference, we list all notations in Figure 4.5.1. In which, the parameter $limit$ is used in Algorithm 4.2 to control the maximum number of iterations.

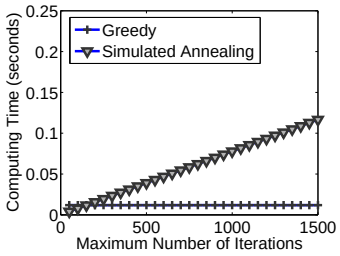
The first experiment is to study partial BMD. We compare the Loc & IterX algorithm proposed in [48] with our greedy heuristic and our SA heuristic with $limit$ of 500. Three algorithms are compared with respect to the reconstruction



(a) ρ_1=0.3, ρ_2=0.3, m=50, n=50, r=10, $limit$=500



(b) ρ_1=0.3, ρ_2=0.3, m=1, n=50, r=30, $noise$=0.2



(c) ρ_1=0.3, ρ_2=0.3, m=1, n=50, r=30, $noise$=0.2

Figure 4.4. Experimental Results on Partial BMD

error ratio, which is defined as $ErrorRatio = ||A - A'||_1/||A||_1$, where A' denotes the reconstructed A.

The concrete experimental design is as follows. (i) Generate a number of data sets $\{X, C, A\}$ with $m = 50$, $n = 50$, $\rho_1 = 0.3$, $\rho_2 = 0.3$ and $noise$ varying from 0 to 0.2; (ii) Run the Loc & IterX, the greedy algorithm, and the SA algorithm with $limit$ of 500 respectively to obtain X and A' given C and A. The experiment result is illustrated as shown in Figure 4.4a. It shows the greedy algorithm and the SA algorithm are significantly better than the Loc & IterX algorithm. The other observation is that the performances of the greedy heuristic and the SA heuristic are comparable.

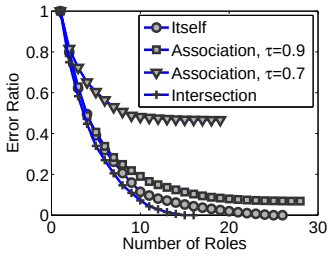
Different from the greedy heuristic, which returns a local optimum in most cases, a SA heuristic can with probability one reach a global optimum with enough

computing time; in other words unlimited number of iterations. However, the meaning of existence for a heuristic is that it can get a good solution in an acceptable time. So we conduct another experiment to investigate how our SA heuristic performs as opposed to $limit$. We generate 100 sets of $\{X, C, A\}$ with ρ_1=0.3, ρ_2=0.3, m=1, n=50, r=30, and $noise$=0.2. $limit$ varies from 50 to 2500. Run both the greedy heuristic and the SA heuristic against them. Figure 4.4b shows that the SA heuristic has better performance than the greedy heuristic when $limit$ is greater than 500. It also shows at the beginning small increase in $limit$ can improve the performance by a lot for the SA heuristic. However, when $limit$ reaches certain point, the improvement becomes much less significant. It somehow shows that the greedy heuristic has relatively satisfactory performance. Figure 4.4c plots computing times for both methods. The computing time for the SA heuristic increases linearly with $limit$. However, to achieve the same performance as the greedy heuristic, the SA heuristic takes more time. So we conclude that the SA heuristic is recommended for small-size problems while the greedy heuristic is suitable for large-size problems.

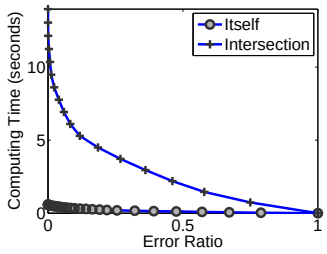
Next we study our greedy heuristic for basic BMD. The heuristic is closely dependent on candidate roles. Three ways of generating candidate roles were introduced: *Itself*, *Intersection*, and *Association*. As *Association* has a parameter of association threshold τ, for a fair comparison, we consider two cases, $\tau = 0.9$ and $\tau = 0.7$. We generate $\{A\}$ with ρ_1=0.3, ρ_2=0.3, m=50, n=50, r=10, $noise$=0. Then run Algorithm 4.3 with each candidate role set respectively to find X and C. The results are illustrated as shown in Figure 4.5a. Both *Intersection* and *Itself* are able to reconstruct A completely, but *Association*. With respect to the error

reducing speed, *Association* is inferior to two other approaches. Overall *Intersection* has the best performance, while the performance of *Association* is far from satisfactory. So ignoring *Association*, we then further study *Itself* and *Intersection* with respect to computing time. Figure 4.5b shows that *Intersection* takes much more time than *Itself*. The underlying reason is that *Intersection* produces $o(m^2)$ candidate roles as opposed to $o(n)$ candidate roles produced by *Itself*. We conclude that for small size problems *Intersection* is preferred while *Itself* is recommended for large size problems.

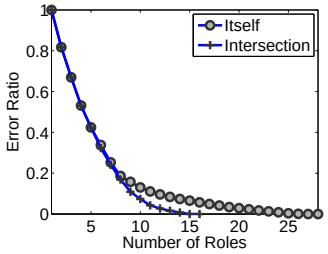
Our greedy heuristic for approximate BMD is same as that for basic BMD, except that it terminates early. So the previous experimental results are valid for studying approximate BMD. Figure 4.5c plots reconstruction error ratio values with respect to required number of roles. If a small amount of errors are allowed, A can be successfully reconstructed with much less number of roles. For example, *Itself* needs only 10 roles to cover more than 80 percent of existing permissions, while 20 extra roles are needed to cover the remaining permissions. Somehow it can be interpreted that roles returned by approximate BMD are more "fundamental". It suggests that if a bottom-up role engineering approach is only used to identify promising roles to assist a top-down engineering approach, approximate BMD is more efficient. Figure 4.5d shows the relation between data density and the number of required roles for full coverage. As the data density is indirectly determined by ρ_1 and ρ_2. The experiment is to let ρ_1 and ρ_2 have the same value and vary them together from 0.1 to 0.6. In Figure 4.5d, both lines suggest that for denser data sets more roles are required. In other words, our greedy heuristic performs better for sparse data sets, while real access control data sets are usually



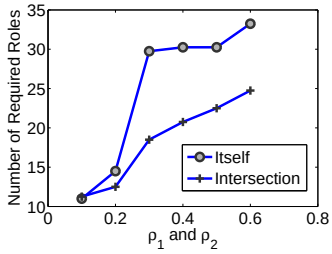
(a) ρ_1,ρ_2=0.3, m,n=50, r=10, $noise$=0



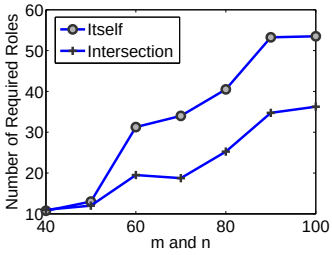
(b) ρ_1,ρ_2=0.3, m=1, n=50, r=30, $noise$=0.2



(c) ρ_1,ρ_2=0.3, m,n=50, r=10, $noise$=0



(d) m,n=50, r=10, $noise$=0



(e) ρ_1,ρ_2=0.3, r=10, $noise$=0

Figure 4.5. Experimental Results on Basic BMD and Approximate BMD

very sparse. We then study the relation between data size and our greedy heuristic performance. We let m and n be same and vary them from 40 to 100. Figure 4.5e illustrates that our heuristic performs good for medium and small sizes.

Next we evaluate the *Two-Stage* algorithm proposed for edge BMD by comparing it with the *Edge-Key* algorithm proposed in [61]. For convenience, we let default parameters for generating A be $\{m = 50, n = 50, \rho_1 = 0.3, \rho_2 = 0.3, noise = 0\}$. We then variate one parameter each time, generate a set of As, and run *Two-Stage* and *Edge-Key* respectively on them. As both algorithms require input candidate roles, we consider both *Itself* and *Intersection*. Hence, we actually compare four approaches: *(Two-Stage, Itself), (Two-Stage, Intersection), (Edge-Key, Itself),*and *(Edge-Key, Intersection)*. Experimental results are illustrated as shown in Figure 4.6. All four graphs suggest that *Two-Stage* is better than *Edge-Key*.

4.5.2 Real Data

We run our greedy heuristics on real data sets collected by Ene et al. [16]. They are **emea**, **healthcare**, **domino**, **firewall 1**, and **firewall 2**.

The first experiment is to study the basic BMD. We run Algorithm 4.3 against each data set with candidate roles generated by *Itself* and *Intersection* respectively. Numbers of roles required for full coverage are recorded in Figure 4.7. The number of required roles is much less than the number of permissions for each case. It successfully demonstrates the power of RBAC and also the effectiveness of our heuristic on discovering roles. Another observation is that the performance of *Intersection* is not always better than *Itself* when working with our greedy heuristic.

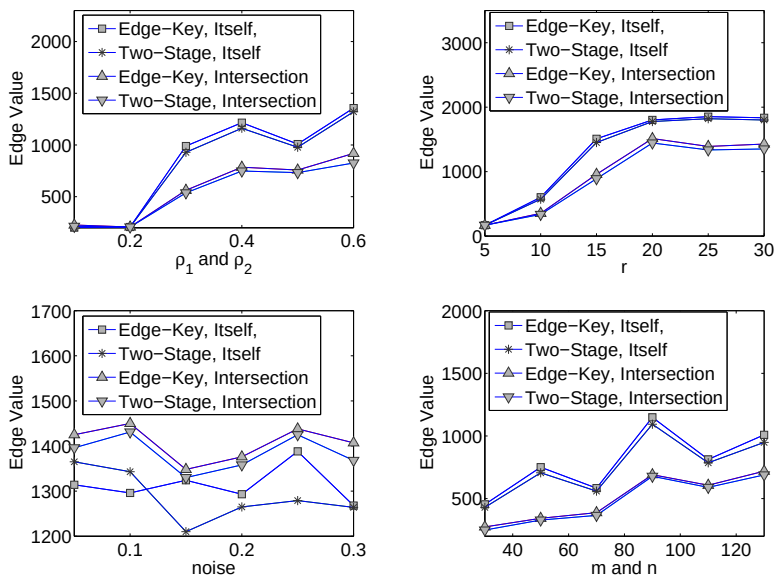


Figure 4.6. Reconstructed Error Ratio w.r.t. Percentage of Required Roles For Real Data Sets

In theory, with *Intersection* the optimal solution of a basic BMD is better than that with *Itself*, as the feasible solution space is expanded. However, if the algorithm runs in a greedy manner, it is not always true.

The second experiment is to study the δ-approximate BMD. We run Algorithm 4.3 and record remaining errors when a new role is identified. Figure 4.8 plots reconstruction error ratios with respect to number of required roles. The first observation is that all lines drop fast at the beginning. It suggests that a few roles are usually able to reduce error ratio to a very low level. Roles early identified can hence be interpreted as ”more valuable”. Another observations is that much more roles are required to cover the remaining few 1’s at the end. It suggests hat if a role mining approach is used as a tool to assist a top-down role engineering approach, δ-approximate BMD might be sufficient. The other important observation is that

Data Set	Users	Permissions	$\|\|A\|\|_1$	Itself		
				# of Roles	Edge Value	
					Edge-Key	Two-Stage
emea	35	3,046	7,220	34	7,281	7,246
healthcare	46	46	1,486	16	624	478
domino	79	231	730	20	810	727
firewall 1	365	709	31,951	71	5,475	4,937
firewall 2	325	590	36,428	10	1,796	1,456

Data Set	Users	Permissions	$\|\|A\|\|_1$	Intersection		
				# of Roles	Edge Value	
					Edge-Key	Two-Stage
emea	35	3,046	7,220	43	9,078	9,014
healthcare	46	46	1,486	14	553	412
domino	79	231	730	21	814	827
firewall 1	365	709	31,951	65	4,508	4,034
firewall 2	325	590	36,428	10	1,796	1,456

Figure 4.7. Number of Roles and Edge Value for Full Coverage for Real Data Sets

Intersection dose not always perform better than *Itself* in our greedy heuristic. In fact, if only considering early mined roles, their performances are comparable.

The third experiment is to study the edge BMD. We run *(Two-Stage, Itself), (Two-Stage, Intersection), (Edge-Key, Itself),*and *(Edge-Key, Intersection)* respectively against each data set. Their edge values are recorded in Figure 4.7. Our *Two-Stage* algorithm performs better than the *Edge-Key* algorithm for nearly all cases except for the **domino** data set with *Intersection*. The experimental results demonstrate the advantage of RBAC over the traditional access control system once again, as the edge-value (administration cost) is much less than $||A||_1$ for any case.

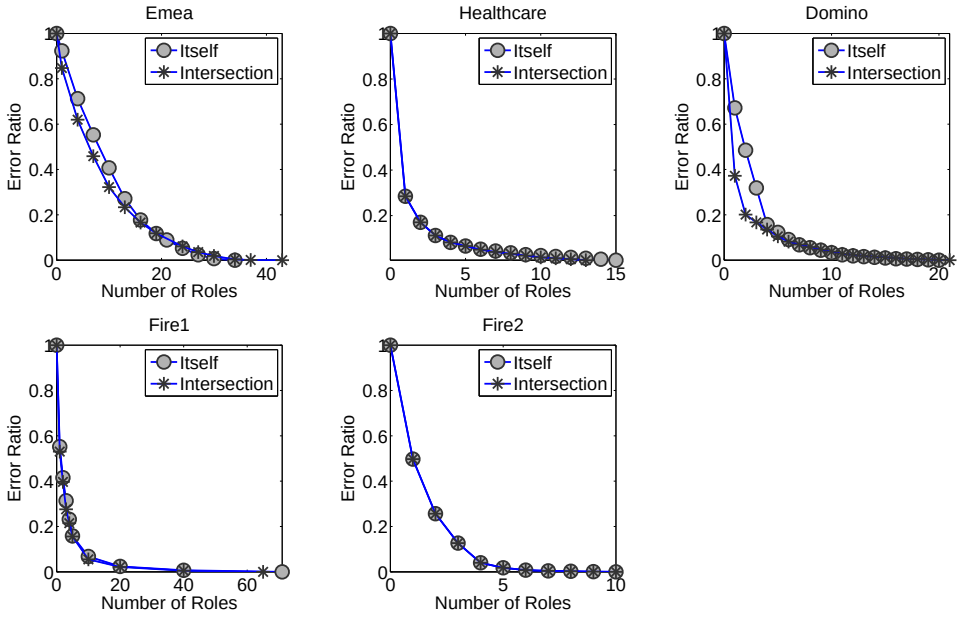


Figure 4.8. Reconstructed Error Ratio w.r.t. Number of Required Roles For Real Data Sets

CHAPTER 5
EXTENDED BOOLEAN MATRIX DECOMPOSITION

Boolean matrix decomposition is to decompose a Boolean matrix into the product of two Boolean matrices, where the first matrix represents a set of meaningful concepts, and the second describes how the observed data can be expressed as combinations of those concepts. The combination is only in terms of the set union. In other words, a successful Boolean matrix decomposition gives a set of concepts and shows how every column of the input data can be expressed as a union of some subset of those concepts.

However, this way of modeling only incompletely represents real data semantics. Essentially, it ignores a critical component – the set difference operation: a column can be expressed as the combination of union of certain concepts as well as the exclusion of other concepts. This has two significant benefits. First, the total number of concepts required to describe the data may itself be reduced. Second, a more succinct summarization may be found for every column. In this paper, we propose the extended Boolean matrix decomposition (EBMD) problem, which aims to factor boolean matrices using both the set union and set difference operations. We study several variants of the problem, show that they are NP-hard, and propose efficient heuristics to solve them. Extensive experimental results demonstrate the power of EBMD.

To better explain EBMD, we will study it in the context of role mining in the following.

5.1 Motivation of EBMD

Role-based access control (RBAC) has proven to be a very successful model for access control. Its flexibility and cost-efficiency have resulted in it being widely adopted by current commercial systems. Indeed, it has been the model of choice for migration in the case of enterprises still employing traditional access control schemes. However, for such enterprises, the first and indeed the most crucial step is to design a good set of roles and assign them appropriately to each user. This process of designing roles is called role engineering [11]. While top-down techniques have been proposed for role engineering, for large-scale enterprises with more than thousands of users and permissions, bottom-up role extraction, called *role mining*, which mines roles purely from existing user-permission assignments without considering their semantic meanings, has become quite popular.

Semantics like *exceptions* and *separation of duty constraint* (SoD) are indispensable parts of RBAC,critical to model real-world cases. While SoD constraints are restrictions on users and roles to capture policy semantics, allowing the semantics related to exceptions are needed for a more succinct translation of access control policies to the actual specification. However, existing solutions for role mining such as BMD are not able to capture or reflect semantics, specifically the exceptions and separation of duty constraints. We will explicitly explain both below.

Exceptions: Exceptions are inherent to any real world access control policy that

uses some notion of abstraction in the authorization. Since the focus of this paper is on role mining, we consider the role based access control policy. Suppose the RBAC policy states that any user with role “manager” is allowed to access the file “project A”. However assume there exists an exception to this policy stating that all users except John (who can play the role of the manager) is not allowed to access “project A” due to certain conflict of interest requirements. Such exceptions are quite common to real world policies. Under a typical RBAC policy this is supported through a negative authorization as it does not make sense to create a new role specifically to John alone. It is important to realize that supporting negative authorizations sometimes may result in conflicting authorizations (in this case due to permission inheritance through role hierarchy). These can be handled by implementing conflict resolution policies (in this example, negatives take precedence). Assume other users assigned to “manager” are Alice, Bob, Cathy, Dave and Eve, and the permission to access “project A” is p, the corresponding user-to-permission assignments of this example would be as shown in figure 5.1. Traditional role mining approaches attempt to mine roles that have the same permission sets. In this case, two roles will be mined, first comprising of Alice, Bob, Cathy, Dave and Eve, and the second with John alone. Our proposed role mining approach in this paper attempts to capture the underlying semantics of such exceptions and eliminates mining of such incorrect role sets. Note that similar exceptions can be found with other abstractions of authorizations. An example of such a policy would be “John is allowed to access all project reports except the report of project A”. Our approach can elegantly handle exceptions of these kinds as well.

user	p
Alice	1
Bob	1
Cathy	1
Dave	1
Eve	1
John	-1

Figure 5.1. User-to-permission assignments of the example above

Separation of Duty Constraints: SoD constraints are an integral part of RBAC, as stated in the definition of $RBAC_2$ [57]. These help to limit exploitation of privileges and limit fraud. Consider the following toy example. Assume the following four permissions of a company: "Purchasing", "Auditing", "Marketing" and "Sales". A person can assume multiple permissions. Suppose that the same person is in charge of purchasing and sales. Hence these two permissions are grouped together as a role, which can be represented by a Boolean vector as $\{1, 0, 0, 1\}^T$. To prevent fraud, the company has a policy stating that a person cannot assume both "Purchasing" and "Auditing" permissions. Simply representing a role as a Boolean vector cannot reflect this constraint. Even though the "auditing" permission is not included in $\{1, 0, 0, 1\}^T$, a person who has been assigned this role, can obtain the "Auditing" permission by acquiring other roles, which is perfectly valid in the BMD model. We however, would like to recognize such constraints as part of the mining process itself.

To address this ineffectiveness of the BMD model in capturing semantics, we propose introducing *negative permissions* or *negative user-role assignment*, which can cleverly resolve both of the above issues.

As distinct from regular permissions, negative permissions mean that once

a permission is assigned to a user negatively, this user can never exercise that permission. Thus, negative permissions have higher priority than positive permissions. Indeed, if the user is already assigned the permission positively through another role or even through the hierarchy, this assignment is automatically revoked. If the user is assigned the permission positively in the future, it still does not become effective. Thus, negative permissions yield a great power and can effectively model both SoD constraints and exceptions.

SoD constraints can be modeled through introducing negative permissions in roles. Consider again the "Purchasing" and "Auditing" example. To enforce the SoD constraint on them, for any role containing one of them, we add the negative permission of the other. Hence, the role of $\{1, 0, 0, 1\}^T$ is changed to $\{1, -1, 0, 1\}^T$, where the cell of -1 denotes the negative "Auditing" permission. As a result any employee assuming that role can never have the "Auditing" permission, unless the role assignment is revoked. We denote such roles, allowing negative permissions, as *semantic roles*.

Exceptions can be modeled through introducing negative user-role assignments. Negative user-role assignments mean that if a role is assigned to a user, the user cannot have access to any permission of that role. The negative user-role assignment is superior to the positive (or regular) user-role assignment. Revisiting the "Manager" example of John. To forbid him from accessing "project A", we only need to assign the "manager" role negatively to him. We call such user-role assignments, which include both positive and negative assignments, as *semantic user-role assignments*.

Indeed, negative authorizations are integral part of many access control sys-

tems. From the work of Bertino et al. [6, 7], introducing negative authorizations have many advantages. They enable a temporary suspension of a permission from a user without having to revoke it (revoking a permission sometimes may have a cascading effect), allow exceptions to be specified, and prevent a user from being able to exercise a privilege.

We observe that in addition to increasing administration flexibility, negative authorizations can help discover semantics underlying existing user-permission assignments during the role mining process. Consider the example of existing user-permission assignments A as shown in Figure 5.2, where $\{u_1, u_2, u_3, u_4\}$ denote users and $\{p_1, p_2, p_3, p_4\}$ denote permissions.

	p_1	p_2	p_3	p_4
u_1	1	0	1	1
u_2	1	0	1	1
u_3	1	1	0	1
u_4	0	1	0	1

Figure 5.2. Existing User-Permission Assignments

One optimal solution of the conventional role mining problem, minimizing required roles, is as shown in Figure 5.3, where $\{r_1, r_2, r_3\}$ denote roles. The first Boolean matrix gives user-role assignments X and the second Boolean matrix represents permission-role assignments C. In fact, X and C are a Boolean matrix decomposition (BMD) solution of A and represented by $A = X \otimes C$ [42].

If we allow negative permissions in roles, a role r_i would consist of two parts, positive permissions P_i^+ and negative permissions P_i^-. Hence, a role can be represented as a vector in $\{-1, 0, 1\}$. For example, a vector $(-1, 0, 1)^T$ denotes a role with the negative authorization for the first permission and the positive au-

	r_1	r_2	r_3
u_1	1	0	1
u_2	1	0	1
u_3	1	1	0
u_4	0	1	0

(a) X

	p_1	p_2	p_3	p_4
r_1	1	0	0	1
r_2	0	1	0	1
r_3	0	0	1	0

(b) C

Figure 5.3. Conventional Role Mining

thorization for the third permission. Assigning r_i to a user means that the user can never have any permission of P_i^- unless r_i is revoked and the user can have a permission of P_i^+ if he is not assigned any role consisting of its negation.

Now let us do the same thing as the conventional role mining problem, minimizing the number of required roles. The only difference is that negative permissions are allowed this time. As we expect it to discover underlying data semantics, we call it the *semantic role mining problem*. For the same user-permission assignments as above, the resultant optimal solution is as shown in Figure 5.4.

	r_1	r_2
u_1	1	0
u_2	1	0
u_3	1	1
u_4	0	1

(a) X

	p_1	p_2	p_3	p_4
r_1	1	0	1	1
r_2	0	1	-1	1

(b) C

Figure 5.4. Semantic Role Mining with Negative Permission

The first impression on the result is that with negative authorization for permissions we need only two roles to reconstruct the same existing user-permission assignments. Further by taking a close look, you can find more information. First, $r_2 : \{0, 1, -1, 1\}$ shows that if r_2 is assigned to a user, he can never has

the privilege of p_3. It implies that p_3 might be exclusive from p_2 and p_4. Second, $r_1 : \{1, 0, 1, 1\}$ shows that p_2 and p_4 can be existent in one role. It leaves one plausible explanation that there is a separation of duty constraint on p_2 and p_3. So the real semantic might be there are indeed only two roles in the system, $\underline{r_1 : \{1, 0, 1, 1\}}$ and $\underline{r_2 : \{0, 1, 0, 1\}}$. The reason that u_3 does not get p_3 even thought he is assigned r_2, is the separation of duty constraint on p_2 and p_3. However, the conventional role mining approach is not able to discover such semantics. Compared to the result in Figure 5.3, the result in Figure 5.4 seems more plausible.

This toy example demonstrates the ability of negative authorization on discovering underlying semantics. To perform semantic role mining, we propose introducing a new approach, extended Boolean matrix decomposition (EBMD). As its name tells, EBMD extends from BMD. It allows -1 in one of the decomposed matrices. Thus, EBMD is to decompose one Boolean matrix into one Boolean matrix and one matrix in $\{-1, 0, 1\}$.

From the technical perspective, semantic role mining is like finding a good EBMD solution of the Boolean matrix corresponding to given user-permission assignments. However, it is more complicated than that. The particular role mining context has to be incorporated in the matrix decomposition process.

5.2 Extended Boolean Matrix Decomposition

In this section, we will introduce a novel matrix decomposition method EBMD, which addresses the ineffectiveness of BMD in its ability of capturing real data semantics. To help better understand the function of EBMD, let us recall BMD first.

BMD is essentially to discover a set of discrete concepts and use them to describe each observed Boolean record as a union of some discrete concepts. The key advantage of BMD is to provide much interpretability to decomposition solutions. To illustrate it, look at the example of Equation 5.2.

$$\begin{pmatrix} & a1 & a2 & a3 & a4 \\ d1: & 1 & 1 & 0 & 1 \\ d2: & 0 & 1 & 1 & 0 \\ d3: & 1 & 0 & 0 & 1 \\ d4: & 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} & a1 & a2 & a3 & a4 \\ c1: & 1 & 1 & 0 & 1 \\ c2: & 0 & 1 & 1 & 0 \\ c3: & 1 & 0 & 0 & 1 \end{pmatrix}. \tag{5.1}$$

The matrix on the left is the observed records. In which, 1 means that the record consists of the attribute. For example, $d1 = \{a1, a2, a4\}$. The matrix on the right is the discovered concepts, each of which is a subset of attributes. For example, $c1 = \{a1, a2, a4\}$. The combination matrix tells how observed records can be described as a union of some discovered concepts. For example, $d4 = c1 \cup c2$.

BMD does provide much interpretability to matrix decomposition solutions. However, it is only able to represent the set union operation. In reality, some data semantics requires the representation of the set difference operation as well. For example, in the access control setting, a role could be negatively assigned to a user, such that any permissions belonging to the role can never be assigned to the user.

To enable BMD to capture the set difference operation, we introduce a new concept EBMD, which allows -1 in the combination matrix and uses it to represent the set difference operation.

Consider the same above example. By introducing -1 in the combination matrix, we obtain a new decomposition solution as below, where $\odot$ denotes the EBMD multiplication operator.

$$\begin{pmatrix} & a1 & a2 & a3 & a4 \\ d1: & 1 & 1 & 0 & 1 \\ d2: & 0 & 1 & 1 & 0 \\ d3: & 1 & 0 & 0 & 1 \\ d4: & 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & -1 \\ 1 & 1 \end{pmatrix} \odot \begin{pmatrix} & a1 & a2 & a3 & a4 \\ c1: & 1 & 1 & 0 & 1 \\ c2: & 0 & 1 & 1 & 0 \end{pmatrix}. \tag{5.2}$$

Notice that with the introduction of the set difference operation only two concepts are needed to represent the same observed data. The combination matrix shows that: $d_1 = c_1$, $d_2 = c_2$, $d_3 = c_1 \setminus c_2$, and $d_4 = c_1 \cup c_2$.

As illustrated, EBMD is to describe a set of observed records with a small set of concepts, such that each record can be represented as inclusion of one subset of concepts with exclusion of another subset of concepts. If a record includes one concept, that record should contain all elements of that concept; if a record excludes one concept, that record should not contain any element of that record. As is natural in set operations, exclusion overrides inclusion. In other words, if a record excludes one concept, any element in that concept is not included in the reconstructed record, even if it is present in any other concept that is included in that record.

The essential task of EBMD is to find a set of concepts and the way of reconstructing the input Boolean matrix with those concepts. Similar to BMD, a concept is represented by a Boolean vector. In BMD, the combinations are represented by a Boolean matrix, where an element of 1 for a record denotes that the

corresponding concept is included, otherwise not. To reflect the set difference operation, we introduce elements of -1. So an EBMD solution of a Boolean matrix $A_{m\times n}$ is in a form of $\{X_{m\times k}, C_{k\times n}\}$, where the concept matrix C is a Boolean matrix and the combination matrix X is in $\{-1, 0, 1\}$ where $x_{ij} = 1$ denotes the jth record includes the ith concept and $x_{ij} = -1$ denotes the jth record excludes the ith concept. In contrast to BMD, we denote EBMD as $A = X \odot C$. The following is the formal set-theoretic EMBD definition:

Definition 5.1 (EBMD) *$\{X \in \{-1, 0, 1\}, C \in \{0, 1\}\}$ is called an EBMD solution of $A \in \{0, 1\}$, denoted by $A = X \odot C$, if $A_i = \cup_{x_{ij}=1} C_j \setminus \cup_{x_{ij}=-1} C_j$, where A_i denotes the item subset corresponding to elements of 1 in the jth row of A and C_j denotes similarly.*

Although the definition of EBMD is intuitive, the $\odot$ operator cannot be directly executed as the $\otimes$ operator of BMD. So we give the following definition of the $\odot$ operator based on logic arithmetic.

Definition 5.2 ($\odot$ operator) *The $\odot$ operator operates over a matrix $X_{m\times k} \in \{-1, 0, 1\}^{m\times k}$ and a matrix $C_{k\times n} \in \{0, 1\}^{k\times n}$. If $A_{m\times n} = X_{n\times k} \odot C_{k\times n}$, we have*

$$\begin{cases} a_{ij=1} & \textit{if } (\exists t_1)\,(x_{i,t_1} = 1 \ \textit{AND} \ c_{t_1,j} = 1) \\ & \textit{AND } \ (\neg\exists\, t_2)\ (x_{i,t_2} = 1\ AND\ c_{t_2,j} = -1) \\ a_{ij=0} & \textit{if } \ (\neg\exists\, t_1)\,(x_{i,t_1} = 1\ AND\ c_{t_1,j} = 1) \\ & \textit{OR } \ (\exists\, t_2)\ (x_{i,t_2} = 1\ AND\ c_{t_2,j} = -1) \end{cases}$$

where $i \in [1, m]$ and $j \in [1, n]$

Note that the $\odot$ and $\otimes$ operators are equivalent when all entries in X are binary.

The EBMD operator is commutative as described as follows.

Property 5.1 (Commutativity) $(X_{m\times k} \odot C_{k\times n})^T = C^T_{k\times n} \odot C^T_{m\times k}$.

The commutativity implies that if $A = C \odot X$ where $C \in \{-1, 0, 1\}$ and $X \in \{0, 1\}$, we have $A^T = X^T \odot C^T$ as well. So EBMD essentially decomposes one Boolean matrix into one Boolean matrix and one matrix in $\{-1, 0, 1\}$, while the order of them does not matter.

Look at Figure 5.4, in which the negative element appears in the second decomposed matrix. To interpret such an EBMD solution, we can consider its inverse as the follows. It gives us another perspective to look at negative permission authorizations in a role. Each "role" r_i can be viewed a set of users. Each permission p_i corresponds to a set of users who are assigned the permission. The EBMD result as shown in Figure 5.3 gives that $p_1 = r_1$, $p_2 = r_2$, $p_3 = r_1 \setminus r_2$, and $p_4 = r_1 \cup r_2$.

$$\begin{pmatrix} & u1 & u2 & u3 & u4 \\ p1: & 1 & 1 & 0 & 1 \\ p2: & 0 & 1 & 1 & 0 \\ p3: & 1 & 0 & 0 & 1 \\ p4: & 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} & r_1 & r_2 \\ p_1 & 1 & 0 \\ p_2 & 0 & 1 \\ p_3 & 1 & -1 \\ p_4 & 1 & 1 \end{pmatrix} \odot \begin{pmatrix} & u1 & u2 & u3 & u4 \\ r1: & 1 & 1 & 0 & 1 \\ r2: & 0 & 1 & 1 & 0 \end{pmatrix}. \tag{5.3}$$

5.3 Semantic Role Mining Problem

We have illustrated how negative authorizations can help identify underlying data semantics and discover a succinct role-based access control system. Negative authorizations can be negative permissions in a role or negative role assignments. However, it does not make sense to have both in a system as it would be difficult

to interpret a negative assignment of a role, which includes a negative permission. So in the access control setting, we limit negative authorization to be only one kind, either negative role assignments or negative permissions.

Roles and role assignments with negative authorizations are called semantic roles and semantic user-role assignments respectively. They are defined as follows.

Definition 5.3 (Semantic Role) *A semantic role r_i is a role consisting of positive permissions P_i^+ and negative permissions P_i^-.*

Definition 5.4 (Semantic Role Assignment) *A semantic role assignment c_i consists of positive role assignments R_i^+ and negative role assignments R_i^-.*

A permission can be assigned to a user both positively and negatively. To resolve such a conflict, we require that a negative permission assignment always overrides a positive permission assignment. In other words, if a permission p_i is negatively assigned to a user, the user can never have that permission, unless the negative assignment of the permission p_i is revoked.

The goal of role mining is to discover a good set of roles. The goodness of a set of roles is usually evaluated by the number of roles. As illustrated before, with negative authorizations, less roles would be needed and some underlying data semantics such as SoD and exception constraints could also be revealed. Therefore, we introduce the semantic role mining problem as follows.

Problem 5.1 (Semantic Role Mining (SRM)) *Given existing user-permission assignments A, discover a RBAC system with the minimum number of roles C (or*

semantic roles) and identify the corresponding user-role assignments X (or semantic role assignments).

SRM is essentially to find an EBMD solution X and C of the Boolean matrix of A. One of X and C is a Boolean matrix and the other is a matrix in $\{-1, 0, 1\}$. Mathematically, SRM can be formulated as an optimization problem as follows.

$$minimize\ k$$

$$s.t.\ A_{m\times n} = X_{m\times k} \odot C_{k\times n}.$$

It is not easy to find such an optimal EBMD solution. We observe that the SRM problem can be broke down into two subproblems: identify one decomposed matrix and then determine the other decomposed matrix. Based on this observation, we propose an alternating approach to solve the SRM problem. For ease of explanation, we describe it in the language of EMBD. The sketch of this alternating approach is depicted as in Algorithm 5.3

Algorithm 5.5 Sketch of An Alternating Approach for EBMD

1: Input: $A \in \{0, 1\}^{m\times n}$
2: Output: $C \in \{-1, 0, 1\}^{m\times k}$ and $X \in \{0, 1\}^{k\times n}$
3: Define an initial value of X and C;
4: **while** The current EBMD solution can be improved **do**
5: Given X, improve C;
6: Given C, improve X;
7: **end while**

Two subproblems arise from this alternating approach. We call them partial SRM I and partial SRM II.

Problem 5.2 (Partial SRM I) *Given original user-permission assignments A and*

regular roles C (or regular user-role assignments X), find semantic user-role assignments X (or semantic roles C), such that $||A - X \odot C||_1$ is minimized.

The partial SRM II can also arise on its own. One possible scenario is that regular roles are given, the system administrator wants to assign fewer roles to each user by employing both positive role assignments and negative role assignments.

Problem 5.3 (Partial SRM II) *Given original user-permission assignment A and semantic roles C (or semantic user-role assignments X), find regular user-role assignment X (or regular C) , such that $||A - X \odot C||_1$ is minimized.*

The partial SRM I problem can arise on its own in a scenario as the following. Recall that semantic roles can be deployed to enforce some SoD polices by introducing negative permissions in roles. Suppose that SoD polices have been enforced and the reflective semantic roles are given. Now we need to assign those roles appropriately to users to match their existing user-permission assignments. This is a partial SRM I problem.

In the language of EBMD, partial SRM problems are essentially given a Boolean matrix and a part of its EBMD solution to find the other part.

Notice that for partial SRM problems, we allow errors instead of requiring exact matching. The rationale is that the role mining phase essentially aims to identifying roles, not finalizing roles. By allowing reconstruction errors, we can obtain a broad picture of role sets. The other important reason is that the input user-permission assignments themselves may contain errors. Exact matching would cause results to be over-fitting.

There are two types of errors, 1 becoming 0 and 0 becoming 1. However, not both are favored in the role mining context. When facing 1 becoming 0 errors, a user can always call help desk to request missing permissions. While, 0 becoming 1 errors can directly harm system safety. Hence the conservative way is to avoid 0 becoming 1 errors in the first place. Therefore, we introduce two variants.

Problem 5.4 (Conservative Partial SRM I) *Given original user-permission assignment A and regular roles C (or regular user-role assignments X), find semantic user-role assignment X (or semantic roles C), such that $||A - X \odot C||_1$ is minimized and $(X \odot C)_{ij} = 0$ if $A_{ij} = 0$.*

Problem 5.5 (Conservative Partial SRM II) *Given original user-permission assignment A and semantic roles C (or semantic user-role assignments X), find regular user-role assignment X (or regular C) , such that $||A - X \odot C||_1$ is minimized and $(X \odot C)_{ij} = 0$ if $A_{ij} = 0$.*

5.4 Theoretical Study

This section studies complexity of presented SRM (EBMD) variants. We start by looking at the partial SRM I problem. Its decision version problem is NP-complete, which can be proven by a reduction to a known NP-complete problem, the decision BU problem [48].

Problem 5.6 (Decision BU) *Given binary matrices $A_{m\times n}$ and $C_{k\times n}$, and a non-negative integer t, is there a binary matrix $X_{m\times k}$ such that $||A - X \otimes C||_1 \leq t$?*

Theorem 5.1 *The decision partial SRM I problem is NP-complete.*

Proof. The decision partial SRM I problem obviously belongs to NP. The decision partial SRM I problem can be polynomially reduced to the decision BU problem. A decision BU instance is a triplet $\{A'_{m\times n}, C'_{k\times n}, t'\}$, where t' is a positive integer. We construct a decision partial SRM II problem instance $\{A, C, t\}$, where A is a $m \times (n + 2t)$ binary matrix where the first $2t$ columns containing all 1's and the remaining m columns are A', and C is a $k \times (n+2t)$ binary matrix where the first $2t$ columns contain all 1's and the remaining n columns are C'. If the solution X for that decision EBU instance consists of cells of -1, $||A - X \otimes C||_1$ must be greater than t. Therefore X can only be in $\{0, 1\}$. When X is limited to be in $\{0, 1\}$, the $\odot$ operator is equivalent to the $\otimes$ operator. Therefore, the decision BU instance is true if and only if the constructed decision EBU instance is true.□

Let us now look at the conservative partial SRM I problem. It is a NP-hard problem as well. For ease of understanding, we would to like to study it in the setting of the red-blue set cover problem (RBSC) [10].

Problem 5.7 (RBSC) *Given a finite set of red elements R and a finite set of blue elements B and a family $\mathcal{S} = \{S_1, ..., S_n\} \in 2^{R\cup B}$, find a subfamily $\mathcal{C} \in \mathcal{S}$ which covers all blue elements, but which covers the minimum possible number of red elements.*

The conservative partial SRM I problem can be viewed as a special variant of the RBSC problem as follows.

Problem 5.8 (Decision Extended RBSC I) *Given disjoint sets R and B of red and blue elements, a collection $\mathcal{S} = \{S_1, ..., S_n\} \in 2^{R \cup B}$, and a nonnegative number t, are there two subcollections $\mathcal{C}^1, \mathcal{C}^2 \subseteq \mathcal{S}$ such that $\bigcup \mathcal{C}^1 \setminus \bigcup \mathcal{C}^2$ covers more than t blue elements, while no red elements are covered?*

Theorem 5.2 *The decision extended RBSC I problem is NP-complete.*

Proof. Given a solution of the decision extended RBSC I problem, it is easy to determine wether it is true or not. Hence the decision extended RBSC problem belongs to NP. Then we will prove that it can be reduced to a known NP-complete problem, the decision RBSC problem. For any instance of the decision RBSC problem $\{R, B, \mathcal{S}, t\}$, we create a corresponding decision extended RBSC I instance $\{R', B', \mathcal{S}', t'\}$, such that:

- for each blue element b_i in B, create a corresponding red element r'_i and include it in R'. Hence, $|B| = |R'|$;
- For each red element r_i in R, create a corresponding blue element b'_i and include it in B'.
- In addition, we create k more blue elements, $\{b'_{|R|+1}, ..., b'_{|R|+k}\}$, where $k \gg |R| + |B|$.
- For each $s_i \in \mathcal{S}$, create s'_i, such that for each b_i in s_i, include the corresponding r'_i in s'_i and for each r_i in s_i, include the corresponding b'_i in s'_i. Include s'_i in $\mathcal{S}'$

- Create a subset of $s'_{|\mathcal{S}|+1}$, such that it contains all blue and red elements. In other words,
  $s'_{|\mathcal{S}|+1} = R \cup B \cup \{b'_{|R|+1}, ..., b'_{|R|+k}\}$.
- Let $\mathcal{S}' = \{s'_1, ..., s'_{|\mathcal{S}|}\} \cup s_{|\mathcal{S}|+1}$.
- Let t'=t.

Illustration to the instance construction. Suppose the decision RBSC instance is $\{\{b_1, b_2, r_1\}, \{b_1, r_1, r_2\}, \{b_2, b_3, r_2\}\}$ and $t = 1$. The constructed extended RBSC II instance is $\{\{r'_1, r'_2, b'_1\}, \{r'_1, b'_1, b'_2\}$,
$\{r'_2, r'_3, b'_2\}, \{r'_1, r'_2, b'_1, b'_2, b'_3, b'_4, b'_5, b'_6, b'_7, b'_8\}\}$.

Because $s'_{|\mathcal{S}|+1}$ contains k new blue elements, which do not belong to any subset, and $k \gg |R| + |B|$, the optimal solution should be that $s'_{|\mathcal{S}|+1}$ excluding a subcollection of $\{s'_1, ..., s'_{|\mathcal{S}|}\}$, such that the subcollection covers the minimum blue elements while covers all red elements. As the blue and red elements in $\mathcal{S}'$ are corresponding to the red and blue elements in $\mathcal{S}$ respectively, the decision RBSC instance is true, if and only if the constructed decision extended RBSC II instance is true. As decision extended RBSC II belongs to NP, it is NP-complete. □

The above proof naturally leads to the following conclusion.

Theorem 5.3 *The decision conservative partial SRM I is NP-complete.*

We now look at the partial SRM II problem. Its decision problem can be proven by a reduction the decision BU problem as well.

Theorem 5.4 *The problem of decision partial SRM II is NP-complete.*

Proof. Given user-role assignments UA, it is easy to determine whether $||UPA - UA \otimes PA||_1 \leq t$ is satisfied or not. The determination can be done in a polynomial time. So the problem of decision partial SRM II belongs to NP. Next we will build a mapping from the decision BU problem to the decision partial SRM II problem. For every instance of the decision BU problem, $\{A, X, t\}$, we can create a corresponding instance of the problem of decision partial SRM II, $\{UPA, PA, t\}$, such that $UPA = A$ and $PA = X$. Because there is no cell with the value of -1 in PA and the partial SRM II problem requires UA to be all positive assignments, we have $UA \odot PA = UA \otimes PA$. So the instance of $\{A, X, t\}$ is true if and only if the instance of $\{UPA, PA, t\}$ is true. Hence, the problem of decision partial SRM II is NP-complete.□

Then we study the conservative partial SRM II problem. Before we give the NP-complete proof, we introduce one known NP-complete problems, Positive-Negative Partial Set Cover (±PSC) [48].

Problem 5.9 (±PSC) *Given disjoint sets P and N of positive and negative elements, respectively and a collection $\mathcal{S}$ of subsets of $P \bigcup N$, find a subcollection $\mathcal{C} \in \mathcal{S}$ minimizing $|P \backslash (\cup \mathcal{C}|) + |N \cap (\cup \mathcal{C}|)$.*

The ±PSC problem is extended from the RBSC problem. If we replace positive and negative elements with blue and red elements respectively, the ±PSC problem becomes to find a subfamily $\mathcal{C} \in \mathcal{S}$ which minimizes the sum of uncovered blue elements and covered red elements. Hence, the only difference between

RBSC and ±PSC is that the RBSC problem requires all blue elements to be covered.

To prove the decision version of conservative partial SRM IU is complete, we will relate it to a special case of ±PSC, where the number of positive elements is same as the number of negative elements. We simply call it equal ±PSC.

Problem 5.10 (Equal ±PSC) *Given disjoint sets P and N of positive and negative elements respectively, where $|P| = |N|$, and a collection $\mathcal{S}$ of subsets of $P \bigcup N$, find a subcollection $\mathcal{C} \in \mathcal{S}$ minimizing $|P \backslash (\cup \mathcal{C})| + |N \cap (\cup \mathcal{C})|$.*

Theorem 5.5 *The decision equal ±PSC problem is NP-complete.*

Proof. Equal ±PSC is a special case of ±PSC. Obviously, it belongs to NP. Next, we will show that for every instance of decision ±PSC, we can find a corresponding decision equal ±PSC instance. Given a decision ±PSC instance as $\{P, N, \mathcal{S}, t\}$, we create a corresponding equal ±PSC instance $\{P', N', \mathcal{S}', t'\}$ such that:

- if $|P| < |N|$
  - ▷ Introduce $|N| - |P|$ new positive elements, $\{p'_{|P|+1}, ..., p'_{|N|}\}$. Let $P' = P \cup \{p'_{|P|+1}, ..., p'_{|N|}\}$
  - ▷ Let $N' = N$.
  - ▷ For every subset $s_i \in \mathcal{S}$, create a subset s'_i such that $s'_i = s_i \cup \{p'_{|P|+1}, ..., p'_{|N|}\}$ and include it in $\mathcal{S}'$. So $\mathcal{S}' = \{s'_1, ..., s'_{|\mathcal{S}|}\}$.

- ▷ $t' = t$.

- else if $|P| > |N|$

  - ▷ Introduce $|P| - |N|$ new negative elements, $\{n'_{|N|+1}, ..., n'_{|P|}\}$. Let $N' = N \cup \{n'_{|N|+1}, ..., n'_{|P|}\}$
  - ▷ Let $P' = P$.
  - ▷ For every subset $s_i \in \mathcal{S}$, create a subset s'_i such that $s'_i = s_i \cup \{n'_{|N|+1}, ..., n'_{|P|}\}$ and include it in $\mathcal{S}'$. So $\mathcal{S}' = \{s'_1, ..., s'_{|\mathcal{S}|}\}$.
  - ▷ $t' = t + (|N| - |P|)$.

- else

  - ▷ $P' = P$; $N' = N$; $\mathcal{S}' = \mathcal{S}$; $t' = t$.

Consider the case of $|P| < |N|$. If the $\pm$PSC instance, $\{P, N, \mathcal{S}, t\}$, is true, there exists a subcollection $\mathcal{C} \in \mathcal{S}$ such that $|P \backslash (\cup \mathcal{C})| + |N \cap (\cup \mathcal{C})| < t$. We can find a subcollection $\mathcal{C}' \in \mathcal{S}'$ corresponding to $\mathcal{C} \in \mathcal{S}$. As $\{p'_{|P|+1}, ..., p'_{|N|}\}$ belong to any subset in $\mathcal{C}'$, we have $|P' \backslash (\cup \mathcal{C}')| = |P \backslash (\cup \mathcal{C})|$. It is obvious true that $|N' \cap (\cup \mathcal{C}')| = |N \cap (\cup \mathcal{C})|$. So we have $|P' \backslash (\cup \mathcal{C}')| + |N' \cap (\cup \mathcal{C}')| < t$. In the other way,if the decision equal $\pm$PSC instance is true, the $\pm$PSC instance must be true.

For $|P| > |N|$, as new negative elements $\{n'_{|N|+1}, ..., n'_{|P|}\}$ are added for each subset, we have $|N' \cap (\cup \mathcal{C}')| = |N \cap (\cup \mathcal{C}) + (|N| - |P|)|$. It is true that $\{p'_{|P|+1}, ..., p'_{|N|}\} = \{p_{|P|+1}, ..., p_{|N|}\}$. Hence, we have $|P \backslash (\cup \mathcal{C})| + |N \cap (\cup \mathcal{C})| = |P \backslash (\cup \mathcal{C})| + |N \cap (\cup \mathcal{C})| + (|N| - |P|)$. Therefore, the decision $\pm$PSC instance is true if and only if the equal $\pm$PSC instance is true.

For the case of $|P| = |N|$, both instances are equivalent. □

Next we will prove the decision version of conservative partial SRM I is NP-complete by relating it to the decision equal ±PSC problem.

Theorem 5.6 *The problem of decision conservative partial SRM II is NP-complete.*

Proof. A decision conservative partial SRM II instance is a triplet $\{UPA, PA, t\}$. Given a solution UA, it is easy to check whether $||UPA - UA \odot PA||_1 \leq t$ is true or not. So the decision conservative partial SRM II problem belongs to NP. Next, we will reduce it to the known NP-complete problem, decision equal ±PSC. For any decision ±PSC instance $\{P, N, \mathcal{S}, t\}$, we create a corresponding decision conservative partial SRM II instance $\{UPA, PA, t'\}$, such that:

- Let $|Permissions| = |P|$ (or $|N|$) and each permission correspond to an element.
- For each subset s_i of $\mathcal{S}$, create a role, which corresponds to a row in PA, such that:
  - ▷ if s_i contains n_j, $PA(i, j) = -1$.
  - ▷ if s_i contains p_j and excludes n_j, $PA(i, j) = 1$.
  - ▷ if s_j has neither p_j, nor n_j, $PA(i, j) = 0$.
- Let UPA be a single row with all elements being 1.
- Let t'=t.

Illustration to the instance construction: Suppose a RBSC instance is as $\{\{p_1, p_2, n_1\}, \{p_2, n_2\}, \{p_1, n_2\}\}$, with only two blue elements and two red elements. The corresponding instance of conservative partial SRM I is : $UPA = (1, 1)$ and $PA = \begin{pmatrix} \text{-1} & 0 \\ 0 & \text{-1} \\ 1 & \text{-1} \end{pmatrix}$.

Recall the definition of semantic roles that if a user is assigned to a role with negative permission i, he can never have that permission, even though he is assigned to other roles with positive permission i. The above mapping actually correspond positive element p_i to positive permission i, negative element n_i to negative permission i, and UPA to all positive elements. Further as we need to cover UPA, which contains all permissions, there will never be 0-1 errors. Hence, we do not even need to consider it for such constructed instances. Suppose a subset of roles are selected, which gives UA. The total number of permissions assigned to the user is:

$$\#(covered\ positive\ permissions) - \#(covered\ negative\ permissions)$$

Hence, the difference between UPA and real assignment is

$$\begin{array}{ll} & |Permissions| - (\#(covered\ positive\ permissions) - \#(covered\ negative\ permissions)) \\ = & |Permissions| - \#(covered\ positive\ permissions) + \#(covered\ negative\ permissions) \\ = & \#(uncovered\ positive\ permissions) + \#(covered\ negative\ permissions) \\ = & P \backslash (\cup \mathcal{C}| + |N \cap (\cup \mathcal{C}|. \end{array}$$

It shows that the equal $\pm$PSC instance is true if and only if the constructed partial SRM I instance is true. □

5.5 Mathematical Programming Formulation

We introduced SRM, partial SRM I and II, and their 0-1 error free and 1-0 error free variants. Although they are NP-hard, standard mathematical programming

software packages can still solve small or even medium scale problems. In this section, we will provide mixed integer program formulation for all presented SRM (EBMD) variants.

We start by looking at SRM, which is given observed binary data to find a EBMD solution with the minimum size. For ease of explanation, let us assume the case of regular roles and semantic user-to-role assignments. The problem becomes to find a EBMD decomposition $\{X_{m\times k}, C_{k\times n}\}$ of $A_{m\times n}$, where C is a matrix in $\{0, 1\}$ and X is a matrix in $\{-1, 0, 1\}$. The EBMD solution space is huge. To further ease the problem, we assume a set of candidate roles C are given. The problem then becomes to finding a minimum candidate role set to describe the observed data A.

$$min \sum_{k \in K} y_k \tag{5.4}$$
$$s.t$$

$$\sum_{k \in K s.t. a_{ij}=1} x_{ik}^{+} c_{kj} \geq 1,\ \forall i \in M,\ j \in N \tag{5.5}$$

$$\sum_{k \in K s.t. a_{ij}=1} x_{ik}^{-} c_{kj} = 0,\ \forall i \in M,\ j \in N \tag{5.6}$$

$$\sum_{k \in K s.t. a_{ij}=0} x_{ik}^{+} c_{kj} \leq t_{ij} B,\ \forall i \in M,\ j \in N \tag{5.7}$$

$$\sum_{k \in K s.t. a_{ij}=0} x_{ik}^{-} c_{kj} \geq 1 - (1 - t_{ij}) M,\ \forall i \in M,\ j \in N \tag{5.8}$$

$$x_{ik}^{+} + x_{ik}^{-} \leq 1,\ \forall k, j \tag{5.9}$$

$$y_k \geq x_{ik}^{+},\ \forall k \in K,\ i \in M \tag{5.10}$$

$$y_k \geq x_{ik}^{-},\ \forall k \in K,\ i \in M \tag{5.11}$$

$$t_{ij} \in \{0, 1\},\ \forall i \in M, j \in N \tag{5.12}$$

$$x_{ik}^{+}, x_{ik}^{-} \in \{0, 1\},\ \forall k \in K, i \in M \tag{5.13}$$

The formulated mixed integer program is as Equations (5.4-5.13). Descriptions about the mixed linear program are given as follows:

- c_{kj} is binary and given. $c_{kj} = 1$ means the candidate role k contains the permission j; otherwise not.
- x_{ik}^{+} and x_{ik}^{-} are binary variables to to be determined. $x_{ik}^{+} = 1$ means the candidate role k is positively assigned to the user i; otherwise not. x_{ik}^{-} means the candidate role k is negatively assigned to the user i; otherwise not.

- Equation 5.5 ensures that when $a_{ij} = 1$, at least one candidate role which contains the permission j is positively assigned to the user i.
- Equation 5.6 ensures that when $a_{ij} = 1$, no candidate role which contains the permission j is negatively assigned to the user i.
- t_{ij} is a binary auxiliary variable and B is a large enough constant. Equations 5.7 and 5.8 ensure that when $a_{ij} = 0$, either no role containing the permission j is positively assigned to the user i, or some role containing the permission j is negatively assigned to the user i.
- Equation 5.9 ensures that a role is assigned positively or negatively.
- y_k is a binary variable. When it is 1, the role k is employed; otherwise not. Equations 5.10 and 5.11 ensures that if some role k has been employed regardless of in a positive way or in a negatively way, y_k is 1.
- The objective function $\sum_k y_k$ is to minimize the number of selected roles.

Partial SRM I is given A and C to find X minimizing $\sum_{ij} |a_{ij} - (A \odot C)_{ij}|$. It is very similar to the SRM problem with candidate roles being given. The mixed integer program formulation for partial SRM I is as Equations (5.14 - 5.25), which is quite similar to the above formulation for SRM I. Descriptions on differences are given as follows:

- In Equations (5.15 - 5.18), auxiliary non-negative variables u_{ij}^{+} and $u^{-}ij$ are utilized to allow inexactness. If one of u_{ij}^{+} and $u^{-}ij$ is positive, there is an assignment error of the permission j to the user j. It could be over-assignment or under-assignment.

- v_{ij} is a binary variable, indicating whether there is an assignment error at a_{ij}. Equation 5.22 ensures that if there is an assignment error of the permission j to the user j, $v_{ij} = 1$.

- For the descriptions of the rest constraints, refer to the preceding descriptions of the mixed integer formulation of SRM.

$$min \sum_{i \in M} v_{ij} \tag{5.14}$$

$$s.t.$$

$$\sum_{k \in K s.t. a_{ij}=1} x_{ik}^{+} c_{kj} + u_{ij}^{+} \geq 1,\ \forall i \in M,\ j \in N \tag{5.15}$$

$$\sum_{k \in K s.t. a_{ij}=1} x_{ik}^{-} c_{kj} - u_{ij}^{-} = 0,\ \forall i \in M,\ j \in N \tag{5.16}$$

$$\sum_{k \in K s.t. a_{ij}=0} x_{ik}^{+} c_{kj} - u_{ij}^{+} \leq t_{ij} B,\ \forall i \in M,\ j \in N \tag{5.17}$$

$$\sum_{k \in K s.t. a_{ij}=0} x_{ik}^{-} c_{kj} + u_{ij}^{-} \geq 1 - (1 - t_{ij}) M,\ \forall i \in M,\ j \in N \tag{5.18}$$

$$x_{ik}^{+} + x_{ik}^{-} \leq 1,\ \forall k, j \tag{5.19}$$

$$y_k \geq x_{ik}^{+},\ \forall k \in K,\ i \in M \tag{5.20}$$

$$y_k \geq x_{ik}^{-},\ \forall k \in K,\ i \in M \tag{5.21}$$

$$u_{ij}^{+} + u_{ij}^{-} \leq v_{ij} B,\ \forall i \in M,\ j \in N \tag{5.22}$$

$$t_{ij} \in \{0, 1\},\ \forall i \in M, j \in N \tag{5.23}$$

$$x_{ik}^{+}, x_{ik}^{-} \in \{0, 1\},\ \forall k \in K, i \in M \tag{5.24}$$

$$u_{ij}^{+}, u_{ij}^{-} \geq 0, \forall i \in M, j \in N \tag{5.25}$$

$$\tag{5.26}$$

SRM II is given A and X to find C. The mixed integer program formulation for SRM II is the same as Equations (5.14 - 5.25). But note that X, a matrix in $\{-1, 0, 1\}$, is represented by two Boolean matrix X^+ and X^-. If the user i is assigned to the role j positively, $x_{ij}^+ = 1$. If it is a negative assignment, $x_{ij}^- = 1$.

5.6 Algorithm Design

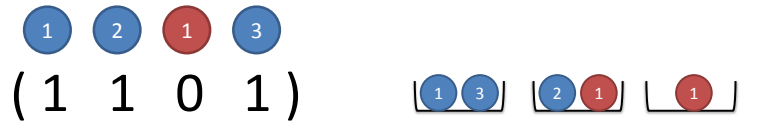
5.6.1 Partial SRM I

In the language of EBMD, the partial SRM I problem is given a Boolean matrix A and a concept matrix C to find the combination matrix $X \in \{-1, 0, 1\}$ such that $||A - X \odot C||_1$ is minimized. As $||A - X \odot C||_1 = \sum_i ||A_i - X_i \odot C||_1$, where A_i and X_i denote the ith row of A and X respectively, a partial SRM I problem can be divided into a set of subproblems with each row of A as an input Boolean matrix. So without loss of generality, we consider A to be a Boolean row vector. As a result of that, the partial SRM I problem can be described as a variant of the RBSC problem as follows.

Consider the following partial SRM I problem, where the variables $\{x_1, x_2, x_3\}$ need to be determined.

$$\begin{pmatrix} 1 & 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \odot \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{5.27}$$

Let each column correspond to a distinct element, where the columns at which the elements of the input Boolean vector are 1, correspond to blue elements and the remaining rows correspond to red elements. Then, the first, second and fourth columns correspond to blue elements $\{b_1, b_2, b_3\}$ respectively and the third column corresponds to the red element $\{r_1\}$, as illustrated in Figure 5.5a. Consequently,



(a) Columns Mapped to Elements

(b) Concepts Mapped to Baskets

Figure 5.5. Mapping Illustration I

the given concept matrix C can be mapped a collection $\mathcal{C}$ of subsets of red-blue elements $\{B \cup R\}$, where B and R denote the blue element set and the red element set respectively, such that $\{\{b_1, b_3\}, \{b_2, r_1\}, \{r_1\}\}$, as illustrated in Figure 5.5b. Hence, the partial SRM I problem becomes:

- *Given a collection $\mathcal{C}$ of subsets of red-blue elements $\{B \cup R\}$, find two subcollections $\mathcal{C}_1$ and $\mathcal{C}_2$ such that $(\cup\mathcal{C}_1)\backslash(\cup\mathcal{C}_2)$ maximizes*

$$\#(covered\ blue\ elements) - \#(covered\ red\ elements).$$

Denote baskets from left to right in Figure 5.5b to be c_1, c_2, and c_3 respectively. It is not difficult to see that the optimal solution is $(c_1 \cup c_2)\backslash c_3$. Therefore $\{x_1, x_2, x_3\} = (1, 1, -1)$.

As proven in the previous section, the decision partial SRM I problem is NP-complete in general. So we propose an efficient and effective greedy heuristic. We first divide the given subset collection $\mathcal{C}$ into three groups $\{\mathcal{C}^B, \mathcal{C}^R, \mathcal{C}^{B,R}\}$, where $\mathcal{C}^B$ includes subsets containing blue elements only, $\mathcal{C}^R$ consists of red elements only and the subsets in $\mathcal{C}^{B,R}$ have both red and blue elements. Obviously, including $\mathcal{C}^B$ in $\mathcal{C}_1$ and $\mathcal{C}^R$ in $\mathcal{C}_2$ dose not introduce any covering error. Since $\mathcal{C}^R$ has been included in $\mathcal{C}_2$, assigning any subset of $\mathcal{C}_2$, in which all contained red elements belong to $\mathcal{C}^R$, to $\mathcal{C}_1$ does not introduce covering error either. Next, we

need to assign the remaining subsets of $\mathcal{C}^{B,R}$ to either $\mathcal{C}_1$ or $\mathcal{C}_2$. We will do it in an iterative fashion. At each step we select a subset c from the remaining $\mathcal{C}^{B,R}$ and put it in $\mathcal{C}_1$. The selection criteria is based on the following function:

$$f_1 = \frac{\#(Newly\ Covered\ Blue)}{\#(Newly\ Covered\ Red)}. \tag{5.28}$$

The numerator part in the criteria function denotes the number of newly covered blue elements by including c, while the denominator part denotes the number of newly covered red elements. We iteratively select the subset with the greatest criteria value till all blue elements are covered in the hope that the least red elements would be included in the final solution. The complete algorithm is as described in Algorithm 5.6.

Algorithm 5.6 Partial SRM I

Input: A collection $\mathcal{C}$ of subsets of $\{B \cup R\}$.
Output: Two subcollections $\mathcal{C}_1$ and $\mathcal{C}_2$.

1: Divide $\mathcal{C}$ into $\{\mathcal{C}^B, \mathcal{C}^R, \mathcal{C}^{B,R}\}$
2: Include $\mathcal{C}^B$ in $\mathcal{C}_1$, and $\mathcal{C}^R$ in $\mathcal{C}_2$;
3: Update $\mathcal{C}^{B,R}$ by deleting elements contained in $\mathcal{C}^B$ and $\mathcal{C}^R$;
4: Set $B' = \cup\mathcal{C}^{B,R} \cap B$ and $R' = \emptyset$;
5: **while** The objective value can be further improved. **do**
6: Select the subset $c \in \mathcal{C}^{B,R}$ with the largest f_1 value and include it in $\mathcal{C}_1$;
7: Update B' as $B' \backslash (c \cap B)$, and R' as $R' \cup (c \cap R)$.
8: **end while**

5.6.2 Conservative Partial SRM I

The conservative partial SRM I problem requires no 0 becoming 1 errors. In the red-blue set cover problem setting, it can be described as follows:

- *Given a collection $\mathcal{C}$ of subsets of red-blue elements $\{B \cup R\}$, find two subcollections $\mathcal{C}_1$ and $\mathcal{C}_2$ such that $(\cup\mathcal{C}_1)\backslash(\cup\mathcal{C}_2)$ maximizes $\#(covered\ blue\ elements)$,*

while no red elements are covered.

Its decision problem has been proven to be NP-complete. Similar to the partial SRM I problem, we propose a greedy heuristic for its conservative version as described in Algorithm 5.7. Notice that the first two steps are the same as that for the partial SRM I problem. After the first two steps, only subsets in $\mathcal{C}\{B, R\}$ remain. For each remaining subset c, if its contained elements are already included in $\mathcal{C}_2$, we include it in $\mathcal{C}_1$, as it will not introduce any red element in the final solution.

Algorithm 5.7 Conservative Partial SRM I

Input: A collection $\mathcal{C}$ of subsets of $\{B \cup R\}$.
Output: Two subcollections $\mathcal{C}_1$ and $\mathcal{C}_2$.
1: Divide $\mathcal{C}$ into $\{\mathcal{C}^B, \mathcal{C}^R, \mathcal{C}^{B,R}\}$
2: Include $\mathcal{C}^B$ in $\mathcal{C}_1$ and $\mathcal{C}^R$ in $\mathcal{C}_2$
3: **for** each $c \in \mathcal{C}^{B,R}$ **do**
4: **if** $c \cap R \in \mathcal{C}^R$ **then**
5: Include c in $\mathcal{C}_1$.
6: **end if**
7: **end for**

5.6.3 Partial SRM II

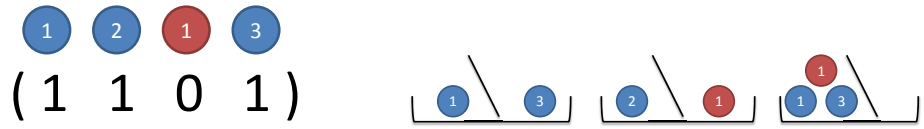
The partial SRM II problem is given a Boolean matrix A and a combination matrix X in $\{-1, 0, 1\}$, to find a concept matrix C, such that $||A - X \odot C||_1$ is minimized. As the $\odot$ operator has the commutative property, the objective function can be rewritten as $||A^T - C^T \odot X^T||_1$, which basically changes the order of C and X. So we can view X as the concept matrix, while concepts may contain negative elements. As $||A^T - C^T \odot X^T||_1 = \sum_i(||A_i^T - (C^T)_i \odot X^T||_1)$, then the original problem can be divided into a set of subproblems with each row of A^T as the input

data. Therefore without loss of generalization, we consider A^T as a Boolean row vector.

$$\begin{pmatrix} 1 & 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \odot \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}. \tag{5.29}$$

For ease of explaining our algorithm later, we firstly look at an example as shown in Equation 5.29. The row vector on the left and the matrix on the right are the input data. $\{x_1, x_2, x_3\}$ are Boolean variables to be determined. This partial SRM II problem can be also viewed as a variant of the red-blue set cover problem. First, we map columns to red-blue elements. The mapping policy is the same as what we did for the partial SRM I problem. The mapping result is as shown in Figure 5.6a. Now we will map each row vector in the concept matrix on the right side of Equation 5.29 to a basket of red-blue elements. Notice that each row vector may contain three different component values {-1,0, 1}. The value of 1 corresponds to the set union operation, while the the value of -1 corresponds to the set different operation. To reflect that, we map each row vector to a special red-blue element basket in the form of $c^+ \backslash c^-$, where both c^+ and c^+ are red-blue element subsets. Based on this mapping rule, the row vectors in the example of Equation 5.29 are mapped to baskets as illustrated in Figure 5.6b, where the symbol of $\backslash$ denotes the set difference operator. The partial SRM II problem can be described as follows:

- *Given a basket set of* $\{\{c_1^+ \backslash c_1^-\}, ..., \{c_k^+ \backslash c_k^-\}\}$*, where* $\{c_i^+, c_i^-\}$ *are red-blue element subsets, select a basket subset* $\mathcal{S}$ *such that* $(\cup_{i \in \mathcal{S}} c_i^+) \backslash (\cup_{i \in \mathcal{S}} c_i^-)$ *maximizes* $\#(covered\ blue\ elements) - \#(covered\ red\ elements)$.



(a) Columns Mapped to Elements

(b) Concepts Mapped to Baskets

Figure 5.6. Mapping Illustration II

For the example of Figure 5.6b, it is not difficult to see that the optimal solution is to select the second and third baskets. The result covers all blue elements without introducing a red elements. However, the partial SRM II problem in general has been proven to be NP-hard in the previous section. So we propose a greedy heuristic. Its basic idea is to iteratively select the best remaining basket based on some selection criteria. We observe that four cases may occur when including a basket into the solution: (1) new blue elements being covered; (2) new red elements being covered; (3) new blue elements being excluded; (4) new red elements being excluded. Obviously the first and the fourth cases are desired while the other two cases are disliked. So our selection criteria is based on the function in Equation 5.30 and the greedy heuristic is described in Algorithm 5.8.

$$f_2 = \frac{\#(Newly\ Covered\ Blue) + \#(Newly\ Excluded\ Red)}{\#(Newly\ Covered\ Red) + \#(Newly\ Excluded\ Blue)}. \tag{5.30}$$

Algorithm 5.8 Partial SRM II

Input: A red-blue element basket set of $\{\{c_1^+ \backslash c_1^-\}, ..., \{c_k^+ \backslash c_k^-\}\}$.
Output: A red-blue element basket subset $\mathcal{S}$.

1: Iteratively include the basket with the greatest f_1 value into $\mathcal{S}$ till the objective value cannot be improved.

5.6.4 Conservative Partial SRM II

The conservative partial SRM II differs from the partial SRM II problem only in the objective value. It can be similarly studied in the setting of the red-blue set cover problem as follows:

- *Given a basket set of* $\{\{c_1^+ \backslash c_1^-\}, ..., \{c_k^+ \backslash c_k^-\}\}$*, where* $\{c_i^+, c_i^-\}$ *are red-blue element subsets, select a basket subset* $\mathcal{S}$ *such that* $(\cup_{i\in\mathcal{S}} c_i^+)\backslash(\cup_{i\in\mathcal{S}} c_i^-)$ *maximizes* $\#(covered\ blue\ elements)$ *while no red elements are covered.*

As the objective is to cover as many blue elements as possible, for simplicity we only consider baskets $\{c_i^+ \backslash c_i^1\}$ with $c_i^1 \subseteq R$ only. The intuition behind is not to exclude any blue elements in the final solution. However, if we select all such baskets, red elements may be included. To eliminate red elements, we remove troubling baskets in an iterative fashion. At each step, we delete the basket which reduces the number of covered red elements the most. The complete description is provided in Algorithm 5.9.

Algorithm 5.9 Conservative Partial SRM II

Input: A red-blue element basket set of $\{\{c_1^+ \backslash c_1^-\}, ..., \{c_k^+ \backslash c_k^-\}\}$.
Output: A red-blue element basket subset $\mathcal{S}$.

1: For each basket $\{c_i^+ \backslash c_i^-\}$, if $c_i^- \subseteq R$, include it into $\mathcal{S}$.
2: Iteratively remove the basket $\{c_i^+ \backslash c_i^-\}$ from $\mathcal{S}$, which reduces the number of covered elements the most, till no red elements being covered.

5.7 Experimental Study

In this section, we present extensive experimental results on both synthetic and real data sets to validate the performance of our proposed heuristics. Our algorithms are compared on one hand against standard matrix decomposition, and on

the other hand against conventional Boolean matrix decomposition. The algorithm computing standard matrix decomposition is SVD, which is a benchmark for computing standard matrix decomposition. As SVD returns results of real values, to be fair, we round them to be binary by setting all values less than 0.5 to 0, and all other values to 1. Hence, this algorithm is called SVD 0/1. The algorithm computing conventional Boolean matrix decomposition is the Loc& IterX algorithm proposed in [48], which has been experimentally proven to have better performance than other Boolean matrix decomposition algorithms.

5.7.1 Synthetic Data

We study the behavior of the heuristics with respect to the decomposition size and noise.

The synthetic data is generated as follows. First, generate k Boolean vectors randomly as basis, each of which has 50 elements and about 1/3 elements are 1. Second, use basis vectors to generate other $100 - k$ vectors. The detailed procedures are : (i) randomly selecting $k/3$ basis vectors; (ii) randomly assigning half selected basis vectors to $\mathcal{C}_1$ and the other half to $\mathcal{C}_2$; (iii) computing $\cup\mathcal{C}_1 \setminus \cup\mathcal{C}_2$ as a generated vector. The size of such a generated matrix is 50×100. After that, add noise to the matrix by randomly flipping the values of a given fraction of the data.

To compare reconstruction error, we use two kinds of measure. The first is $ER1 = ||A - A'||_1/size(A)$, where A is the input matrix and A' is the reconstructed matrix. The second is $ER2 = ||A - A'||_1/||A||_1$. $ER1$ reflects how much fraction of the data is not correctly reconstructed. $ER2$ is to compare the

amount of reconstruction error against the total number of 1's cells. The reason of employing $ER2$ is that if the input Boolean matrix is sparse, a low value of $ER1$ does not demonstrate the input matrix is correctly reconstructed, as simply returning a matrix of zeros would have a low value of $ER1$.

The first experiment is to test the effect of size k with respect to reconstruction error. We vary k from 4 to 20 and for each size we generate 5 matrices. Reported results are mean values over these five matrices. For algorithms of Loc&IterX, EBMD and 0-1 error free EBMD we use basis vectors as C.

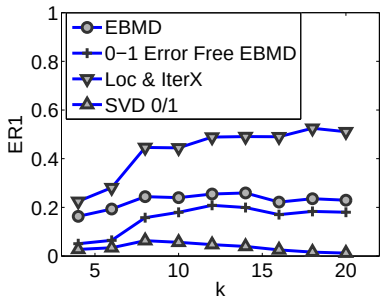


Figure 5.7. Reconstruction Error Ratio with $ER1$ w.r.t. k

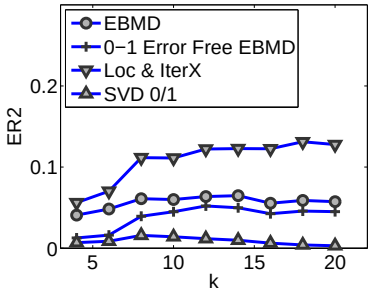


Figure 5.8. Reconstruction Error Ratio with $ER1$ w.r.t. k

The experimental results are as shown in Figures 5.7 and 5.8. As we can see, the reconstruction error ratios of all three EBMD approaches are lower than those of Loc&IterX and close to those of SVD 0/1. With $ER1$, the reconstruction error

ratios of EBMD approaches are as low as 0.05 on average. With $ER2$, they are still as low as 0.2 on average.

The second experiment is to test the effect of noise with respect to reconstruction error. We vary noise ratio from 0 to 0.5. The experimental results are as shown in Figures 5.9 and 5.10. The reconstruction error ratios of EBMD and 1-0 error free EBMD are still lower than those of Loc&IterX and close to SVD 0/1. However, the reconstruction error ratios of 0-1 error free EBMD are much higher than other approaches even though the amount of noise is little.

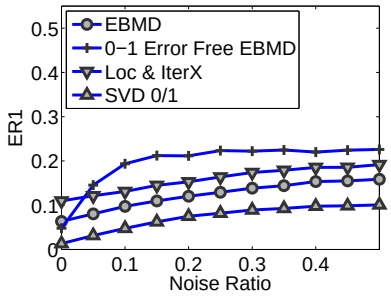


Figure 5.9. Factorization Cost w.r.t. Reconstruction Error Ratio for Synthetic Data

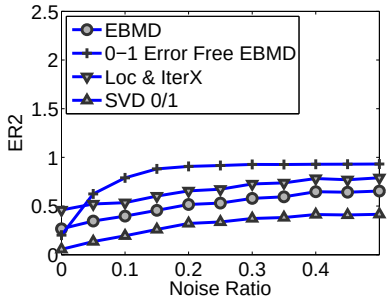


Figure 5.10. Factorization Cost w.r.t. Reconstruction Error Ratio for Synthetic Data

5.7.2 Real Data

One of our main contributions that we claim is that our heuristic algorithm can decompose a Boolean matrix with much less reconstruction error than the conventional Boolean matrix decomposition method. In this section, we demonstrate that the claim holds for real data as well.

Four real data sets are used. The News data set is a subset of 20 Newsgroups dataset [1]. We select the first 400 messages and the top frequent 100 words in them, and replace the counts with 1 or 0. Then, we obtain a 100×400 matrix, which happens to have no repetitive columns. Votes dataset [2] contains plenary votes of Finnish Parliament. Same as [48], we only consider those MPs that served an entire term. During 1999-2001, there were 773 plenary votes and 196 MPs served the entire term. As an MP can cast four different types of votes (yea, Nay, Abstain, and Absent), two different dataset are actually used: VotesYes sets Yeas as 1s and all other votes are 0s, while VotesNo sets Nays as 1's and all other votes as 0's. The Query data set is a user/clicked URL binary matrix, extracted from a large-scale query log. The query log data include two important attributes, UserID (the identity query issuer), and ClickedURL (the URL eventually clicked by that user in that single query). We have first selected the top 40 frequent clicked URLs from the query log with 1,889,761 queries in total and removed all the queries that are not related to those 40 Clicked URLs (we thus obtain 196,218 queries with 40 clicked URLs). Consequently, we have generated another dimension of the matrix by choosing the top 1000 users who have executed most queries in this

[1] http://people.csail.mit.edu/jrennie/20Newsgroups/

[2] http://www.fsd.uta.fi/english/data/cagalogue/FSD2117/meF2117e.html

Data	k	EBMD	0-1 EBMD	Loc& IterX	SVD 0/1
News	5	0.2575	0.3966	0.2811	0.2085
	10	0.2350	0.3791	0.2633	0.1750
	15	0.2200	0.3647	0.2397	0.1457
VotesNo	5	0.0777	0.1921	0.0830	0.0642
	10	0.0704	0.1860	0.0763	0.0496
	15	0.0684	0.1755	0.0722	0.0398
VotesYes	5	0.1531	0.6491	0.1613	0.0779
	10	0.1334	0.6421	0.1459	0.0929
	15	0.1244	0.6320	0.1336	0.0775
Query	5	0.1162	0.1220	0.1168	0.0892
	10	0.0921	0.0974	0.1071	0.0536
	15	0.0710	0.0812	0.1451	0.0301

Table 5.1. Reconstruction Error Ratios with $ER1$ for real datasets

small group of query log. The result is a binary matrix with the dimension of 40*1000. After deleting repetitive columns, finally we have a matrix of 40×200.

The experimental results are shown in Tables 5.1 and 5.2. We can see that the heuristic of EBMD decomposes real datasets with less reconstruction errors than Loc&IterX and close to SVD 0/1, while decomposition solutions provided by the heuristics for 0-1 EBMD have higher reconstruction error ratios.

Data	k	EBMD	0-1 EBMD	Loc& IterX	SVD 0/1
News	5	0.6154	0.9477	0.6718	0.4982
	10	0.5616	0.9058	0.6293	0.4181
	15	0.5258	0.8714	0.5728	0.3482
VotesNo	5	0.3829	0.9464	0.4091	0.3161
	10	0.3471	0.9164	0.3761	0.2445
	15	0.3368	0.8647	0.3557	0.1963
VotesYes	5	0.2311	0.9799	0.2435	0.1176
	10	0.2014	0.9692	0.2203	0.1403
	15	0.1878	0.9540	0.2017	0.1170
Query	5	0.6961	0.7305	0.6992	0.5340
	10	0.5516	0.5831	0.6415	0.3211
	15	0.4251	0.4865	0.4925	0.1804

Table 5.2. Reconstruction Error Ratios with $ER2$ for real datasets

CHAPTER 6
RANK-ONE BOOLEAN MATRIX DECOMPOSITION

Rank-one Boolean matrix decomposition is a special BMD case, where the decomposed matrices are limited to Boolean vectors. Mathematically speaking, it is to decompose a Boolean matrix $A_{m\times n}$ into the product of $X_{m\times 1} \otimes C_{1\times n}$. One important application of BMD is to mine discrete patterns in binary data, which is important for many data analysis tasks, such as data sampling, compression, and clustering. An example is that replacing individual records with their patterns would greatly reduce data size and simplify subsequent data analysis tasks. As a straightforward approach, rank-one binary matrix approximation has been actively studied recently for mining discrete patterns from binary data. It factorizes a binary matrix into the multiplication of one binary pattern vector and one binary presence vector, while minimizing mismatching entries.

However, this approach suffers from two serious problems. First, if all records are replaced with their respective patterns, the noise could make as much as 50% in the resulting approximate data. This is because the approach simply assumes that a pattern is present in a record as long as their matching entries are more than their mismatching entries. Second, two error types, 1-becoming-0 and 0-becoming-1, are treated evenly, while in many application domains they are discriminated. To address the two issues, we propose weighted rank-one binary ma-

trix approximation. It enables the tradeoff between the accuracy and succinctness in approximate data and allows users to impose their personal preferences on the importance of different error types.

In this section, we will study weighted rank-one BMD. Its decision problem will be proved to be NP-complete. To solve it, several different mathematical programming formulations are provided, from which 2-approximation algorithms are derived for some special cases. An adaptive tabu search heuristic is presented for solving the general problem, and our experimental study shows the effectiveness of the heuristic.

6.1 Motivation of Weighted Rank-One BMD

With the fast development of computer and internet technologies and the decreased cost of data storage devices, a large volume of data are generated and gathered every day. Many datasets have discrete attributes, such as those generated from information retrieval, bio-informatics, and market transactions. Much attention has been focused on efficient techniques for analyzing large and high dimensional datasets. Common tasks for analyzing high dimensional data include extracting correlations between data items, classification, and clustering data items and finding condensed representations. The analysis of large-scale datasets commonly has to deal with the curse of dimensionality. A useful approach is to compress datasets while preserving important underlying patterns. Conventional matrix factorization techniques can effectively and efficiently compress datasets with continuous attributes. For example, singular value decomposition (SVD) can efficiently reduce a given matrix into a low-rank matrix while

minimizing the Frobenius norm of the difference. However, result interpretation is difficult for datasets with discrete attributes.

Existing techniques for mining discrete patterns from binary data include PROXIMUS [34], which can serve the purpose of data compression as well. The idea is to decomposes a given binary matrix into a pattern vector and a presence vector, which are restricted to be binary, such that the multiplication of two decomposed vectors has the least mismatching entries with the original binary matrix. It is also called *rank-one binary matrix approximation*. Given the presence vector, the matrix is partitioned into two parts. The part with the presence of the pattern is grouped together. By recursively applying the same process, all row vectors are clustered and their respective patterns are identified. The task of analyzing the original binary matrix can be switched on those mined patterns, which have much smaller size. This technique has received increased attention recently [35, 59]. For example, Shen et al. [59] even provide an efficient 2-approximation algorithm for rank-one binary matrix approximation recursively conducted in a process of PROXIMUS.

Rank-one binary matrix approximation is to solve $min_{X,Y}||A - XY^T||_F$, where X and Y are binary vectors and called presence vector and pattern vector respectively. Based on entry values of X, rows of A can be divided into two parts. The part associated with entries with the value of 1 in X are considered having pattern Y. By recursively applying rank-one binary matrix approximation, a collection of discrete patterns are mined, which can be utilized to approximate the original data and cluster row vectors.

Example 6.1 *Given a matrix A, a rank-one approximation is computed as follows:*

$$A = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix} \approx \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \end{pmatrix} = XY^T. \qquad (6.1)$$

The discrete pattern mining approach based on rank-one approximation is straightforward and not difficult to implement. However, it suffers from two serious issues.

The first issue is that according to the objective function $min_{X,Y}||A-XY^T||_F$, a row vector $A_{i:}$, denoting the ith row of A, is considered having pattern Y as long as their matching components are greater than mismatching components. In other words, the mismatching ratio could be nearly as much as fifty percent. As a result, the final collection of mined discreet patterns cannot serve as a good approximation to the original data matrix and all subsequent data analysis results would be questionable. To illustrate it, we give Example 6.2. Clearly rank-one binary matrix approximation is not able to divide row vectors of the binary matrix on the left side of Equation (3). As a result, rank-one binary matrix approximation is only able to identify one discrete pattern, while there are two obvious discrete patterns in the original data. Furthermore, the mined pattern $\{1, 1, 1, 1, 0\}$ can hardly represent row vector $\{0, 1, 1, 1, 1\}$, as their matching entries are only one more than mismatching entries.

Example 6.2 *The rank-one binary matrix approximation for a binary matrix with*

two obvious patterns is computed as below:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix} \approx \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \end{pmatrix}. \tag{6.2}$$

The second issue with rank-one binary matrix approximation is that it does not support discrimination on 1-becoming-0 errors and 0-becoming-1 errors, which however is requested by many applications. To illustrate it, we give Example 3. With $(1, 1, 1, 1, 0)$ as the pattern to approximate all row vectors, it introduces one 0-becoming-1 error to the first row vector and one 1-becoming-0 error to the third row vector. For the role mining problem arising from implementing role-based access control [42], the value of 1 in a binary matrix represents a permission assignment. Analyzing an approximate user-to-permission matrix with many 0-becoming-1 errors would generate roles with surplus permissions, which seriously affect system security and safety. In this application, 0-becoming-1 errors should be forbidden.

Example 6.3 *Consider the following rank-one approximation:*

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix} \approx \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \end{pmatrix}. \tag{6.3}$$

6.2 Weighted Rank-One Binary Matrix Approximation

Desired features of a discrete pattern mining technique should include allowing the trade-off between the approximation accuracy and the simplicity of mined

patterns, and enabling users to impose their preferences on the error type distribution. We achieve them by proposing a new measure on the significance of a pattern in a row vector and a new definition of pattern presence. They are defined as below.

Definition 6.1 ***(Pattern Significance)*** *The significance of a pattern $Y \in \{0,1\}^{1\times n}$ in an object $A \in \{0,1\}^{1\times n}$ denoted by $S(Y,A)$, is measured by*

$$S(Y,A) = max\{0, f_{11}(Y,A) - w_1 f_{10}(Y,A) - w_2 f_{01}(Y,A)\}$$

where $f_{ij}(Y,A)$ is the number of attributes which are i in Y and j in A, and w_1 and w_2 are positive weight parameters.

Definition 6.2 ***(Pattern Presence)*** *If $S(Y,A) > 0$, the pattern $Y \in \{0,1\}^{1\times n}$ is considered present in the object $A \in \{0,1\}^{1\times n}$.*

Weight parameters w_1 and w_2 have two purposes. First, the sum value of w_1 and w_2 can be utilized to control the level of error tolerance, in other words approximation accuracy. The greater sum value leads to less error tolerance, hence higher accuracy in approximation. The ratio between w_1 and w_2 reflects the preferences on two error types. The greater weight means being more disliked.

The essential goal of rank-one binary matrix approximation is to discover a pattern with the most significance in the data matrix. We call such a pattern *dominant discrete pattern*. *Weighted rank-one binary matrix approximation* is to take penalty weights into account at the basis of conventional rank-one matrix approximation. When $w_1 = 1$ and $w_2 = 1$, the weighted problem is the same as the

conventional problem. Given a pattern Y, it is easy to determine presence vector X even with the presence of penalty weights. So instead of looking for the pair of X and Y at the same time, we consider weighted rank-one binary matrix approximation as the dominant discrete pattern mining problem defined as following.

Definition 6.3 *(**Dominant Discrete Pattern Mining**) Given m objects consisting of n binary attributes represented by a matrix $A \in \{0,1\}^{m\times n}$, find a dominant pattern $Y \in \{0,1\}^{1\times n}$, such that its total values of pattern significance $\sum_i S(Y, A_{i:})$ are maximized, where $A_{i:}$ denotes the ith row vector of A.*

Two following examples are illustrated to show how weighted rank-one binary matrix approximation can effectively address the two issues with rank-one binary matrix approximation.

Example 6.4 *Reconsider the binary matrix in Example 6.2. By letting $w_1 = w_2 = 2$, its weighted rank-one binary matrix approximation, where the vector on the right side is the dominant discrete pattern, is as below:*

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix} \approx \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \end{pmatrix}. \tag{6.4}$$

In the above example, by doubly penalizing errors, row vectors are successfully divided and $\{1,1,1,1,0\}$ is indeed a true pattern for rows associated with the presence vector. By applying weighted rank-one binary matrix approximation to the other part, pattern $\{0,1,1,1,1\}$ is also successfully revealed.

Example 6.5 *Look at the binary matrix in Example 6.3. By letting* $w_1 = 1$ *and* $w_2 = 3$*, its weighted rank-one binary matrix approximation is as below:*

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix} \approx \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \end{pmatrix}. \tag{6.5}$$

It is free of 1-becoming-0 errors.

Due to the binary data type, each row can be represented as a subset of n attributes. Such a representation provides a convenient way to describe some relationships between two binary row vectors which we will utilize later. They are defined as following.

Definition 6.4 ***(Superset, Strict Superset, Subset, and Strict Subset)*** *For two binary* n*-dimensional row vectors* X *and* Y*, if* $Y_i = 1, \forall X_i = 1$ *,* Y *is a superset of* X *and* X *is a subset of* Y*. If* Y *is superset of* X *and there exists* i *such that* $Y_i = 1$ *and* $X_i = 0$*,* Y *is a strict superset of* X*, while* X *is a strict subset of* Y*.*

6.3 Relation with Other Existing Problems

The dominant discrete pattern mining problem can be related to many research problems that have been studied in the literature. Those problems can be viewed as either its special case or its variant.

Definition 6.5 ***(Maximum Edge Biclique Problem*** *[53]) Given a bipartite graph* $G = (V_1 \cup V_2, E)$ *and a positive integer* K*, does* G *contain a biclique with at least* k *edges?*

A bicilique is a kind of bipartite graph where every vertex of the first set is connected to every vertex of the second set. The maximum edge biclique problem defined as above is a special case of the dominant discrete pattern mining problem when w_1 is 0 and w_2 is greater than the maximal number of 1's entries in a row of A, denoted by $max_i||A_{i:}||_1$. With w_1 being 0, $f_{10}(Y, A_{i:})$ is not counted in the pattern significance formula of $S(Y, A_{i:})$. With w_2 being greater than $max_i||A_{i:}||_1$, a pattern Y can never be present in a row vector $A_{i:}$, which is an exact subset of $A_{i:}$. Therefore the dominant discrete pattern mining problem with $w_1 = 0$ and $w_2 \geq max_i||A_{i:}||_1$ is to find a pattern Y which covers the most 1's entries in row vectors of A, which are supersets of Y. Any binary matrix $A_{m\times n}$ can be expressed as an equivalent bipartite graph $G = (V_1 \cup V_2, E)$, where V_1 has m vertices corresponding to all rows , V_2 has n vertices corresponding to all attributes, and there is an edge connecting vertices $V_1(i)$ and $V_2(j)$ if $A_{i,j} = 1$. As a 1's entry in A corresponds to an edge in its equivalent bipartite graph, a dominant discrete pattern with $w_1 = 0$ and $w_2 \geq max_i||A_{i:}||_1$ induces a maximum edge biclique.

For illustration, consider the binary matrix in Equation (6.1) as an example. It can be expressed as the bipartite graph in Figure 6.1a. Its dominant discrete pattern with $w_1 = 0$ and w_2 greater than n is given as in Equation (6.6) and its induced biclique is as shown in Figure6.1b.

$$\begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix} \approx \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \end{pmatrix}. \tag{6.6}$$

Definition 6.6 *(**Maximum Tile** [19]) Given a database D as a binary matrix $A_{m\times n}$, find the tile with the largest area in D, where a tile corresponds to a row*

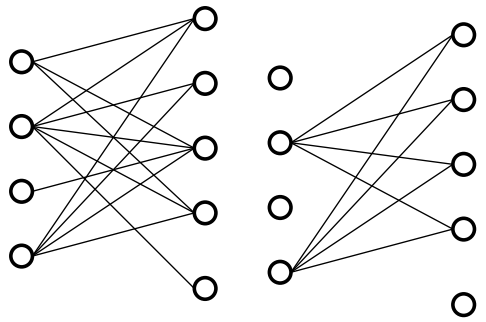


Figure 6.1. A Bipartite Graph and Its maximum Edge Biclique

index subset R and a column index set C, such that $A(i,j) = 1, \forall i \in R, j \in C$.

The maximum tile problem defined as above is essentially to find the largest tile full of 1's entries in a binary matrix, allowing the manipulation on the order of rows and columns. It is equivalent to the maximum edge clique problem, and hence equivalent to the dominant discrete pattern problem with $w_1 = 0$ and $w_2 \geq max_i||A_{i:}||_1$.

Consider the binary matrix on the left side of Equation (6.6) as a tiling database. The largest tile induced by the dominant discrete pattern is as shown in Equation (6.7).

$$\begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ \boxed{1 \;\; 1 \;\; 1 \;\; 1} & & & & 1 \\ 0 & 0 & 1 & 0 & 0 \\ \boxed{1 \;\; 1 \;\; 1 \;\; 1} & & & & 0 \end{pmatrix} \tag{6.7}$$

Definition 6.7 *(**Maximum Edge Weight Biclique Problem** [43]) Given a bipartite graph $\{V_1 \cup V_2, E\}$ and weights $\{w_{ij}\}$ associated with edges $\{(V_1(i), V_2(j))\}$ respectively, find a biclique C, where the sum of the edge weights is maximum.*

The dominant discrete pattern problem with w_1 and w_2 being equal can be mapped to a special case of the maximum edge weight biclique problem. The maximum edge weight biclique problem is to find a biclique with the maximum weight from a complete bipartite graph. As we have illustrated, any binary matrix $A_{m\times n}$ can be expressed as a bipartite graph $G(V_1 \cup V_2, E)$. We can further expand it as a weighted complete bipartite graph. First complete all missing edges to make it as a complete graph. Then assign a weight of 1 to all edges in the original bipartite graph and a negative weight of $-w_1$ to the newly added edges. A dominant discrete pattern in the original binary matrix would correspond to a maximum weight biclique in the constructed weighted complete bipartite graph.

For illustration, reconsider the binary matrix in Equation (6.1). Its corresponding maximum edge weight biclique problem instance is as shown in Figure 6.2, where dashed lines are original edges with the weight of 1 and thick lines are added edges with the weight of $-w_1$.

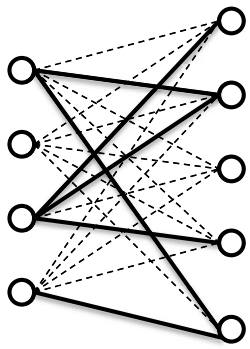


Figure 6.2. Edge-Weighted Complete Bipartite Graph

6.4 Mathematical Programming Formulation

The main result of this section is to present mathematical programming formulations for the dominant discrete pattern mining problem. In particular, we give an unconstrained quadratic binary programming formulation, an integer linear programming formulation, and a relaxed linear programming formulation, which for the case of $w_1 = w_2 = 1$ induces a 2-approximate algorithm.

6.4.1 Unconstrained Quadratic Binary Programming

The dominant discrete pattern mining problem is to find the pattern Y maximizing the global pattern significance $\sum_i S(Y, A_{i:})$ for a given binary matrix A. Its corresponding presence vector X can be obtained easily after the dominant discrete pattern Y is found. But in the quadratic binary programming formulation given as below, we treat presence vector X as variables along with pattern vector Y.

$$
\begin{aligned}
& \textstyle\sum_i S(Y, A_{i:}) \\
= \; & \textstyle\sum_i max\{0, f_{11}(Y, A_{i:}) - w_1 f_{10}(Y, A_{i:}) - w_2 f_{01}(Y, A_{i:})\} \\
= \; & \textstyle\sum_i X_i (f_{11}(Y, A_{i:}) - w_1 f_{10}(Y, A_{i:}) - w_2 f_{01}(Y, A_{i:})) \\
= \; & \textstyle\sum_i X_i (\sum_j A_{ij} Y_j - w_1 \sum_j (1 - A_{ij}) Y_j \\
& \textstyle -w_2 \sum_j A_{ij} (1 - Y_j)) \\
= \; & \textstyle\sum_{ij} (A_{ij} + w_1 A_{ij} + w_2 A_{ij} - w_1) X_i Y_j - \sum_{ij} w_2 A_{ij} X_i \\
= \; & X^T U Y - X^T V \vec{1}
\end{aligned}
\tag{6.8}
$$

where U is defined as $U_{ij} = A_{ij} + w_1 A_{ij} + w_2 A_{ij} - w_1$, V_{ij} is defined as $w_2 A_{ij}$, and $\vec{1}$ denotes the all-ones vector.

The explanation of the above equation deduction process is as follows: (i) The

first induction step is obvious; (ii) The second induction step is by the fact that if $f_{11}(Y, A_{i:}) - w_1 f_{10}(Y, A_{i:}) - w_2 f_{01}(Y, A_{i:}) \leq 0$, $X_i = 0$; (iii) In the third induction step, $f_{11}(Y, A_{i:})$ counting the number of attributes where Y and $A_{i:}$ both are 1 is measured by $\sum_j A_{ij} Y_j$, and $f_{10}(Y, A_{i:})$ and $f_{01}(Y, A_{i:})$ are by $(1 - A_{ij})Y_j$ and $A_{ij}(1 - Y_j)$ respectively; (iv) The last induction step arranges the whole formula as matrix multiplications.

Therefore the dominant discrete pattern mining problem can be simplified as an unconstrained binary quadratic programming problem as $max\{X^T UY - X^T V\vec{1} | X \in \{0,1\}^m, Y \in \{0,1\}^n\}$. A binary matrix is closely related to a bipartite graph. It is also a useful approach for problems involved with a bipartite graph to be formulated as an unconstrained quadratic binary programming problem [4].

6.4.2 Integer Linear Programming

The unconstrained quadratic binary programming formulation can be linearized as an integer linear programming problem by introducing auxiliary binary variables $\{Z_{ij}\}$ as below.

$$\begin{aligned} & max \sum_{ij} U_{ij} Z_{ij} - \sum_{ij} V_{ij} X_i \\ & s.t. \begin{cases} -X_i - Y_j + 2Z_{ij} \leq 0 \quad \forall i, j \\ X_i + Y_j - Z_{ij} \leq 1 \quad \forall i, j \\ X_i, Y_j, Z_{ij} \in \{0,1\} \end{cases} . \end{aligned} \tag{6.9}$$

The linearization is done by replacing $X_i Y_j$ with Z_{ij}. As the value of $X_i Y_j$ can only be either 0 or 1, so Z_{ij} is binary. Constraints $-X_i - Y_j + 2Z_{ij} \leq 0$ and $X_i + Y_j - Z_{ij} \leq 1$ guarantee: (i) $Z_{ij} = 1$ when both X_i and Y_j are 1; (ii) $Z_{ij} = 0$ when one of X_i and Y_j is 0.

The linearization skill of replacing $X_i Y_j$ with a binary variable Z_{ij} is also em-

ployed by Shen et al. in [59] for formulating the rank-one approximation problem. However their enforcing constraints are as below.

$$\begin{cases} -X_i - Y_j + 2Z_{ij} \leq 0 \quad for\ A_{ij} = 1 \\ X_i + Y_j - Z_{ij} \leq 1 \quad for\ A_{ij} = 0 \\ X_i, Y_j, Z_{ij} \in \{0, 1\} \end{cases} . \tag{6.10}$$

Notice that constraints $-X_i - Y_j + 2Z_{ij} \leq 0$ and $X_i + Y_j - Z_{ij} \leq 1$ are not being enforced for every (ij). Therefore, technically speaking their integer linear programming formulation is not strict, but a relaxed version.

6.4.3 Linear Programming Relaxation

In general the integer linear programming problem is NP-hard, while the linear programming problem is not. It has polynomial algorithms such as interior point method [32] and also has simplex method algorithms which have very good practical performance despite exponential worst-case running time [12]. Thus a typical approach to solve an integer linear programming problem is to derive an approximate solution, also an upper bound to its optimum (if it is a maximization problem) by solving its linear programming relation.

The linear programming relaxation for the dominant discrete pattern mining problem can be easily obtained by simply replacing the constraint set $\{X_i, Y_i, Z_{ij} \in \{0, 1\}\}$ in Equation (6.9) by $\{X_i, Y_i, Z_{ij} \in [0, 1]\}$. The relaxation essentially expands the feasible solution region to a polytope and makes the problem easier to solve. But the optimal solution of the relaxation problem is not necessarily a feasible solution of the original problem, because its components could be fractional. So people usually round either down or up those fractional components to make a feasible solution and use it as an approximate solution. However, the

approximation ratio is not guaranteed.

As we mentioned earlier, the integer programming formulation for the rank-one approximation problem provided by Shen et al. in [59] is not a strict formulation, but a relaxed version. However, such a relaxed integer programming formulation surprisingly leads to a 2-approximation algorithm via simplex methods. We may directly apply their approach to our problem. Accordingly a new linear programming relaxation for Equation (6.11) is given as below.

$$\begin{aligned} & max \sum_{ij} U_{ij} Z_{ij} - \sum_{ij} V_{ij} X_i \\ s.t. & \begin{cases} -X_i - Y_j + 2Z_{ij} \leq 0 \quad for\ A_{ij} = 1 \\ X_i + Y_j - Z_{ij} \leq 1 \quad for\ A_{ij} = 0 \\ X_i, Y_j, Z_{ij} \in [0,1] \end{cases} . \end{aligned} \tag{6.11}$$

Theorem 6.1 *The optimal solution of Equation (6.11) via simplex algorithms is integral, in other words a feasible solution of Equation (6.9).*

Proof. As Equation (6.11) has the same constraints as the linear programming relaxation problem studied in [59], the same result holds that the coefficient matrix of the inequality constraint set is a totally unimodular matrix. A totally unimodular matrix is a matrix for which the determinant of every square non-singular submatrix is 1 or -1. All parameters on the right side of inequalities are also integral. Two properties lead to that the optimal solution of Equation (6.11) via a simplex algorithm is integral. The reason is as following. A simplex method is searching from one basic feasible solution to another basic feasible solution till the optimum is reached. Suppose the constraint set has a standard form as $\{AX = b, X \geq 0\}$, to which any linear constraint set can be transformed. A

basic feasible solution is corresponding to a partition $\{X_B, X_N\}$ of variables X and accordingly the constraint set can be reformed as $BX_B + NX_N = b$. By letting $X_N = 0$, $X_B = B^{-1}b$. If $B^{-1}b > 0$, $\{B^{-1}b, 0\}$ is a basic feasible solution. According to the Cramer's rule [22], the ith competent of X_B is $\frac{det(B')}{det(B)}$, where $det(B)$ denotes the determinant of B, B' is B except its ith column is replaced by b. Because in Equation (6.11) the coefficient matrix of the constraint set is totally unimodular, the determinant of every square non-singular submatrix is 1 or -1. Hence the denominator of $\frac{det(B')}{det(B)}$ is 1 or -1. Since all parameters in Equation (6.11) is integral, the nominator of $\frac{det(B')}{det(B)}$ is also integral. Thus every component of any basic feasible solution of Equation (6.11) via a simplex algorithm is integral, and the optimal solution of the relaxation problem is a feasible solution of its original problem.

Our proof is built on the basis of the proof given by Shen et al. [59]. But in addition to pointing out the existence of two sufficient conditions that a totally unimodular coefficient matrix and all integral parameters, we provide a deep insight on why those two conditions make all basic feasible solutions via a simplex algorithm integral.

Unfortunately, one cannot prove that the optimal solution of Equation (6.11) is 2-approximate to the optimal solution of Equation (6.9). We conjecture that they are at least very close. More importantly the optimal solution of Equation (6.11) is a feasible solution of Equation (6.9). The problem of finding a feasible solution of a system of inequalities in integers in general is NP-complete. Many good heuristics for complex combinatorial optimization problems are starting from a feasible solution and then iteratively improving the current solution by certain

rules. A good starting feasible solution such as the optimal solution of Equation (6.11) could save much computing time.

6.5 Computational Complexity and Approximation Algorithm

The main result of this section is to prove that the decision problem of dominant discrete pattern mining is NP-complete and present 2-approximation algorithms to some special cases of the dominant discrete pattern mining problem.

6.5.1 Computational Complexity

We prove NP-completeness of the decision problem of dominant discrete pattern mining by a reduction from the decision maximum edge biclique problem.

Lemma 6.1 *The decision maximum edge biclique problem is NP-complete [53].*

Theorem 6.2 *The decision problem of dominant discrete pattern mining is NP-complete.*

Proof. The decision problem of dominant discrete pattern mining is given a binary matrix $A_{m\times n}$ and a value δ to determine if there is a pattern Y such that $\sum_j S(A_{i:}, Y) \geq \delta$. Given a dominant discrete pattern Y, it is easy to determine whether the instance is true. Thus the decision problem of dominant discrete pattern mining belongs to NP. In the previous section, we showed that the dominant discrete pattern mining problem is a generalization of the maximum edge biclique problem, the decision version of which is NP-complete. Therefore, the decision problem of dominant discrete pattern mining is NP-complete.

6.5.2 Approximation Algorithm

We will study three special cases of the dominant discrete pattern mining problem and present approximation algorithms for them. Before doing that, we first introduce a result.

Lemma 6.2 *Any integer linear programming problem with n variables subject to m linear constraints with at most two variables per inequality, and with all variables bounded between 0 and U, has a 2-approximation algorithm which runs in polynomial time $O(mnU^2log(Un^2m))$ [26].*

Hochaum et al. in [26] present a 2-approximation algorithm for integer linear programs with at most two variables per inequality. We briefly introduce this algorithm here. It transforms the integer program into a monotone integer system first [1], computes an optimal solution for it, and then modifies the result via some simple rule to obtain a feasible solution to the original problem, which is also a 2-approximate solution.

Case 1: $w_1 = 1$ and $w_2 \geq T$ where T is the maximal number of entries with the value of 1 in a row of A.

As we have shown in the preceding section, the dominant discrete pattern mining problem with $w_1 = 1$ and $w_2 \geq T$ is equivalent to the maximum edge biclique problem. Hochbaun in [25] gives a linear programming formulation for the edge weighted biclique problem which is to delete from a bipartite graph $\{V_1, V_2, E\}$,

[1] For the definition of a monotone integer system, refer to the paper [24].

a minimum weight collection of edges so that the remaining edges induce a complete bipartite graph. By slight modifications, we obtain a linear programming formulation for our special case as below.

$$max \sum_{A_{ij}=1} Z_{ij}$$
$$s.t. \begin{cases} 2Z_{ij} - (X_i + Y_j) \leq 0, \ for \ A_{ij} = 1 \\ X_i + Y_j \leq 1, \ for \ A_{ij} = 0 \\ X_i, Y_j, Z_{ij} \in \{0, 1\} \end{cases} . \tag{6.12}$$

Because $w_2 \geq T$, XY^T cannot cover 1's entries in A. In other words $X_i Y_j = 0$ if $A_{ij} = 0$, which is guaranteed by the constraint set $\{X_i + Y_j \leq 1, \ for \ A_{ij} = 0\}$. The objective function $\sum_{A_{ij}=1} Z_{ij}$ counts (i, j) such that both A_{ij} and Z_{ij} are 1. Z_{ij} can be 1 if and only if both X_i and Y_j are 1, which is guaranteed by the first constraint set $\{2Z_{ij} - (X_i + Y_j) \leq 0, \ for \ A_{ij} = 1\}$.

The constraint of $2Z_{ij} - (X_i + Y_j) \leq 0$ can be spit into two equivalent constraints $Z_{ij} - X_i \leq 0$ and $Z_{ij} - Y_j \leq 0$. The similar skill is also used in [25]. Then Equation (6.12) can be put in the form with at most two variables per inequality as below.

$$max \sum_{A_{ij}=1} Z_{ij}$$
$$s.t. \begin{cases} Z_{ij} - X_i \leq 0, \ for \ A_{ij} = 1 \\ Z_{ij} - Y_j \leq 0, \ for \ A_{ij} = 1 \\ X_i + Y_j \leq 1, \ for \ A_{ij} = 0 \\ X_i, Y_j, Z_{ij} \in \{0, 1\} \end{cases} . \tag{6.13}$$

It naturally leads to the following theorem.

Theorem 6.3 *The dominant discrete pattern mining problem with* $w_1 = 1$ *and*

$w_2 \geq T$ *is 2-approximable.*

Case 2: $w_1 \geq T$ *where* T *is the maximal number of entries with the value of 1 in a row of* A*.*

$w_1 \geq T$ enforces that a pattern Y can only be present in its subset. In the preceding section, we have shown that the dominant discrete pattern mining problem with $w_1 \geq T$ is equivalent to finding a maximum weight biclique from a complete weighted bipartite graph. In which, the weight for the edges corresponding to entries with the value of 1 in A is 1 and the weight for the other edges is $-w_1$. It is a special case of the edge weighted biclique problem studied in [25]. Again by modifying the linear programming formulation for the edge weighted biclique problem provided in [25], we obtain a linear programming formulation for our problem as below.

$$\begin{aligned} & max \sum_{A_{ij}=1} Z_{ij} - \sum_{A_{ij}=0} w_2 Z_{ij} \\ & s.t. \begin{cases} Z_{ij} - X_i \leq 0 \\ Z_{ij} - Y_j \leq 0 \\ X_i, Y_j, Z_{ij} \in \{0,1\} \end{cases} . \end{aligned} \tag{6.14}$$

Equation (6.14) is similar to Equation (6.13), except that: (i) the constraint set of $X_i + Y_j \leq 1,\ for\ A_{ij} = 0$ is deleted because it is not necessary, and the weights of $\{Z_{ij}\}$ are replaced with 1 and $-w_2$ accordingly. Every inequality constraint of Equation (6.14) has only two variables. Obviously the approximation algorithm presented in [26] applies to it. We state it as below.

Theorem 6.4 *The dominant discrete pattern mining problem with* $w_1 \geq T$ *is 2-approximable.*

Case 3: $w_1 = 1$ *and* $w_2 = 1$.

The dominant discrete pattern mining problem with $w_1 = 1$ and $w_2 = 1$ is equivalent to the rank-one approximation problem. A 2-approximation algorithm via linear programming relaxation is proposed by Shen et al. in [59]. For simplicity, we skip its details.

6.6 Adaptive Tabu Search Heuristic

A tabu search heuristic is presented for the dominant discrete pattern mining problem. For large-scale problems running time for mathematical programming is always an issue. A 2-approximation algorithm sometimes cannot produce satisfactory results in practice. The dominant discrete pattern mining problem essentially is a combinatorial optimization problem. For such type of problem heuristics usually bring good practical performance. An iterative heuristic is employed for the rank-one approximation problem by both [59] and [34]. Its basic idea is that starting from a feasible solution of pattern Y and presence vector X, alternatively fixing one of them and then searching for the best solution of the counterpart till both are stable. The only difference between [59] and [34] is that in [59] instead of a random starting solution the starting feasible solution is the optimal solution of Equation (6.11), which is 2-approximate to the optimal solution of the original problem. Their iterative heuristic can be viewed as a greedy heuristic, which greedily picks a neighboring solution at each iteration. One drawback with greedy heuristics is that it is easy to reach local optimums, and once reaching it the solution cannot be improved any more.

To address the local optimum issue, we present an adaptive tabu search heuris-

tic. Tabu search is a mathematical optimization method. It essentially belongs to the class of local search techniques as well. But it enhances the performance of a local search method by using memory structures. When a potential solution has been determined, it is marked as "taboo" so that the algorithm does not visit that possibility repeatedly. In Section 6.4, we have shown that the dominant discrete pattern mining problem can be formulated as an unconstrained quadratic binary program. In literature, tabu search has been successfully employed to solve quadratic binary programming problems [21]. This is another reason why we choose tabu search.

The dominant discrete pattern mining problem is given a binary matrix A to maximize $\sum_i S(A_{i:}, Y)$. It is easy to compute $S(A_{i:}, Y)$ once Y is determined.

The adaptive tabu search approach taken here is a modification of the adaptive memory tabu search presented in [21]. The basic procedure of our algorithm is as following. It starts with an initial solution of Y and then goes through a series of alternative constructive phases and destructive phases. In the constructive phases, progressively set 0 components of Y to 1, while in the destructive phase, progressively set 1 components of Y to 0. At each iteration, one component of Y is chosen and its value is flipped.

A greedy rule would be choosing the component by changing which improves the objective function the most. However, to avoid visiting some solutions repeatedly, frequency and recency information is utilized. Frequency information is about *critical solutions* encountered to date. A critical solution is the solution at which the next move (either add 1 or drop 1) causes the objective function to decrease. Such an event is called a *critical event*. Recency information is about k

$span$	number of further steps after a critical event
$span^*$	maximum value of $span$
Dir	direction on changing the value of $span$
p_r	penalty weight from the recency tabu list
p_f	penalty weight from the frequency tabu list
$IterCount$	count of total iterations
$IterCount^*$	maximum value of $IterCount$
$IterSpan$	count of iterations at the current value of $span$
t	scale parameter on the maximum value of $IterSpan$

Table 6.1. Parameter Descriptions

Algorithm 6.10 Tabu Search Algorithm

Input: A, $IterCount^*$, p_r, p_f, $span^*$, t;
Output: X, Y;
1: Y=0, Y_{best}=0, $IterCount$=0, $span$=1, Dir=*increase*, $Count$=0, $IterSpan$=0;
2: **while** $IterCount < IterCount^*$ **do**
3: Conduct the constructive phase;
4: Conduct the transitive phase;
5: Conduct the destructive phase;
6: Conduct the transitive phase;
7: **end while**

critical solutions recently visited. Two pieces of information are stored in two tabu lists respectively. A greedy approach would stop at a critical event. But the tabu search heuristic taken here would keep moving. The number of further moving steps is based on *strategic oscillation*. Such a scheme would enable the solution to step out some local optimums. The amplitude of oscillation, in other words the number of further moving steps, is determined by a parameter, *span*. The span parameter is fixed for a certain number of iterations and then changed in a systematic fashion. To manage the value of *span*, an additional transitive phase is added between each constructive phase and each destructive phase. So technically speaking the adaptive tabu search approach consists of three phases.

The general structure of the tabu search approach is as outlined in Algorithm 6.10. Parameter descriptions are provided in Table 6.1. Variable $span$ determines the further steps that can be made after encountering a critical solution. $span^*$ is an input parameter, limiting the maximum value of $span$. The value of Dir indicates the direction of changing the value of $span$, either increasing or decreasing. p_r and p_f are penalty weights on information derived from recency and frequency tabu lists respectively. Its usage will be elaborated later. Variable $IterCount$ counts the total iterations to date. $IterCount^*$ is an input parameter limiting the maximum value of $IterCount$ and hence the algorithm runtime. $IterSpan$ counts iterations that have been made at the current value of $span$. t is an input scale parameter on the maximum value of $IterSpan$. We explain it explicitly later. The termination condition of Algorithm 6.10 is the total number of iterations less than $IterCount^*$. It could be replaced with a termination condition on the runtime. In each loop, three phases are traversed in order.

6.6.1 Constructive Phase

In the constructive phase, we progressively pick a 0 component of Y and flip it to 1. The intuitive component-picking policy is to choose the component by flipping which contributes the largest net increase to the objective function value $\sum_i S(A_{i:}, Y)$. But we want to avoid repeatedly visiting some solutions. To this end, two tabu lists are utilized to influence the search process. The recency tabu list is a vector denoted by V_r, which is the sum of the most recent k critical solutions. At each critical event, it is updated as below: $V_r = V_r + Y(current) - Y(current - k)$, where $Y(current)$ denotes the current critical solution and

$Y(current-k)$ denotes the critical solution encountered before k critical events. The frequency tabu list is the sum of all critical solutions encountered so far. At each critical event, it is updated as below: $V_f = V_f + Y(current)$. To avoid repeatedly visiting some critical solutions, we give certain penalty on a 0 component which was 1 in recent critical solutions or was 1 frequently in the past critical solutions. So the final evaluation on a 1 component at the jth position is determined by the following measure:

$$f_1(j) = \begin{array}{l} \sum_i S(A_{i:}, Y + e_j) - \sum_i S(A_{i:}, Y) \\ -p_r * V_r(j) - p_f * V_f(j) \end{array} . \tag{6.15}$$

In above, e_j is a unit vector with the jth entry of 1, p_r and p_f are penalty weights, and $V_r(j)$ and $V_f(j)$ are the jth entry of V_r and V_f respectively.

The details of the constructive phase are as described in Algorithm 6.11. It consists of two parts. The first part is to iteratively flip the component which maximizes the evaluation formula of Equation (6.15). The second part is to make $span$ more iterations after a critical event occurs.

6.6.2 Destructive Phase

In the destructive phase, we progressively pick a 1 component in Y and change it to 0. The evaluation measure for component picking is as below. It is similar to Equation (6.15) except that we add penalty terms.

$$f_2(j) = \begin{array}{l} \sum_i S(A_{i:}, Y - e_j) - \sum_i S(A_{i:}, Y) \\ +p_r * V_r(j) + p_f * V_f(j) \end{array} . \tag{6.16}$$

The details of the destructive phase is as shown in Algorithm 6.12. It is same as Algorithm 6.11 except that the component picking rule is changed.

Algorithm 6.11 Constructive Phase

while $|Y| < n$ **do**
  Find j' maximizing $f_1(j)$ subject to $Y(j) = 0$;
  if $f_1(j') > 0$ **then**
    $Y = Y + e_j$;
    $IterCount = IterCount + 1$;
    Update Y_{best} if necessary;
  end if
end while
while $CountSpan \leq span$ and $|Y| > 0$ **do**
  $j' = argmax_{Y(j)=1}(f_2(j))$;
  if $f_1(j') \leq 0$ **then**
    Update V_r and V_f;
  end if
  $Y = Y + e_j$;
  $CountSpan = CountSpan + 1$;
  $IterCount = IterCount + 1$;
  Update Y_{best} if necessary;
end while
$CountSpan = 0$;

6.6.3 Transitive Phase

If $span$ is an unchanged constant, the constitutive phase and the destructive phase constitute a complete tabu search algorithm. A large $span$ would enlarge the search space at the expense of increasing runtime. But with a small $span$ the algorithm might not be able to find a satisfactory solution. To address the issue, adaptive tabu search adjusts the value of $span$ in a systematic fashion. We fix the value of $span$ for a certain number of iterations and then increase it by 1. Keep doing this until $span$ reaches $span^*$, the maximum value predefined. After that, we gradually decrease the value of $span$ by 1. Therefore, the value of $span$ will transverse back and forth between 1 and $span^*$. The complete procedure of transitive phase is as described in Algorithm 6.13.

Algorithm 6.12 Destructive Phase

while $|Y| > 0$ **do**
  Find j' maximizing $f_2(j)$ subject to $Y(j) = 1$.
  if $f_2(j') > 0$ **then**
    $Y = Y - e_j$;
    $IterCount = IterCount + 1$;
    Update Y_{best} if necessary;
  end if
end while
while $CountSpan \leq span$ and $|Y| > 0$ **do**
  $j' = argmax_{Y(j)=1}(f_2(j))$;
  if $f_2(j') \leq 0$ **then**
    Update V_r and V_f;
  end if
  $Y = Y - e_j$;
  $CountSpan = CountSpan + 1$;
  $IterCount = IterCount + 1$;
  Update Y_{best} if necessary;
end while
$CountSpan = 0$;

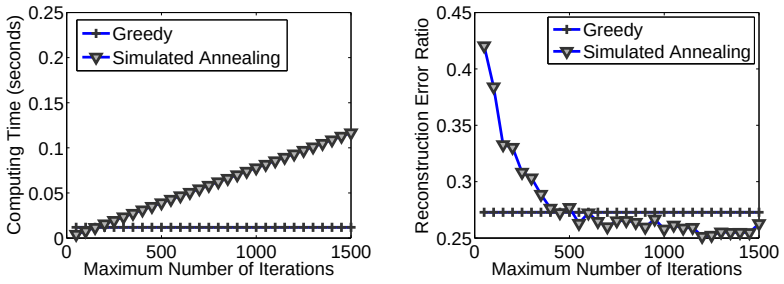
6.7 Experiments

In this section, we illustrate the properties of weighted rank-one binary matrix approximation by implementing it on synthetic data. The synthetic data are created as following: (i) Generate eight random binary row vectors, such that each vector is of size 1×50 and has exactly $50 * \rho$ components of 1, where ρ is the parameter determining data sparseness and within $[0, 1]$; (ii) Generate 200, 200, 150, 150, 100, 100, 50, and 50 copies respectively for each of the eight vectors, and put them together to constitute a binary matrix with size 1000×50. Therefore the resultant binary matrix contains eight discrete patterns;(iii) Flip the value for each cell in the binary matrix with probability $\rho * \xi$, where ξ is the parameter determining noise ratio and is within $[0, 1]$.

Algorithm 6.13 Transitive Phase

```
1: if Dir = increase then
2:    if span > span* then
3:       span = span*;
4:       Dir = decrease;
5:       IterSpan = 0;
6:    else
7:       if IterSpan > t * span then
8:          span = span + 1;
9:          IterSpan = 0;
10:      end if
11:   end if
12: else
13:   if span = 0 then
14:      span = 1;
15:      Dir = increase;
16:      IterSpan = 0;
17:   else
18:      if IterSpan > t * span then
19:         span = span − 1;
20:         IterSpan = 0;
21:      end if
22:   end if
23: end if
```

We first study the performance of the adaptive tabu search heuristic with respect to its speed of convergence. Parameter settings for the adaptive tabu search heuristic are as: $p_r = 1$, $p_f = \frac{IterCount}{50}$, $span^* = 3$, $IterCount^* = 500$, and $t = 3$. We run it on two synthetic binary matrices with generating parameter values as $\{\rho = 0.3, \xi = 0.2\}$ and $\{\rho = 0.3, \xi = 0\}$ respectively. The experimental results are as shown in Figures 6.3a and 6.3b. In Figure 6.3a, it shows that the adaptive tabu search heuristic only takes about 15 iterations to reach a discrete pattern with significance value of 200. The pattern is the global optimum solution, because according to the data generating process with the noise parameter $\xi = 0$ the dominant discrete pattern covers 200 row vectors. In Figure 6.3b, it shows



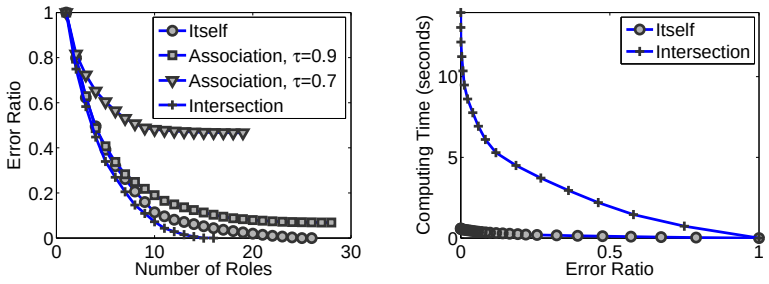
(a) $\rho = 0.3, \xi = 0$ (b) $\rho = 0.3, \xi = 0.2$

Figure 6.3. Speed of Convergence

that the adaptive tabu search heuristic only takes about 15 iterations to reach a pattern with significance value of about 220. As no efficient way to check if the found pattern is dominant, but according to the data generating process, it should be nearly dominant at least. Both graphs verify the high convergence rate of our adaptive tabu search heuristic.

We then evaluate the performance of the adaptive tabu search heuristic by comparing it to the alternating iterative approach proposed in [34] with respect to approximation ratio. The approximation ratio measure is defined as $\frac{f_{11}-f_{10}-f_{01}}{||A||_1}$ where f_{11} is the number of matching entries with the value of 1, f_{10} is the number of 1-becoming-0 errors, f_{01} is the number of 0-becoming-1 errors, and $||A||_1$ is the number of 1 entries in the original data. For a fair comparison, we let penalty weighes w_1 and w_2 for weighted rank-one binary matrix approximation be 1. Specific parameter setting for the adaptive tabu search heuristic is: $p_r = 1$, $p_f = \frac{IterCount}{50}$, $span^* = 3$, $IterCount^* = 500$, and $t = 3$. Regarding p_f, $\frac{IterCount}{50}$ means that the frequency tabu list has more and more impact when the algorithm proceeds.

We compare our algorithm with the alternating iterative approach on various



(a) Varying Density (b) Varying Noise

Figure 6.4. Comparison w.r.t Approximation Ratio on Synthetic Data

datasets. We first generate data by fixing the noise ratio ξ to be 0.2 and varying the density parameter value ρ from 0.1 to 0.5. As both algorithms require an initial solution. To be favorable to the alternating iterative approach, we employ the *Partition* and *Neighbor* procedures designed for the alternating iterative approach proposed in [34] to generate initial solutions. The experimental results are as shown in Figures 6.4a. The observation is that the tabu search heuristic performances significantly better than the alternating iterative approach in any case. According to the data-generating process, the dominant discrete pattern should cover about 20 percent of the whole binary matrix. So the maximum approximation ratio is around 0.2. Experimental results show that the adaptive tabu search heuristic produces nearly optimal solutions, while the performance of the alternating iterative approach is unsatisfactory. We then generate data by fixing the density parameter value ρ to be 0.3 and varying noise from 0 to 0.4. The results is as shown in Figure 6.4b. They again confirm the conclusion that the adaptive tabu search heuristic has significantly better performance.

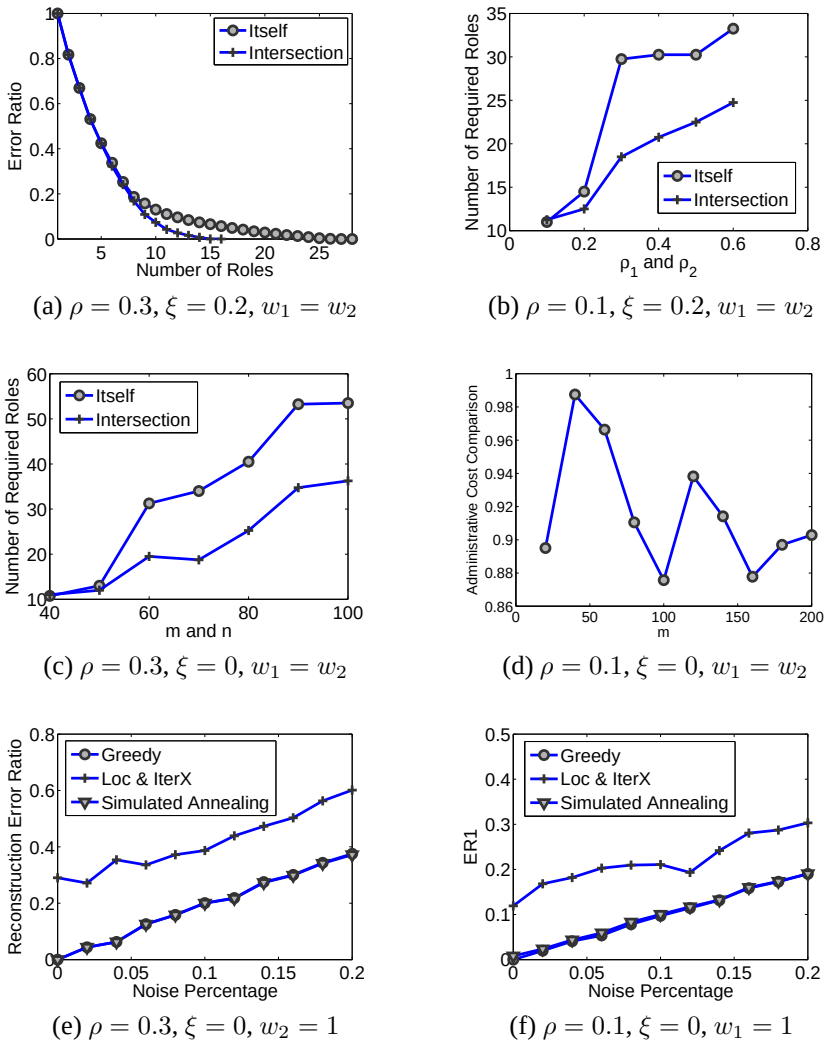
Next we investigate the impact of penalty weights on the quality and properties of mined patterns. Two measures could be employed to evaluate the quality

of mined patterns. The first measure is the number of patterns. The objective of rank-one matrix approximation is to reduce the size of original data by replacing individual records with their corresponding patterns. So less patterns means more succinct approximate data. The second measure is approximation ratio. It is always desirable to retain the original data information in the approximate data, because severely contaminated data would hardly produce any convincing data analysis result.

Again we run our adaptive tabu search heuristic on synthetic data with the same data-generating process as before. First, we let $w_1 = w_2$ and increase their values gradually. We run our pattern mining algorithm on various data with different data-generating parameter settings. The experimental results are as shown in Figures 6.5a-6.5d. In these four graphs, the same observation is that by increasing penalty weights approximation ratio increases consistently. However, with conventional rank-one binary matrix factorization, approximation ratio is not adjustable. When penalty weights are increased to 4, the approximation ratio can be improved to be nearly 1, while the required number of patterns is still significantly less than the number of original records.

We then fix one of penalty weights and vary the other to investigate the impact of penalty weights on error type distribution. First, we fix w_2 to be 1 and then vary w_1 from 1 to 4. The experimental result is as shown in Figure 6.5e. It is clearly observed that type I error rate decreases consistently with increasing w_1. When $w_1 = 4$, the type I error rate is as low as 0.1. We then fix w_1 and vary w_2. The experimental result is as shown in Figure 6.5f. The same fact is observed. It supports the second main advantage of weighted rank-one binary

matrix factorization. By adjusting the value of w_1 and w_2, users can easily reflect their preferences on error type distribution.



(a) $\rho = 0.3, \xi = 0.2, w_1 = w_2$ (b) $\rho = 0.1, \xi = 0.2, w_1 = w_2$

(c) $\rho = 0.3, \xi = 0, w_1 = w_2$ (d) $\rho = 0.1, \xi = 0, w_1 = w_2$

(e) $\rho = 0.3, \xi = 0, w_2 = 1$ (f) $\rho = 0.1, \xi = 0, w_1 = 1$

Figure 6.5. Impact of Penalty Weights on Approximation Ratio, Number of Patterns, and Error Type Distribution

CHAPTER 7
CONCLUSION AND FUTURE WORK

This dissertation studies BMD, extended BMD and weighted rank-one BMD. BMD has many applications including text mining, role mining, clustering, and information retrieval. Important BMD variants are extensively investigated and a general integer programming framework for studying BMD variants is provided. Extended BMD improves the conventional BMD model by including the set difference operation, which not only makes the decomposition solution more interpretable, but also decomposes an input Boolean matrix in a more succinct way. Weighted rank-one BMD provides a flexible approach to mine discrete patterns from binary records. It allows users to effectively impose their preferences on the approximation level of mined discrete patterns and the error type distribution in the approximation. For each presented Boolean matrix decomposition variant, computational complexity is performed, integer programming formulation is given, and alternative approaches such as approximation algorithms and heuristics are provided. Extensive experiments on synthetic and real data sets are conducted to evaluate the performance of our proposed algorithms.

There are still a lot of future work remaining. We will name a few here. First, there are some other interesting applications of BMD that have not been investigated. For example, BMD can be applied to social network analysis, because

social network data can be represented as a Boolean matrix and a BMD solution identifies cliques among the data. Another application is overlapping clustering which is important in many domains such as DNA microarray analysis . A BMD solution divides observed Boolean records into clusters, while a record can belong to multiple clusters. Second, in reality data often change over time. So the real observed data could be a three-dimensional Boolean matrix. There is no any existing approach that can effectively discover patterns from three-dimensional Boolean matrix. In the future, we plan to look at those interesting remaining problems.

REFERENCES

[1] A. KERN, M. KUHLMANN, A. S., AND MOFFETT, J. Observations on the role lifecyle in the context of enterprise security managemen. In *7th ACM Symposium on Access Control Models and Technologies* (June 2002).

[2] A. SCHAAD, J. M., AND JACOB, J. The role-based access control system of a european bank: A case study and discussion. In *ACM Symposisum on Access Control Models and Technologies* (May 2001).

[3] ANDERSON, T. W. *An Introduction to Multivariate Statistical Analysis (Wiley Series in Probability and Statistics) (Hardcover)*. Wiley-Interscience.

[4] B. ALIDAEE, F. GLOVER, G. K. H. W. Solving the maximum edge weight clique problem via unconstrained quadratic programming. *Eur. J. Oper. Res. 181*, 2 (2007), 592–597.

[5] BERRY, M. W. *Survey of Text Mining*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[6] BERTINO, E., SAMARATI, P., AND JAJODIA, S. Authorizations in relational database management systems. In *CCS '93: Proceedings of the 1st ACM conference on Computer and communications security* (New York, NY, USA, 1993), ACM, pp. 130–139.

[7] BERTINO, E., SAMARATI, P., AND JAJODIA, S. An extended authorization model for relational databases. *IEEE Trans. on Knowl. and Data Eng. 9,* 1 (1997), 85–101.

[8] BINGHAM, E., MANNILA, H., AND SEPPÄNEN, J. K. Topics in 0-1 data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)* (Edmonton, Alberta, Canada, July 2002), D. Hand, D. Keim, and R. Ng, Eds., pp. 450–455.

[9] BLEI, D. M., NG, A. Y., JORDAN, M. I., AND LAFFERTY, J. Latent dirichlet allocation. *Journal of Machine Learning Research 3* (2003), 2003.

[10] CARR, R. D., DODDI, S., KONJEVOD, G., AND MARATHE, M. On the red-blue set cover problem. In *SODA '00: Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms* (Philadelphia, PA, USA, 2000), Society for Industrial and Applied Mathematics, pp. 345–353.

[11] COYNE, E. J. Role engineering. In *RBAC '95: Proceedings of the first ACM Workshop on Role-based access control* (New York, NY, USA, 1996), ACM, p. 4.

[12] DANTZIG, G. *Linear Programming and Extensions*. Princeton University Press, 1998.

[13] DAVID, B. I., AND N., T. L. *Numerical Linear Algebra*. Philadelphia: Society for Industrial and Applied Mathematics, 1997.

[14] DEERWESTER, S., DUMAIS, S. T., FURNAS, G. W., LANDAUER, T. K., AND HARSHMAN, R. Indexing by latent semantic analysis. *Journal of the*

American Society for Information Science 41 (1990), 391–407.

[15] DING, C., HE, X., AND SIMON, H. D. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proc. SIAM Data Mining Conf* (2005), pp. 606–610.

[16] ENE, A., HORNE, W., MILOSAVLJEVIC, N., RAO, P., SCHREIBER, R., AND TARJAN, R. E. Fast exact and heuristic methods for role minimization problems. In *SACMAT '08: Proceedings of the 13th ACM symposium on Access control models and technologies* (New York, NY, USA, 2008), ACM, pp. 1–10.

[17] FRANK, M., BASIN, D., AND BUHMANN, J. M. A class of probabilistic models for role engineering. In *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security* (New York, NY, USA, 2008), ACM, pp. 299–310.

[18] GAREY, M. R., AND JOHNSON, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[19] GEERTS, F., GOETHALS, B., AND MIELIKAINEN, T. Tiling databases. In *Discovery Science* (2004), Springer, pp. 278–289.

[20] GLOVER, F. Tabu search ł part i. *ORSA Journal on Computing 1*, 3 (1989), 190–206.

[21] GLOVER, F., KOCHENBERGER, G. A., AND ALIDAEE, B. Adaptive memory tabu search for binary quadratic programs. *Management Science 44*, 3

(1998), 336–345.

[22] GOLUB, G., AND LOAN, C. V. *Matrix Computations*. The Johns Hopkins Universtiy Press, Baltimore, MD, USA, 3rd edition, 1996.

[23] HOCHBAUM, D. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing* (1955), 2:83–97.

[24] HOCHBAUM, D. S. Simple and fast algorithms for linear and integer programs with two variables per inequality. *SIAM J. Comput. 23*, 6 (1994), 1179–1192.

[25] HOCHBAUM, D. S. Approximating clique and biclique problems. *J. Algorithms 29* (1998), 174–200.

[26] HOCHBAUM, D. S., MEGIDDO, N., NAOR, J., AND TAMIR, A. Tight bounds and 2-approximation algorithms for integer programs with two variables per inequality. *Mathematical Programming 62* (1992), 69–83.

[27] HOFMANN, T. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international conference on Research and development in information retrieval (SIGIR)* (New York, NY, USA, 1999), ACM, pp. 50–57.

[28] HOLLAND, J. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[29] HORN, R. A., AND JOHNSON, C. R. *Matrix Analysis*. 1997.

[30] HWEE TAN, A. Text mining: The state of the art and the challenges. In *In Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases* (1999), pp. 65–70.

[31] HYVÖNEN, S., MIETTINEN, P., AND TERZI, E. Interpretable nonnegative matrix decompositions. In *KDD* (2008), pp. 345–353.

[32] KARMARKAR, N. A new polynomial-time algorithm for linear programming. In *STOC '84: Proceedings of the sixteenth annual ACM symposium on Theory of computing* (New York, NY, USA, 1984), ACM, pp. 302–311.

[33] KIRKPATRICK, S., GELATT JR., C., AND VECCHI, M. Optimization by simulated annealing. *Science 220*, 4598 (1983), 671C680.

[34] KOYUTÜRK, M., AND GRAMA, A. Proximus: a framework for analyzing very high dimensional discrete-attributed datasets. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, 2003), ACM, pp. 147–156.

[35] KOYUTURK, M., GRAMA, A., AND RAMAKRISHNAN, N. Compression, clustering, and pattern discovery in very high-dimensional discrete-attribute data sets. *IEEE Trans. on Knowl. and Data Eng. 17*, 4 (2005), 447–461.

[36] KUHN, K. The hungarian method for the assgnment problem. *Navel Research Logistics Quarterly* (1955), 2:83–97.

[37] LAND, A. H., AND DOIG, A. G. An automatic method of solving discrete programming problems. *Econometrica 28*, 3, 497–520.

[38] LEE, D. D., AND SEUNG, H. S. Learning the parts of objects by non-negative matrix factorization. *Nature 401*, 6755 (October 1999), 788–791.

[39] LEE, D. D., AND SEUNG, H. S. Algorithms for non-negative matrix factorization. In *NIPS* (2000), pp. 556–562.

[40] LI, T. A general model for clustering binary data. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining* (2005), pp. 188–197.

[41] LOVASZ, L. On the ratio of optimal integral and fractional covers. *Discrete Mathematics* (1975), 13:383–390.

[42] LU, H., VAIDYA, J., AND ATLURI, V. Optimal boolean matrix decomposition: Application to role engineering. *IEEE 24th International Conference on Data Engineering* (2008), 297–306.

[43] M. DAWANDE, P. K., AND TAYUR, S. On the biclique problem in bipartite graphs. GSIA working paper 1996-04, Carnegie-Mellon University, 1997.

[44] MADEIRA, S., AND OLIVEIRA, A. Biclustering algorithms for biological data analysis: a survey. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on 1*, 1 (Jan.-March 2004), 24–45.

[45] MARLIN, B. Modeling user rating profiles for collaborative filtering. In *In NIPS*17* (2003), MIT Press.

[46] MARLIN, B., AND ZEMEL, R. S. The multiple multiplicative factor model for collaborative filtering. In *In Proceedings of the 21st International Conference on Machine learning* (2004), p. 2004.

[47] MARTIN KUHLMANN, D. S., AND SCHIMPF, G. Role mining - revealing business roles for security administration using data mining technology. In *SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies* (New York, NY, USA, 2003), ACM, pp. 179–186.

[48] MIETTINEN, P. The boolean column and column-row matrix decompositions. *Data Min. Knowl. Discov. 17*, 1 (2008), 39–56.

[49] MIETTINEN, P. On the positive–negative partial set cover problem. *Inf. Process. Lett. 108*, 4 (2008), 219–221.

[50] MIETTINEN, P., MIELIKAINEN, T., GIONIS, A., DAS, G., AND MANNILA., H. The discrete basis problem. In *Knowledge Discovery in Databases: PKDD 2006 C 10th European Conference on Principles and Practice of Knowledge Discovery in Databases* (2006), Springer, pp. 335–346.

[51] MISHRA, N., RON, D., AND SWAMINATHAN, R. On finding large conjunctive clusters. In *In Computational Learning Theory* (2003), Springer, pp. 448–462.

[52] PAULI, M., TANELI, M., ARISTIDES, G., GAUTAM, D., AND HEIKKI, M. The discrete basis problem. *IEEE Transactions on Knowledge and Data Engineering 20*, 10 (2008), 1348–1362.

[53] PEETERS, R. The maximum edge biclique problem is np-complete. *Discrete Appl. Math. 131*, 3 (2003), 651–654.

[54] PETER, J. B., GREEN, P., HIGDON, D., AND MENGERSEN, K. Bayesian computation and stochastic systems, 1995.

[55] PRELIC, A., BLEULER, S., ZIMMERMANN, P., WILLE, A., BUHLMANN, P., GRUISSEM, W., HENNIG, L., THIELE, L., AND ZITZLER, E. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics 22*, 9 (May 2006), 1122–1129.

[56] ROSS, S. M. *Simulation, Third Edition (Statistical Modeling and Decision Science) (Hardcover)*. Academic Press, 2002.

[57] SANDHU, R. S., COYNE, E. J., FEINSTEIN, H. L., AND YOUMAN, C. E. Role-based access control models. *IEEE Computer 29*, 2 (1996), 38–47.

[58] SCHLEGELMILCH, J., AND STEFFENS, U. Role mining with orca. In *SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies* (New York, NY, USA, 2005), ACM, pp. 168–176.

[59] SHEN, B.-H., JI, S., AND YE, J. Mining discrete patterns via binary matrix factorization. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, 2009), ACM, pp. 757–766.

[60] VAIDYA, J., ATLURI, V., AND GUO, Q. The role mining problem: finding a minimal descriptive set of roles. In *SACMAT* (2007), pp. 175–184.

[61] VAIDYA, J., ATLURI, V., GUO, Q., AND LU, H. Edge-rmp: Minimizing administrative assignments for role-based access control. *J. Comput. Secur. 17*, 2 (2009), 211–235.

[62] VAIDYA, J., ATLURI, V., AND WARNER, J. Roleminer: mining roles using subset enumeration. In *The 13th ACM conference on Computer and communications security* (2006), pp. 144–153.

[63] ZHANG, Z., LI, T., DING, C., AND ZHANG, X. Binary matrix factorization with applications. In *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining* (2007), pp. 391–400.

VITA

Haibing Lu

1981	Born at Dafeng, Jiangsu, China.
1998-2002	B.S., Information and Computing Science, Xi'an Jiaotong University, China.
2002-2005	M.S., Computing Mathematics, Xi'an Jiaotong University, China.
2005-2006	Full-Time Research Assistant, School of Information Systems, Singapore Management University, Singapore.
2006-2010	Graduate Assistantship, Rutgers Business School.
2007-2009	Rutgers Business School Dean Award - Ph.D. Competition.
2008	Rutgers Business School CIMIC Award (one per school).
2010-2011	Dissertation Fellowship, Rutgers Business School.
2011	Ph.D. in Management (IT Major), Rutgers University.

Publications

2006	H. Lu, Y. Li, and X. Wu. Disclosure Risk in Dynamic Two-Dimensional Contingency Tables. ICISS 2006.
	H. Lu, Y. Li, and X. Wu. Disclosure Analysis for Two-Way Contingency Tables. PSD 2006.
	Y. Li, H. Lu, and R. Deng. Practical Inference Control for Data Cubes. S&P 2006.
2008	H. Lu and Y. Li. Practical Inference Control for Data Cubes. IEEE Transaction on Dependable and Secure Computing, 5(2): 87-98.
	Y. Li and H. Lu. Disclosure Analysis and Control in Statistical Databases. ESORICS 2008.
	H. Lu, X. He, J. Vaidya, and N. Adam. Secure Construction of Contingency Tables from Distributed Data. DBSec 2008.
	H. Lu, J. Vaidya, and V. Atluri. Optimal Boolean Matrix Decomposition: Application to Role Engineering. ICDE 2008.
2009	J. Vaidya, V. Atluri, Q. Guo, and H. Lu. Role Engineering for Minimizing Administrative Assignment. Journal of Computer Security, 17 (2): 211-235.
	H. Lu, Y. Li, V. Atluri, and J. Vaidya. An Efficient Online Auditing Approach to Limit Private Data Disclosure. EDBT 2009.
	H. Lu, J. Vaidya, V. Atluri, and Yuan Hong. Extended Boolean Matrix Decomposition. ICDM 2009.
2010	X. He, H. Lu, J. Vaidya and V. Atluri. Secure Construction of Contingency Tables from Distributed Data. Journal of Computer Security.
	J. Vaidya, V. Atluri, Q. Guo, and H. Lu. Role Mining in the Presence of Noise. DBSec 2010.
2011	H. Lu, J. Vaidya, V. Atluri, H. Shin and L. Jiang. Weighted Rank-One Binary Matrix Factorization. SDM 2011.
	Y. Hong, J. Vaidya and H. Lu. Efficient Distributed Linear Programming with Limited Disclosure. DBSec 2011.
	E. Uzun, V. Atluri, H. Lu and J. Vaidya. An Optimization Model for the Extended Role Mining Problem. DBSec 2011.
	H. Lu and S. Huang. Clustering Panel Data. SDM Workshop on Marketing 2011.