CERIAS Tech Report 2010-34
Automatic Migration to Role-Based Access Control
by Ian Molloy
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

Graduate School ETD Form 9
(Revised 12/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Ian Molloy

Entitled

Automatic Migration to Role-Based Access Control

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

Ninghui Li
Chair

Elisa Bertino

Eugene Spafford

Christopher Clifton

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): Ninghui Li

Approved by: Aditya Mathur / William J. Gorman 07 June 2010
Head of the Graduate Program Date

Graduate School Form 20
(Revised 1/10)

PURDUE UNIVERSITY
GRADUATE SCHOOL

Research Integrity and Copyright Disclaimer

Title of Thesis/Dissertation:

Automatic Migration to Role-Based Access Control

For the degree of Doctor of Philosophy

I certify that in the preparation of this thesis, I have observed the provisions of *Purdue University Teaching, Research, and Outreach Policy on Research Misconduct (VIII.3.1)*, October 1, 2008.*

Further, I certify that this work is free of plagiarism and all materials appearing in this thesis/dissertation have been properly quoted and attributed.

I certify that all copyrighted material incorporated into this thesis/dissertation is in compliance with the United States' copyright law and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and save harmless Purdue University from any and all claims that may be asserted or that may arise from any copyright violation.

Ian Molloy
Printed Name and Signature of Candidate

07 June 2010
Date (month/day/year)

*Located at http://www.purdue.edu/policies/pages/teach_res_outreach/viii_3_1.html

AUTOMATIC MIGRATION TO ROLE BASED ACCESS CONTROL

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Ian M. Molloy

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2010

Purdue University

West Lafayette, Indiana

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

To Alicia and our Sixteen-Paws. You kept me going and made graduate school more enjoyable than it ought to be.

ACKNOWLEDGMENTS

My sincerest thanks to my advisor Dr. Ninghui Li for mentoring me all these years, helping me focus on my research, and allowing me to pursue random topics I found interesting. I would also like to thank my other committee members, Dr. Elisa Bertino, Dr. Eugene Spafford, and Dr. Christopher Clifton, for their helpful comments and input.

My time as a graduate student and my research has been heavily influenced by my colleagues at IBM Research. I owe a great amount of gratitude to everyone there, especially Jorge Lobo and Pau-Chen Cheng, who acted as both mentors and friends.

Additionally, I need to thank my friends Brian Bue and Nick Sumner for never being too busy to give their opinion to a problem; my parents for their love, support, and first hand anecdotes of graduate school; and my wife, Alicia, for always humoring me and letting me ramble about my research even when it didn't make any sense. Finally, I am indebted to my furry kids for keeping me grounded: Murphy, you told me when I worked too hard and should take breaks—but mostly to feed you; Porter, you told me when my keyboard was too dry and could use a little slobber; and Cayo and Luna, who always knew my laptop keyboard made for a warm place to sleep and never could figure out what all the typing was about.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

ABBREVIATIONS

ACL	*Access Control List*
AM	*AttributeMiner* Role Mining Algorithm
DUPA	*Direct User-Permission Assignment* Relation
HM	*HierarchicalMiner* Role Mining Algorithm
NMF	*Non-Negative Matrix Factorization*
PA	*Permission-Assignment* Relation
RBAC	*Role Based Access Control*
RH	*Role Hierarchy* Relation
SVD	*Singular Value Decomposition*
UA	*User-Assignment* Relation

ABSTRACT

Molloy, Ian M. Ph.D., Purdue University, August 2010. Automatic Migration to Role Based Access Control. Major Professor: Ninghui Li.

The success of role-based access control both within the research community and industry is undeniable. One of the main reasons for RBAC's adoption is its ability to reduce administration costs, help eliminate errors, and improve the security of a system. Before these advantages can be observed, an organization must first migrate their access control data over to RBAC. This is a process known as role engineering, and is potentially expensive.

We view the problem of role engineering as an optimization of the existing access control information (permission assignments, attributes, usage logs, etc.) that maximizes the return on investment in deploying RBAC given a model of a human administrator. We focus on three main objectives: the RBAC state must be compact, and minimize the costs to administer access to users; the RBAC state must contain semantically meaningful roles that correspond to real-world concepts and job duties; and the RBAC state must be correct and secure.

We develop a two-phase process for role mining: first clean the data, and then find candidate roles. Techniques based on rank-reduced matrix decomposition and a model of security are used to clean the data to eliminate errors, predict unknown values, and identify assignments that are applicable to RBAC. Second, we develop a measure of administrative costs based on the structural complexity of the RBAC system. The complexity is parameterized and makes few assumptions regarding administrative effort. Two algorithms founded in the theory of formal concept analysis are developed to minimize the RBAC state complexity while maintaining semantically meaningful roles.

This dissertation illustrates that it is possible to automatically specify an access control system, such as role-based access control, from incomplete and noisy input data consisting of allow-deny decisions and attributes describing the subjects, objects, and rights. Further, one can ensure the resulting access control system has low administrative costs as measured by the structural complexity and semantic meaning of the resulting access control state.

1 INTRODUCTION

Much of information security is, at its core, about classifying actions into two classes, e.g., allow and deny, good and bad, or safe and dangerous, and ensuring this division is not violated. The division balances the *confidentiality*, *integrity*, and *availability* of the resources based on the needs of resource owners or organizations and is specified by a *policy*. One of the primary mechanisms used to ensure a safe division between the two classes is *authorization and access control*. Access control systems make the distinction between allowed and denied at many levels and are central to the security of any organization.

In the most basic access control models, permissions are directly assigned to users. When a user enters an organization they must be assigned a set of permissions required to perform their duties. If a user changes jobs within the organization, their set of permissions is likely to change; permissions authorizing new job duties must be assigned and permissions authorizing old job duties revoked. Many organizations have thousands of employees, hundreds of applications and tens-of-thousands of permissions [1], making the assignment of permissions to users costly and error prone. Many access control models have been proposed that attempt to balance the expressibility, provable security, and ease of administration of desirable policies. New models often add levels of abstraction between the users and permissions. One such model that has gained considerable traction in academia, research, and industry, is role-based access control (RBAC).

In role-based access control permissions are assigned to roles, and roles are assigned to users. A user may use the permissions assigned to roles which they are a member of. When a user changes job functions within the organization, the roles pertaining to their old job function must be revoked, and the new roles assigned. When each role is assigned many permission, this greatly simplifies administration. Given m

permissions and n users, if we directly assign the permissions to users, and the number of permissions assigned to each user is large, then we need to maintain on the order of mn relationships. However, using RBAC, the number of relationships that we need to maintain could be reduced to the order of $(m + n)$.

Role-based access control is widely used in enterprise security management and identity management products. According to research conducted by IBM, RBAC is "creating both a valid Return On Investment (ROI) and driving better control over the assets of an organization" [2]. Attracted by strong ROI, more and more companies are driven to migrate to RBAC. However, for most companies, creating an RBAC configuration from scratch is not easy. According to a study by NIST [1], building an RBAC system is the costliest part of migrating to an RBAC implementation. Any improvement on methodology that can reduce the cost of RBAC system creation will further improve the ROI of RBAC and will accelerate RBAC's adoption in practice.

There are two general approaches to construct an RBAC system: *top-down* and *bottom-up*. In the top-down approach, system administrators, security professionals, and business domain experts, perform a detailed analysis of business processes and derive roles from such analysis. Because such a top-down analysis is human-intensive, it is believed to be slow and expensive, and potentially error prone. To overcome the drawback of top-down approaches, researchers have proposed to use data mining techniques to discover roles from existing system configuration data [3–15]. Such a bottom-up approach is called *role mining*, and can potentially accelerate RBAC system construction to a great extent.

Like all human processes, top-down role engineering is also prone to human errors. The vast size of many organizations containing thousands of users, hundreds of applications, and tens-of-thousands of permissions [1], makes it prohibitive to perform role engineering manually. While some may believe the top-down approach is more desirable despite the higher cost, because it produces higher quality results, analysis indicates that this is not always the case. Based on a number of real-world organizations, many RBAC configurations are poorly designed. In an extreme case,

a company created one role (and occasionally two roles) for each of the 486 permissions in the system, which results in 489 roles in total [16]. With such an almost one-to-one correspondence between roles and permissions, the company can hardly enjoy the advantages of RBAC.

There are several reasons that top-down approaches may sometimes fail to produce "good" RBAC systems in practice. First, building an RBAC system is challenging. It is common for people to adopt some trivial design, such as blindly creating one role for each job position of the organization regardless of whether these job positions share the same set of permissions. Second, some system designers have been deeply influenced by Discretional Access Control (DAC). When they are asked to construct an RBAC system, they tend to do it in a DAC manner, such as creating a role for each permission, thinking that the most flexibility is provided in that way. Third, many organizations do not have expertise in designing RBAC systems. They do not know what is a "good" RBAC system, and such a definition is lacking in the literature.

This dissertation considers the problem of automatically migrating to role-based access control given existing access control decisions and applicable business information. There are several challenges we consider:

- How to formally define the problem of role mining? Several works in the literature [5–8, 17–19] introduce new problem definitions and evaluation criteria. We attempt to unify these works under a common framework.
- What defines a "good" RBAC state or a "good" role mining algorithm?
- When is an authorization correct? Migrating to RBAC is not always exact, and existing access control information may contain errors.

This work is motivated by models of real-world usage of role-based access control and administrative costs. It argues a good RBAC state should have several properties that make administration efficient and secure. These include compactness, semantic meaning, correctness, stability, and applicability. These properties will be formally

defined, and a formal foundation is presented. Role mining algorithms that produce RBAC states with each of these properties are presented.

Specifically, this dissertation proposes formal concept analysis [20] as a theoretical foundation on which to base many role mining problems, and illustrates how many role mining algorithms from the literature can be expressed in terms of formal concepts. The compactness and administrative costs of an RBAC state are measured using weighted structural complexity (WSC), a parameterized flexible role mining objective that subsumes many popular complexity models in the literature [6–8, 17, 19]. WSC can be used to model both the human costs, business process costs, and technical costs of an RBAC state. It is shown that minimizing the weighted structural complexity of an RBAC state can be performed in polynomial time for some input values, yet is **NP**-hard for a wide variety of parameters of interest, even to approximate. *HierarchicalMiner*, a role mining algorithm based on formal concept analysis that heuristically minimizes the WSC of an RBAC state, is presented. *HierarchicalMiner* is a heuristic greedy algorithm that locally optimizes the RBAC state.

Semantically meaningful roles are provided by two mechanisms: the properties of formal concepts; and attributes. Formal concept analysis defines candidate roles that are maximal sets of users and permissions and arranges them in a partial ordered lattice, giving more meaning and interpretation than the "set of permissions" used elsewhere. Attributes describing users and permissions relate permissions to meaningful real-world concepts. By defining roles based on attributes, the roles are thus semantically meaningful. An extension to the *HierarchicalMiner* algorithm, called *AttributeMiner*, leverages user attributes to increase the semantic meaning of candidate roles by defining candidate roles using user attributes.

A third extension to the *HierarchicalMiner* algorithm is proposed that leverages top-down role mining analysis, creating a hybrid approach to role engineering. Semantic meaning can be interpreted from the predefined roles.

Finally, this dissertation describes a framework that divides the role mining process into two phases: data cleaning and candidate role generation. In the data cleaning

phase authorizations that are applicable to role-based access control, and authorizations that are correct, are identified. In the candidate role generation phase the cleaned input data is converted into a role-based access control state using a role mining algorithm such as *HierarchicalMiner*.

Rank-reduced matrix factorization, a popular tool in many machine learning tasks, is used to perform data cleaning and prediction. Given a noisy or incomplete[1] user-permission relation, a low rank approximation can reconstruct the most likely ground truth without being overfit to the data. Overfitting an RBAC state may create roles for legitimate, but exceptional, access or errors in the existing access control state. Experimental analysis indicates the the singular value decomposition, non-negative matrix factorization [21, 22] and logistic PCA [23], have very good performance on real-world access control datasets. If the user-permission relation contains unknown values, the reconstruction can be used to perform prediction of unknown access control decisions. A possible scenario is to perform role mining using user access logs. When additional information, such as user and permission attributes, are available, collective matrix factorization [24] is used to decrease the reconstruction error and increase predictive performance. Collective matrix factorization clusters users using both their assigned permissions and attributes; permissions are similarly clustered if permission attributes are available. To measure the susceptibility of a role mining algorithm to noise and the stability of a set of candidate roles, a distance measure between two sets of candidate roles is presented. It satisfies the required properties of a metric.

Experimental results on nine real-world datasets are used to evaluate the methods proposed in this dissertation and illustrate the effectiveness of the proposed results.

The remainder of this dissertations is organized as follows. Chapter 2 provides background on access control models and role mining. A formal framework for role mining and the *HierarchicalMiner* and related algorithms are presented in Chapter 3. How to handle noisy data and errors in access control is discussed in Chapter 4 and experimental evaluation in presented in Chapter 5. Finally, Chapter 6 concludes.

[1]Containing unknown or missing values.

2 ACCESS CONTROL, DATA MINING, AND ROLE MINING

Computer security is the endeavor of ensuring the confidentiality, integrity, and availability of the information and resources within a system. The objective is to restrict the interactions of entities with resources to only allow actions that are legitimate, and are considered safe or secure. A *security policy* specifies what actions are legitimate, or what states of the system are safe. If the system can be restricted to the set of safe states and only allows legitimate requests, then it is said to be *secure*.

A policy itself does not ensure the security of a system—it must also be enforced. Policy enforcement can be accomplished in a combination of ways: *prevention*, ensuring the system cannot enter a nonsecure state; *detection*, identifying when the system has entered a nonsecure state; or *recovery*, assessing and repairing the damage and returning the system to a secure state if possible. A *security mechanism* is a means with which the policy can be enforced. This dissertation considers the security mechanisms of *authorization* and *access control*. Authorization and access control help enforce a security policy by preventing illegitimate requests. When a request is safe or secure as specified by the security policy, we say the request is authorized.

The focus of this dissertation is on *role mining*, the application of *data mining* techniques to the state of a system and leveraged to configure an access control model that satisfies a given policy. This is a growing body of work [5–8,13,17,19] that aims at automatically learning an access control policy and converting it into a model where it can be administrated, deployed, and enforced. There are many side objectives, such as discovering errors in the policy, that will be discussed in this work.

This chapter provides the necessary background in authorization and access control, and presents the role-based access control model that will be the focus of this dissertation. This chapter will formalize the notion of a secure access control system

and policy that will be used when defining policy errors. Finally, a brief introduction on data mining and related work on role mining will be provided.

2.1 Access Control

A basic access control model is the *access control matrix*. In 1971, Lampson [25] described a very basic and general access control model that was later formalized by Harrison, Ruzzo, and Ullman [26] (the HRU model). In the HRU model, there exists a set of objects O that need protection, a set of subjects $S \subseteq O$ (called domains by Lampson) that perform operations on objects, and a finite set of rights R that authorize subjects to perform actions on objects. The state of the access control system can be represented as a matrix M where each row is a subject $s \in S$ and each column is an object $o \in O$. Each entry $M[s, o]$ gives the set of rights the subject s has to the object o. That is, if $r \in M[s, o] \subseteq R$, we say s is *authorized* to perform r on o. To update the access control matrix, there exists a finite set of commands C, such as adding or removing rights from $M[s, o]$, or creating and destroying subjects and objects. The *state* of the system in the access control matrix model can be represented as a four tuple $\langle S, O, R, M \rangle$.

An example access control matrix is shown in Figure 2.1. There are two subjects, S_1 and S_2, five objects, and three rights, read, write, and execute (`r`, `w`, `x` respectively). Consider object O_2; subject S_2 has both read and execute permissions on O_2 while subject S_1 has no permissions on O_2.

	S_1	S_2	O_1	O_2	O_3
S_1			`rw`		`w`
S_2			`r`	`rx`	`x`

Figure 2.1. An example access control matrix.

2.1.1 Capability and Access Control List

An access control matrix is often large and sparse, making it inefficient to store and difficult to maintain when subjects or objects are added and removed. Two alternative access control mechanisms that are more efficient are the capability list and the access control list.

In these mechanisms, a system only stores the cells of the matrix that are non-empty, such as $\langle S, O, R$ (subject, object, right) tuples. If the tuples are grouped by subject and the $\langle O, R$ tuples are stored with the subject, this is called a *capability list*. Grouping the tuples by object and storing $\langle S, R$ with the object is called an *access control list*.

2.1.2 Safety, Security, and Errors

In [26], Harrison, Ruzzo, and Ullman define the "safety" problem for an access control system as follows. Given an access control matrix M, a system *leaks* a right r if a right r can be assigned to a subject s on an object o that did not previously contain r. A system that cannot leak a right r is considered *safe* (a state with the leaked right is *unsafe*). Because there are often privileged trusted users that may grant permissions, such as an administrator or the *root* user on Unix-like systems, these users are removed before attempting to answer the safety question.

While the safety analysis question has been shown to be Turing Undecidable in general [26], there are many systems of interest where it is decidable. This dissertation does not consider problems of decidability, however, issues regarding safety and security will be mentioned.

Safety can be defined as the inability of the system to enter an unsafe sate. *Secure* can be defined as a system's ability to defend against attack. Many attacks against access control systems consider privilege escalation, which is subsumed by the definition of safety. The safety problem is thus an attractive definition for security as well. These definitions of safety and security assumes the system is initially in a safe state,

In large, complex organizations and access control policies, this assumption is often spurious. The initial state of the system is an approximation of the security policy that should be enforced.

In [27], Bishop et al. present a policy hierarchy that captures the notion of errors in policies, implementations, and executions. At the top of the hierarchy is an *Oracle Policy* which represents an ideal policy and has perfect and complete information. This includes a user's or subject's *intention* which cannot be known to any system. As we move down the hierarchy we begin to approximate practical policies and actual running systems (including implementation errors). The full hierarchy is presented in [27], and a modified version is shown in Figure 2.2.

The *Feasible Policy* represents policies that can be represented and modeled in any practical system. The *Model Feasible Policy* is added in this dissertation, which represents policies that can be represented using a given model. For example, in RBAC one cannot state the "Teller" role can access accounts only during business hours of 9:00AM to 5:00PM. This is because the standard RBAC model cannot be used to represent time (the RBAC model will be formally defined in Section 2.2). However, the Generalized Temporal RBAC model can [28], and is a more expressive model.

In role-mining, an access control policy at the *Configured Policy* level of the hierarchy is provided as input. A policy may be considered safe at this level, yet contains errors. The *Feasible Policy* level is the highest level attainable on a real system and closest to the ideal policy at the *Oracle Policy* level. The security of a policy may be determined by measuring the gap between the *Configured Policy* level and the *Model Feasible Policy* level. The more a policy resembles the ideal policy, the more secure it is. The detection and recovery enforcement strategies are designed to cover the policy gaps.

Unifying Policy Hierarchy		
Level	**Domain**	**Description**
Oracle Policy	all possible (s, o, a, e) tuples	Captures notion of an "ideal policy" even if such a policy isn't explicitly defined.
Feasible Policy	system-definable (s, o, a) tuples	Represents what can in practice be captured on an actual system.
Model Feasible Policy	system-definable (s, o, a) tuples	Represents what can in practice be captured on an actual system, *and* what can be expressed in a given model.
Configured Policy	system-defined (s, o, a) tuples	Represents the policy as configured on an actual system.
Real-Time Policy	system-defined (s, o, a) tuples	Represents what is possible on an actual system.

s: subject, o: object, a: action / right, e: environment / intent

Figure 2.2. Unifying Policy Hierarchy from [27].

2.1.3 Access Control Policies

There are two types of access controls that can be used: discretionary access control (DAC) and mandatory access control (MAC). In a *discretionary access control* system, the users can control the level of access in the system. A DAC system is based on two key concepts: identity and ownership. The owner of an object can specify what requests are authorized and access is controlled based on the identity of the subject. The subject that created an object is typically the owner.

Alternatively, when the system controls which requests are authorized, and the users cannot change the access, then the system is a *mandatory access control* system. In a MAC system, neither the subject nor the object owner can grant access. MAC is used in multilevel security systems where the access is determined by properties of

the subject and the object, such as the sensitivity of the object and the clearance of the subject.

2.1.4 Access Control Models

Researchers have proposed many alternative models to the access control matrix that aim at easing the administration costs while efficiently enforcing desirable properties. Three popular models are the Bell-La Padula model [29] for confidentiality, the Chinese Wall model [30] for conflict of interest, and the Clark Wilson model [31] for integrity.

The Bell-La Padula model [29] is a mandatory access control model that enforces the confidentiality security policy. Bell-La Padula formalizes multilevel security and one-dimensional information flow. Objects are given a label indicating their sensitivity (typically a classification and set of compartments) and users are given clearance (authorization) to read information up to a given sensitivity level. The sensitivity levels of the objects naturally form a lattice structure. The *simple security property* restricts a user at a given clearance level to only read objects at the same or lower sensitivity level. The $*$-*property* restricts write access to objects with the same or higher level than the subject's clearance. These two security properties ensure confidentiality.

The Chinese Wall model [30] is a hybrid confidentiality and integrity model designed to protect against breaches of conflict of interests. Objects are placed into datasets and datasets are grouped by conflict of interest classes. When a user reads an objects from one dataset, they are prevented from accessing, e.g., reading and writing, all objects from a different datasets in the same conflict of interest class.

The Clark Wilson model [31] is an integrity model whose aim is to ensure the consistency (validity) of the data. This is accomplished by first validating the consistency of the data, i.e., certifying it, producing what Clark and Wilson call *constrained data items*. The integrity of the constrained data items is ensured by only allowing

well-formed transactions, i.e., arbitrary operations on the data are not allowed, and enforcing *separation of duty*, i.e., a single subject cannot perform all actions in a well-formed transaction.

An access control model that has received considerable support from both researchers and industry, and the focus of this dissertation, is role-based access control (RBAC) [32–35]. The RBAC model will be presented next.

2.2 Role-Based Access Control

Role-based access control (RBAC) was first proposed by Ferraiolo and Kuhn [33] and was later expanded into a family of models by Sandhu et al. [34], and eventually became an ANSI INCITS standard [35]. An earlier model based on grouping permissions that resembles RBAC was created by Baldwin [32]. RBAC is a policy neutral access control model capable of expressing both DAC [36] and MAC [37] policies.

In RBAC, roles are a semantic construct that represent a job function a user may perform within an organization. A role thus embodies the permissions a user requires to perform the duties described by the role they hold. The ANSI RBAC Standard presents a core specification and a set of optional components that may be included [35]. In the ANSI standard, RBAC has four components:

- Core RBAC
- Hierarchical RBAC
- Static Separation of Duty (SSD) Constraints
- Dynamic Separation of Duty (DSD) Constraints

Core RBAC is required in all implementations of RBAC, while hierarchical RBAC, and static and dynamic separation of duty constraints may be optionally added. In Core RBAC, an administrator assigns users to roles and assigns permissions to roles. When a user activates a role, they obtain the permissions it contains.

The ANSI RBAC Standard is not a perfect model, and several flaws and shortcomings have been discovered and discussed in the literature [38,39]. For the purposes of this work, only a subset of the ANSI RBAC Standard is required and several small changes based on some of the findings of Li, Byun, and Bertino [38] are included. Specifically, this dissertation focuses on Core RBAC without sessions, with role hierarchies, and adding direct user-permission assignments that are not part of the standard. See Figure 2.3. While this work does not consider any of the numerous extensions to the RBAC model, for example [40–43], this work should be applicable as it considers the core RBAC model.

2.2.1 Core RBAC

The subset of Core RBAC used in the literature is the following. There exists a set of users $USERS$, a set of permissions $PERMS$, and a set of roles $ROLES$. For role mining, the notion of sessions is often ignored. The set of permissions is defined as the operations that may be performed on an object, i.e., $PERMS = 2^{OPS \times O}$. An operation $op \in\rangle OPS$ is semantically the same as a right in the access control matrix model. The remainder of this work will use the term *permission* to define an authorization to perform an action on a object.

Finally, the state of an RBAC system is defined by two relations over the set of users, permissions, and roles. From the ANSI RBAC standard:

- UA $USERS \times\rangle ROLES$, a many-to-many mapping user-to-role assignment relation.
- PA $PERMS \times\rangle ROLES$, a many-to-many mapping permission-to-role assignment relation.

The UA relation stores the roles that are assigned to each user, and the PA assignment stores the permissions that are assigned to each role. Between these two relations, one can determine the permissions each user is authorized.

- $\mathsf{authorized\_perms}(u : USERS) \rightarrow\rangle 2^{PERMS}$, the mapping of user u onto a set of permissions. Formally: $\mathsf{authorized\_perms}(u) = \bigcup\sum_{\not\rightarrow\langle u,r\rangle \in UA} \{\, p \mid (p,r) \in\rangle PA \,\}$

In the ANSI standard, a user initiates a session and activates a subset of their roles for use in the session. The standard defines a set of available permissions for the given session as the union of the authorized permissions for the roles activated in the session. Without sessions, the concept of authorized permissions is used instead. Note that the set of authorized permissions is the same as the set of available permissions when the user activates all of their assigned roles.

2.2.2 Hierarchical RBAC

Hierarchical RBAC introduces role hierarchies to Core RBAC. A role hierarchy is a partial order on the set of roles, i.e., $RH \quad ROLES \times\rangle ROLES$, written as $\succeq$. If $(r_1, r_2) \in\rangle RH$ then $r_1 \succeq\rangle r_2$ and r_1 is said to be *senior* to r_2, and r_2 is the *junior* role to r_1.

Implicitly a role hierarchy defines an inheritance relation. A senior role inherits the permissions assigned to its junior roles, and a junior role inherits the users assigned to its senior roles.

This is formalized by as follows:

- $\mathsf{authorized\_users}(r : ROLES) = \{\, u \in\rangle USERS \mid r' \infty\succeq\rangle r \wedge\rangle (u, r') \in\rangle UA \,\}$
- $\mathsf{authorized\_perms}(r : ROLES) = \{\, p \in\rangle PERMS \mid r \succeq\rangle r' \wedge\rangle (p, r') \in\rangle PA \,\}$
- $\mathsf{authorized\_perms}(u : USERS) = \bigcup\sum_{\not\rightarrow\langle u,r\rangle \in UA} \mathsf{authorized\_perms}(r)$

A role r_1 is the immediate senior role to r_2, denoted by $r_1 \succ\succ\rangle r_2$ if $r_1 \succeq\rangle r_2$ and there does not exist a role r_3 such that $r_1 \succeq\rangle r_3 \succeq\rangle r_2$ and $r_1 \neq r_3$ and $r_2 \neq r_3$. The RH relation stores only the inheritance relations $\succ\succ\rangle$specified by the an administrator and the partial order is the reflexive and transitive closure over RH.

As noted by Li et al. [38], the ANSI standard is potentially misleading. When introducing role hierarchies they state that:

> *"Inheritance has been described in terms of permissions; i.e., r_1 'inherits' role r_2 if all privileges of r_2 are also privileges of r_1. ... role hierarchies are managed in terms of user containment relations: role r_1 'contains' role r_2 if all users authorized for r_1 are also authorized for r_2 [35]."*

The role hierarchy facilitates user and permission inheritance, and not conversely as the above implies. We now state that $r_1 \succeq\rangle\ r_2 \Rightarrow\rangle$ $\mathsf{authorized\_perms}(r_2)$ $\mathsf{authorized\_perms}(r_1) \wedge\rangle \mathsf{authorized\_users}(r_1)$ $\mathsf{authorized\_users}(r_2)$. It should be noted that there is not an equivalence between the role hierarchy and the subset relations for authorized users and permissions, i.e., ($\mathsf{authorized\_perms}(r_2)$ $\mathsf{authorized\_perms}(r_1) \wedge\rangle$ $\mathsf{authorized\_users}(r_1)$ $\mathsf{authorized\_users}(r_2)$) $\not\Rightarrow\rangle r_1 \succeq\rangle r_2$.

The Standard describes two types of hierarchies: general role hierarchies and limited role hierarchies. A limited role hierarchy is constrained such that $\forall r, r_1, r_2 \in\rangle$ $ROLES$, $(r \succ\succ\rangle r_1 \wedge\rangle r \succ\succ\rangle r_2) \Rightarrow\rangle r_1 = r_2$. Limited role hierarchies are not considered in this work.

2.2.3 Direct User-Permission Assignments

In additional to Core RBAC and role hierarchies, direct user-permission assignments are added to the RBAC model. Direct user-permission assignments (DUPA) allow an administrator to assign permissions directly to users without an intermediate role. This extension to the RBAC model is more general. There exists many real-world implementations of RBAC that allow permissions to be directly assigned to users. Without including such permission assignments into our model, we cannot model many real-world implementations, which hinders the applicability of this work.

- $DUPA$ $USERS \times\rangle PERMS$, a many-to-many mapping user-permission relation.

These permissions are authorized in addition to the permissions a user obtains from the roles to which they are assigned. Formally,

- authorized_perms(u : $USERS$) = $\{ p \mid (u,p) \in \rangle DUPA \}$ $\cup\rangle$ $\bigcup\sum_{r(u,r)\in UA}$ authorized_perms(r)

Through the remainder of this dissertation we use the following notation. Given an RBAC state γ, the set of permissions assigned to a user or role $\varsigma\sigma$is $\mathcal{P}_\varsigma$ and the set of users assigned to a role r is $\mathcal{U}_r$. When the RBAC state is ambiguous, we will annotate with the access control configuration, such as $\mathcal{U}_r^\gamma$.

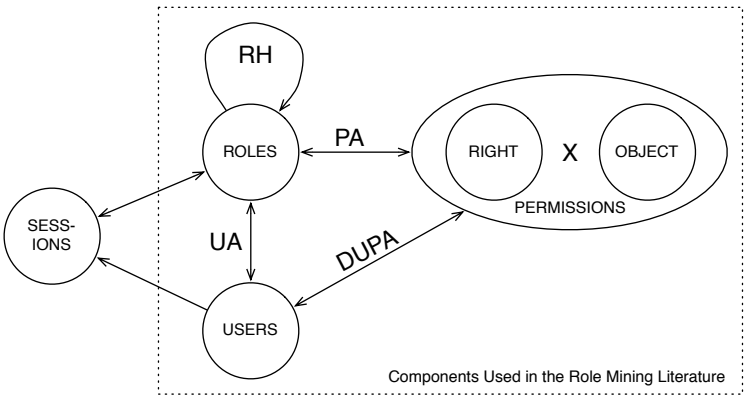


Figure 2.3. Core RBAC with a role hierarchy and direct user-permission assignments. The components relevant to role mining are shown in the dashed box.

2.2.4 Constraints

The RBAC model includes the notion of constraints which place restrictions on the set of roles a user may be assigned (static separation of duty), or may activate in a session (dynamic separation of duty). Constraints are formalized as t-out-of-n restrictions on user assignments and role activation. Static constraints state a user cannot be assigned (or authorized in the case of a role hierarchy) t or more roles from a set of n roles. Dynamic constraints state a user cannot activate t or more roles from a set of n roles in the same session. Constraints will not be considered in this work.

2.3 Role Engineering

Role-based access control offers many advantages over other access control systems. It is a conceptually simple and familiar concept for many organizations. RBAC eases the administration costs by reducing the number of assignments and revocations required when users enter and leave the system or change jobs within the organization. When used in conjunction with sessions, RBAC allows users to activate only the roles required to perform their duties under the principle of least privilege. Further, statically and dynamically mutually exclusive roles (constraints) allow separation of duty policies to be defined.

Before any of the benefits of RBAC can be realized, security administrators have to define a set of roles that reflect the duties, responsibilities and tasks within the organization. The definition of such a set of roles is known as role engineering, and was first proposed by Coyne [44]. There are two approaches to role engineering, *top-down*, and *bottom-up*. These will be discussed next.

2.3.1 Top-Down Role Engineering

Coyne [44] was the first to propose the role engineering problem and the top-down approach to role engineering. In top-down role engineering, administrators and domain experts analyze business practices and workflows of an organization. Roles should resemble the competency, duties, responsibilities, and tasks within the organization [34].

Fernandex and Hawkins [45] describe a top-down approach to role engineering using use cases. Use cases typically depict the interaction between an actor and a system to perform a task. Fernandez and Hawkins define a role by the actor and the permissions required to perform the task.

Epstein and Sandhu [46] discuss how to design roles by adding several layers of abstraction, such as jobs, work patterns, and tasks. These abstractions make it easier to decompose the business practices and requirements.

The top-down approach to role engineering is often time consuming and costly due to the extensive human involvement which can be considerable for large organizations. If administrators do not have experience in RBAC, they may often create inefficient roles. Researchers at IBM Research and HP Labs have independently found that administrators often create inefficient roles, for example creating a role for each user or each permission in the system. Often, administrators will create roles that are redundant and unnecessary, and can easily be removed without affecting the permissions assigned to each user [16, 19]. In these instances the administrators do not gain many of the advantages RBAC brings to an organization. This dissertation will focus on the bottom-up approach to role engineering, and is orthogonal to top-down role engineering.

2.3.2 Bottom-Up Role Engineering

Top-down role engineering defines roles by manually specifying the requirements of the system and determining which users and permissions satisfy them. This can be seen as role engineering "from scratch." Bottom-up role engineering leverages the existing access control information to identify latent roles that have been deployed but not formally specified. Kuhlmann, Shohat and Schimpf [3] were the first to consider bottom-up role engineering and proposed the application of data mining, coining the term *role mining*. Before describing the existing work on role mining, a brief introduction to data mining is given.

2.4 Data Mining

This section gives a brief introduction to data mining. *Data mining* is the process of extracting useful, and usually unknown and unexpected, knowledge from *data* and *information*. Data mining is often called a secondary task; the data is typically collected for a primary purpose, such as access control, auditing, or payroll, as is the

case for the data that is applicable for role mining. To draw a distinction between data and information, the following definition from Per Brinch Hansen is used:

> *"**Data:** Physical phenomena chosen by convention to represent certain aspects of our conceptual and real world. The meaning we assign to data are called information. Data is used to transmit and store information and to derive new information by manipulating the data according to formal rules [47]."*

Knowledge is a higher level form of information gained from interpreting, analyzing, combining, and understanding information from data. These concepts can be expressed as a progression:

$$Data \rightarrow\rangle Information \rightarrow\rangle Knowledge.$$

The knowledge extracted from observational data often takes the form of *models* or *patterns*. A model is a global representation of the data and can be used to make inferences about any datapoint or summarize the data. A model is often a smaller representation of the observational data, known as a *compression*. A pattern is a statement about a restricted region of the data under certain constraints. For example, a pattern may be restricted to users with a given set of attributes, such as work location and salary. Models and patterns are useful for two common data mining tasks that are applicable to role mining: prediction and clustering.

In *prediction*, one is interested in estimating the value of an unknown variable given other known inputs. For example, determining whether a user should be assigned a permission or role given a set of attributes or other permission assignments. This is not a common task in the role mining literature, but is useful in many settings.

Clustering, or grouping similar objects, is a more common task; roles are easily defined by clusters of users and permissions. Often users are clustered together, and all users in a cluster are assigned a common set of roles.

Finally, how well a given model or pattern explains the observational data must be measured. This is called a *score* or *fitness*, and is often measured by a distance

measure between the model and observed data. A model can be *overfit* to the data and may begin to capture the anomalies and not the underlying relationships. Overfitting is often the result of a poor choice of model or parameters with too many degrees of freedom.

The processing of data mining involves several steps, the three most important to the task of role mining are:

- Select a model or representation
- Select a score or fitness function
- Optimize the score function

A popular data mining technique used in many role mining algorithms is association rule mining for discovering patterns, which will be described next. For a more extensive discussion of data mining, see [48].

Association Rule Mining Association rule mining is a popular technique for discovering patterns and relations between items in large transactional datasets. In association rule mining, there exists a set of items I, and a database of transactions T. Each transaction $t \in\rangle T$ is taken from items in I, i.e., $t \quad I$. An association rule is an implication of the form $X \Rightarrow\rangle Y$, where $X, Y \quad I$ are called *itemsets* and $X \cap\rangle Y = \emptyset$. We say the support of an itemset X is the number of transactions in T that contains X, i.e., $\mathsf{supp}^T(X) = |\{\, t \in\rangle T \mid X \quad t \,\}|$, and the confidence of an association rule is $\mathsf{conf}^T(X \Rightarrow\rangle Y) = \mathsf{supp}^T(X \cup\rangle Y)/\mathsf{supp}^T(X)$. Finally, an itemset is *large* if $\mathsf{supp}^T(X) > \quad$ for threshold $\quad$. The aim of association rule mining is to identify large itemsets with high confidence implications. An itemset X is said to be *closed* if $\forall_{Y \supset X}, \mathsf{supp}^T(Y) < \mathsf{supp}^T(X)$. Two popular algorithms for identifying association rules are Apriori [49] and FP-Tree [50].

2.4.1 Machine Learning

A closely related field to data mining is machine learning. While there is no clear distinction that separates data mining from machine learning, they vary slightly in their focus and approaches. Data mining tasks are often more data driven, stemming from database applications and exploratory data analysis. Here, the focus is on discovery and extraction of unknown patterns in the dataset.

Machine learning has an emphasis on point estimation for classification and prediction, making it closer to statistics than data mining. Two common types of machine learning algorithms are *supervised* and *unsupervised* learning. In supervised learning the desired label is given in addition to the data, and the task is to predict the label of unseen data. The learning task is called regression if the label is a continuous value, and classification if the label is discrete. In unsupervised learning data labels are not given, and the objective is to learn the distribution of the data. *Semi-supervised* learning combines both labeled and unlabeled data.

Role mining tasks can fall into either supervised or unsupervised learning tasks. Identifying a suitable set of candidate roles is an unsupervised learning task because roles are not known a priori. Assigning new users to existing roles using permission and attribute assignments [51] is a supervised learning task, and predicting unknown access control decisions can be a supervised or semi-supervised learning.

2.5 Role Mining

Since 2005, most research on role engineering has focused on the bottom-up or hybrid approaches. Kuhlmann et al. [3] first proposed the use of data mining techniques and coined the term "role mining." The authors described their experiences performing role mining using an off the shelf data mining tool, *IBM's Intelligent Miner for Data*, in a seven-step process. The problem space for role mining depends on two main dimensions: the input data, and the objective, typically defined by a score or fitness function. We discuss these next.

2.5.1 Problem Domain

This section discusses the two main dimensions that define the role mining domain: input data and objective. First consider the data input dimension. At the bare minimum, one would have *user permission* information, that is, the set of users, the set of permissions, and the binary user-permission relation. For example, the access control matrix can be converted into a user-permission relation by converting each object-right tuple into a permission. An existing RBAC state can be converted to a user-permission relation by *flattening* it. A flattened RBAC state may be obtained by calculating the set of authorized permissions (authorized_perms) for each user.

In some cases, one also has *user attribute* information, e.g., a user's job title, the department and location a user is in, etc. Often, one also has *permission parameter* information, which is similar to user attribute information, but for permissions. For example, a number of permissions may be about the same enterprise information management application, or a permission may define accounts on machines in one domain. In some systems, one may have *permission update* information, i.e., from logs that record how the access control state has evolved in the past. For example, a log entry may record, at a certain time in the past, a user was assigned a number of permissions soon after the user was revoked certain permissions. This piece of information would be useful for role mining because it may reflect a job position change event. One may have *permission usage* information. For example, one may have logs showing which permissions are used and at what time. Such logs are often stored for compliance and auditing purposes. Finally, there may exist a set of *predefined roles* from an existing RBAC system or a top-down role mining process.

Next consider the problem dimension. The first problem that naturally arises is to mine an RBAC state (i.e., roles, role hierarchy, role-permission assignments, and user-role assignments) while optimizing some complexity measure. The second problem is to mine roles with good *semantic meanings*, i.e., roles that correspond to real-world concept units, e.g. a role for lecturers in the CS department. A similar

problem is to construct *parameterized roles* that correspond to categories of concepts. For example, we may create a role for lecturers with the course name as a parameter. Finally, an access control configuration may contain *noise or outliers*. For example, one may find that a permission or a role has been assigned to all but one user in the same department. It would be nice if such outliers are discovered and reported to the administrator for investigation to discover and correct potential authorization errors.

By combining the data dimension and the problem dimension, a picture on what problems can be solved (or partially solved) with different data availability is formed. A summary of the discussion from [16] is given in Table 2.1. The ability of predefined roles to be useful in various role mining tasks depends on the quality of the roles. For example, if a predefined role is created for each permission, it is unlikely they are useful for any role mining tasks. For a more complete discussion of the problem space, see [16, 52].

Table 2.1
Summary of potential role engineering problems with different information availability. "✓" indicates that the corresponding problem is worth studying and a good solution to the problem is possible; "`Limited`" indicates that a solution could be provided for the problem, but the solution may be limited without more information; an empty cell indicates that the provided information is insufficient to study a problem.

	Low Complexity	*Good Semantics*	*Parameterized Roles*	*Least Privilege*	*Detect Outliers*
User Permission Only	✓	`Limited`			`Limited`
With User-Attribute	✓	✓			✓
With Permission-Parameter	✓	✓	✓		✓
With Update Log	✓	✓			✓
With Usage Log	✓	✓		✓	✓
Predefined Roles	✓	`Limited`	`Limited`	`Limited`	`Limited`

Note that when performing role engineering or role mining, each access control policy is individually evaluated. This dissertation does not assume knowledge learned from performing role mining on one access control state can be beneficially applied to another. Further, it is not assumed two datasets, such as a firewall and file server, can be directly compared. The process of leveraging knowledge learned on one dataset to another is known as transfer learning [53], and is out of scope for this dissertation and considered future work.

2.5.2 Related Work in Role Mining

We now present several representative role mining algorithms from the literature. We roughly divide the algorithms based on their input data, output, and objectives. We will then discuss the role mining objectives used in the literature.

Role mining algorithms may be differentiated by whether they assume the input data is *clean* and *correct*, or whether it assumes the data contains noise. There are generally two types of noise in access control data that may impact a role mining algorithm. First, the data may contain errors, such as false positives and false negatives that need to be identified and corrected to ensure the security (confidentiality, integrity, and availability) and the resources. The second type of noise are legitimate exceptions that don't adhere to the motivation underlying role-based access control. More precise definitions of noise will be presented in this dissertation.

Exact Role Mining Algorithms

All role mining algorithms take a user-permission relation as input, and many find an RBAC state that represents the same level of access as this input relation. We consider these algorithms first, and describe several seminal approach to role mining.

ORCA Schlegelmilch and Steffens [4] were the first to study role mining as a new algorithmic problem and proposed the *ORCA* role mining tool. The ORCA

algorithm does hierarchical clustering on permissions. One starts with the set $S = \{\{p_1\}, \{p_2\}, \cdots, \{p_n\}\}$, where $p_1, p_2, \cdots, p_n$ are all the permissions. Iteratively, one finds a pair $s_i, s_j \in \rangle S$ such that the number of users having both s_i and s_j is the largest among all such pairs, and update S by removing s_j and s_j and merging s_i and s_j. $s_i \cup \rangle s_j$ is then added to S. This approach constructs a role hierarchy, but limits the role hierarchy to a strict tree structure such that each permission and each user can be assigned to only one role in the tree.

FastMiner and CompleteMiner Vaidya et al. [5] proposed a role mining approach that consists of two phases. The first phase generates a set of candidate roles, each of which is given by a set of permissions. They proposed *CompleteMiner*, which starts with every user's permission set, and computes the intersection of all subsets of the initial set of roles. To reduce the running time of *CompleteMiner*, they then proposed *FastMiner* which computes the set of candidate roles as all possible intersections of at most two initial roles. The second phase selects roles from the candidates based on a priority calculated from the number of users who have exactly the permissions in the role and the number of users who have a superset of the permissions.

Vaidya et al. [6] studied the problem of finding a minimal number of roles such that all user-permission relation (UP) assignments can be performed through these roles, which they refer to as the role mining problem (*RMP*, and later basic-RMP). They show that *RMP* is **NP**-complete, and is closely related with several existing data mining problems such as the minimal titling problem and the discrete basis problem. [6] then suggests that the techniques and solutions for these known problems may be used for the role mining problem. Similarly, Lu et al. [7] view the role mining problem as a decomposition of a boolean matrix. An input matrix, the user-permission relation UP, is decomposed into a user-role matrix UA and a role-permission matrix PA, such as $UP = UA \otimes \rangle PA$, where $\otimes \rangle$is defined as

$$UP_{ij} = \bigvee_{\ell=1}^{k} UA_{i\ell} \wedge \rangle PA_{\ell j}.$$

Both works suggest greedy algorithms that select roles that solve cover a maximum number of remaining relations. These techniques are limited to mining RBAC systems that do not have a hierarchy.

Graph Optimization In [8], Zhang et al. presented a heuristic algorithm for role mining. The algorithm views an RBAC state as a graph with each user, permission and role as a vertex and the user-role, role-permission, and role-role relationships as edges. The goal is to minimize the number of edges while maintaining the same connectivity. Their algorithm starts with an initial RBAC system and iteratively improves the system by identifying pairs of roles such that merging or splitting the two roles will result in a graph with a lower cost.

Frequent Permission Set Mining Colantonio et al. [10] tailor an association rule mining algorithm for the task of role mining. They attempt to minimize the administration cost of the RBAC state by choosing large roles (by support) that are assignable to many users. Their algorithm is based on the Apriori [49] association rule mining algorithm, and they call their algorithm *RBAM* (Role-Based Association-rule Mining). Colantonio et al. use Apriori to output roles that are large when viewing each user as a transaction and each permission as an item. A cost metric, similar to weighted structural complexity [16] (see Section 3.1.1), is used to determine the minimum support and prune unnecessary roles.

Zhang et al. [9] present another algorithm based on association rule mining of permission sets. Unlike Colantonio [10], they use the FP-Tree algorithm [50] instead of Apriori. When roles are assigned a large number of permissions, this makes Zhang's approach more practical[1]. Roles are defined as large itemsets over a static threshold and placed in a partial order role hierarchy lattice. If a role is not assigned users beyond those it inherits from its senior roles, the role is pruned. Their algorithm is evaluated as varies by the percentage of the original user-permission relation that

[1]Colantonio et al.'s algorithm processes the lattice pertaining to the powerset of the permissions, $\wp(P)$.

is covered with only large itemsets. Additionally they consider the number of original roles recovered and the affect of various parameters on the time required.

Biclique Covering Ene et al. [19] present a fast method to solve the *RMP* precisely in practice by performing graph reductions on the bipartite graph representing the user-permission relation. The role mining problem is viewed as a biclique cover of a bipartite graph where the *RMP* is the minimum biclique cover. They also introduce several heuristic algorithms to approximate the *RMP* and other variants, such as minimizing the number of edges (the sum of user-role and role-permission relationships), and create minimal role hierarchies of height two or three.

Mining Optimal Role Hierarchies Guo et al. [18] consider the problem of selecting an optimal role hierarchy given a set of roles. An optimal role hierarchy minimizes the size of the role-role relations such that the role hierarchy captures the complete partial order over the set of roles. That is, given two roles r_1 and r_2, if $\mathsf{authorized\_perms}(r_2)$ $\mathsf{authorized\_perms}(r_1)$ then $r_1 \succeq \rangle r_2$. This is a contrived and unrealistic problem in role engineering that likely results from the misleading role hierarchy specification in the ANSI standard.

Minimum Perturbation Role Mining Unlike the above works, Vaidya et al. [17] assumes there is an existing RBAC state or set of predefined roles. They define the minimum perturbation role mining problem where an administrator is interested in reducing the complexity of the existing RBAC state while minimizing the number of changes. Minimum perturbation role mining is a dual optimization problem: minimize the complexity of the RBAC system while maximizing the similarity between the predefined and candidate roles. The similarity between two roles is measured by the Jaccard coefficient, $\mathsf{J}(A, B) = \frac{|A \cap B|}{|A \cup B|}$, and the average maximum Jaccard coefficient between all predefined roles and the set of candidate roles as a measure of similarity between the sets of roles. The final optimization is a weighted sum of the average maximum Jaccard, and the number of roles k. A weight is used to control

the tradeoff between the two objectives and the objectives. Each objective must be normalized because the average maximum Jaccard is bounded between zero and one while role minimization is a positive integer k.

Approximate and Probabilistic Role Mining

A growing number of role mining algorithms produce an RBAC state that does not represent the same level of access as the input data. These approaches may add or remove permission assignments, granting users more or less access.

δ-Approximate Role Mining The first role mining work to discuss an approximate solution was Vaidya et al. [6]. They define the basic role mining problem (basic-RMP), to minimize the number of roles required to cover the user-permission relation, and two approximation problems: δ-RMP, to minimize the number of roles while allowing no more than $\delta\sigma$deviations (over-assignments and under-assignments); and MinNoise-RMP, to minimize the number of deviations using at most k roles.

In [7] Lu et al. provide solutions to both δ-RMP and MinNoise RMP. Their solution to δ-RMP works as follows. First, select $\delta\sigma$user-permission pairs (u, p) at random; if user u is assigned permission p, revoke the assignment, otherwise, add it. Next, they run their solution for RMP and obtain the minimal number of roles k to cover the approximation, and repeat the procedure many times. Using their approach, an optimal solution must search the entire space of $\binom{|U|*|P|}{\delta}$ user-permission assignments that may be incorrect.

Disjoint Decomposition Model Another approach is the disjoint decomposition model (DDM) [13]. In DDM, each user is assigned to a single business role, and each permission is assigned to a single functional or technical role. A two-layer role hierarchy connects business roles to technical roles, authorizing permissions to users. This is similar to the enterprise RBAC (ERBAC) model [54, 55] with the constraint that each user and each permission is assigned to only a single role. See Figure 2.4.

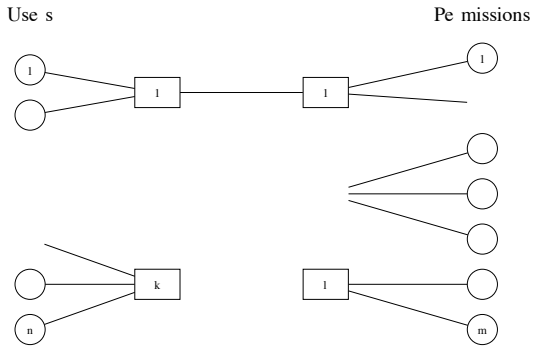


Figure 2.4. In the DDM model each user and each permission is assigned to a single role.

Frank et al. use what is called the infinite relational model (IRM) from Kemp et al. [56] to cluster users into user groups and permissions into permission groups. One disadvantage of DDM stems from the constraints placed on the user and permission assignment relations and is best explained in the context of mining clean data. DDM restricts each user to be assigned to a single business role; all users assigned to the same business role must have *exactly* the same set of permissions, and there must exist a business role for each *unique* user. This requires the creation of a large number of roles, something the IRM model penalizes by automatically selecting the number of roles. Small differences between users, or small user group, are thus placed together in larger groups. In experiments [57] it is shown this causes significant permission under-assignment, including the complete revocation of all permissions from some users.

Multi-Assignment Clustering Because DDM (and IRM) requires each user and each permission to be assigned to a single role, Streich et al. [15] propose multi-assignment clustering (MAC) for binary relations that removes this restriction. MAC works by assuming each assignment (u, p) comes from either a signal or a noise distribution, and the signal distribution allows each user to obtain a permission from

multiple clusters it is assigned. A cost function for assigning a user to a particular cluster is based on the probability the user obtains the given permission from either the signal or noise distributions. The authors use deterministic annealing to select the user-cluster assignments that minimize the assignments' costs.

Both DDM and MAC combine the noise identification and candidate role generation phases of role mining. A system administrator cannot reject the cleaning results or make further changes. When noise identification is done separately, the system administrators can be asked to confirm whether the identified noise is really noise.

Leveraging Additional Inputs

The above role mining algorithms only accept a user-permission relation as input. This restricts the types of problems role mining can attempt to solve, and the quality of the results. Since 2008, a growing number of role mining algorithms [12, 14] have accepted additional inputs, such as user attributes. Colantonio [12] use an organization hierarchy to define roles assigned to closely related users. Frank et al. [14] select roles with high attribute compliance[2] from otherwise similar roles generated with MAC [15] by maximizing the number of attributes shared by users assigned to a role. Their approach reduces permission assignment accuracy to increase attribute compliance, increasing the number of errors in order to gain semantic meaning.

Safety of Role Mining

One must consider how role mining affects the safety analysis of a system as defined by [26] (see Section 2.1.2). A system is considered safe if it is initially in a safe state and it cannot enter an unsafe state. An unsafe state is often defined by the leakage of a right r to a user u that did not have r. When performing exact role mining, the RBAC state has the same level of access as the input state, and it is easy

[2]Number of attributes shared by users assigned to a role.

to conclude if the input state was safe, the resulting RBAC state is also safe (we do not consider the impact of sessions, constraints, etc. here).

When performing inexact role mining, a user may be assigned additional permissions, and the role mining process can be seen as leaking these assignments. One cannot simply conclude that inexact role mining is unsafe, because inexact role mining operates under the assumption the input data (the original state), was in fact *not* in a safe state. That is, if the input state to role mining is not in a safe state, the results from exact role mining are not safe, and the results from inexact role mining *may* be safe. The objective here is to place an unsafe state into a safe state through the process of role mining.

2.5.3 Evaluation Criteria and Objectives

This section considers the role mining objectives and evaluation criteria that have been used in the literature. An abstract objective to "convert an existing system to RBAC" is ill defined and trivial to resolve by creating a new role for each user or permission (or both). Such a solution is clearly undesirable and a systematic objective function or criteria under which RBAC states may be compared is required.

Several evaluation criteria or metrics have been used in the literature. Vaidya et al. [5] randomly generate RBAC states and mine the flattened UP relationship. To evaluate a set of candidate roles, they count the number of original roles that are recovered. This method is only useful for evaluating an algorithm against known desirable data.

In [6], Vaidya et al. use the following formulation of role-mining. Let $UP \in\rangle \{0,1\}^{n\times m}$ with n users and m permissions such that $UP_{ij} = 1$ if and only if user i has permission j. Role-mining is defined as a decomposition of UP into the relations $UA \in\rangle\{0,1\}^{n\times k}$ and $PA \in\rangle\{0,1\}^{k\times m}$ such that $UP = UA \otimes\rangle PA$.

$$UP = UA \otimes\rangle PA, \tag{2.1}$$

where $\otimes\rangle$is defined as

$$UP_{ij} = \bigvee_{\ell=1}^{k} UA_{i\ell} \wedge\rangle PA_{\ell j},$$

creating k roles. The L_1 norm of a matrix A,

$$||A||_1 = \sum_{ij} | A_{ij} |, \tag{2.2}$$

is used to calculate the number of deviations $\delta\sigma$between the original input data UP and the mined relations $UA \otimes PA$ where $\delta\sigma = ||UP -\langle(UA \otimes\rangle PA)||_1$. They say UA and PA are δ-consistent with UP.

Using this framework, they present three evaluation criteria. First, know as the basic role-mining problem (basic-RMP), find a minimal set of roles that satisfy Equation 2.1, i.e., minimize k above. It is believed that by minimizing the number of roles, the administration costs are significantly reduced. This reduction can be significant compared to the trivial solution of creating a single role for each user or permission.

The second variant is δ-approximate RMP. In δ-RMP the objective is to minimize k such that $||UP -\rangle UA \otimes\rangle PA||_1 \leq\rangle \delta\sigma$(note that basic-RMP is a special case of δ-RMP with $\delta\sigma = 0$). The motivation behind δ-RMP is that the input UP relation is likely to contain errors in the form of over and under assignments. Such errors may be common as users change roles in the organization and not all of their new permissions are assigned or old permissions revoked. In many instances, it is easier to add new permissions and not revoke old permissions. The disadvantage of δ-RMP is it requires the administrator to know *how many* errors are present in the input data and there is no guarantee the $\delta\sigma$errors discovered are correct.

In their third criteria, MinNoise-RMP, the goal is to minimize $\delta\sigma$using at most k roles. MinNoise-RMP is useful when an administrator needs to limit the number of roles created but retain as much of the original policy as possible. The difficulty with MinNoise-RMP (and all δ-approximate solutions) is there is no guarantee that all permission assignments are equally sensitive. Such solutions may revoke permissions that adversely damage availability or grant permissions that harm confidentiality or integrity.

Two variants of RMP were later introduced. In [8], Zhang et al. suggest $||UA||_1 + ||PA||_1 + ||RH||_1$ and $||UA||_1 + ||PA||_1 + ||RH||_1 + |\,ROLES\,|$ as possible evaluation criteria. Lu et al. [7] also consider $||UA||_1 + ||PA||_1$ which they call edge-RMP, and Ene et al. [19] call edge-concentration. These complexity measures more closely model the work of an administrator because they count the assignment an administrator must make to create the UP relation.

Colantonio et al. [10] describe a cost measure to for flat RBAC states,

$$f = \alpha\,|\,UA\,| + \beta\,|\,PA\,| + \gamma\,|\,ROLES\,| + \delta\sigma \sum_{r \in ROLES} \sum c(r),$$

where $c : \mathcal{R} \rightarrow\rangle \mathbb{R}$ and $\mathcal{R}$ is the set of all possible roles. The function c allows an administrator to increase or decrease the cost associated with a specific role based on its desirability, such as underlying business processes or semantic meaning. Without domain knowledge, c maps all values to a (presumably non-negative) constant.

The probabilistic models for role mining [13–15] attempt to find user-assignment (UA) and permission-assignment (PA) relations that maximizes the data probability, $\Pr[\,UP, UA, PA \mid k\,]$. The main distinction between [13] and [15] are the constraints placed on UA and PA. In [15], similar to MinNoise-RMP, the correct number of roles k is assumed known in advance. In [13] the probability of an error in known, allowing one to approximate δ, and k is kept small. These models can be viewed as solutions to RMP (role minimization) under the assumption the input data is noisy. These approximate objectives [7, 13–15] treat over-assignments and under-assignments identically, and are inconsistent with the principle of least privilege and fail-safe defaults [58].

In [14], user attributes are added to the probabilistic model from [15]. They define a dual objective to maximize the data model probability, and maximize the number of common attributes shared by users assigned a common role. The final maximization objective is a linear combination of the data likelihood and attribute compliance. By increasing attribute compliance, reconstruction errors are introduced.

3 MINING ROLES WITH MULTIPLE OBJECTIVES

In this chapter we define the notion of weighted structural complexity to measure the complexity of RBAC systems and suggest the framework of formal concept analysis as a foundation to base many problems in role mining. Using this framework, we present a role mining algorithm that mines RBAC systems with low structural complexity and address the problem of discovering roles with semantic meaning. We study the problem in two primary settings with different information availability. When the only information is a user-permission relation, we propose to discover roles whose semantic meaning is based on formal concept lattices. We argue that the theory of formal concept analysis provides a solid theoretical foundation for mining roles from a user-permission relation. When user-attribute information is also available, we propose to create roles that can be explained by expressions of user-attributes. Since an expression of attributes describes a real-world concept, the corresponding role represents a real-world concept as well. Furthermore, the algorithms we proposed balance the semantic guarantee of roles with system complexity. Finally, we indicate how to create a hybrid approach combining top-down candidate roles.

Weighted structural complexity is a more general complexity model compared to previously proposed complexity measures in the literature [6–8]. These complexity measures are fixed, and measure a particular component of the role-based access control state. Weighted structural complexity is parameterized, giving a system administrator more control when designing an RBAC system that minimizes their administrative effort. A paramaterized objective also allows for different objectives to be used in different contexts.

Second, we show that the theory of formal concept analysis [20] provides a solid theoretical foundation for role mining, in the case that only user-permission data is available. Formal concept analysis has been applied extensively in software engineer-

ing, for example, on the problem of generating class hierarchies from non object-oriented code, which is very similar to the problem of mining role hierarchies. We develop *HierarchicalMiner* based on formal concept analysis and show that it is able to mine roles with low complexity. Unlike previous role mining algorithms, it naturally generates excellent role hierarchies. Our evaluation shows that it often generates better RBAC systems than those generated in ways similar to the top-down approach.

Third, we study the problem of role mining with users' attribute information in addition to user-permission relation, and give the first definition of mining roles with semantic information. Finally, we indicate how these methods can be extended to include predefined roles to ease interpretation and increase semantic meaning.

This chapter is organized as follows. In Section 3.1 we define the role mining problem in the context of user-permission data and present weighted structural complexity and related optimization problems. The theoretical complexity of role mining is also discussed. We present our role mining algorithm that minimizes complexity using formal concept analysis in Section 3.2, and Section 3.3 studies the problem of finding roles with semantic meanings from user-attribute data. We define the problem of hybrid role mining and extend our approaches in Section 3.4.

3.1 Using user-permission data

When the input data consists of only a user-permission relation, the role mining problem can be defined as follows.

Definition 3.1.1 (Bottom-Up Role Engineering) Given an access control *configuration* $\rho\sigma = \langle U, P, UP$, where U is a set of all users, P is a set of all permissions and $UP \quad U \times \rangle P$ is the user-permission relation, we want to find an *RBAC state* $\gamma\sigma = \langle R, UA, PA, RH, DUPA$ that is consistent with ρ.

In the state, R is a set of roles, $UA \quad U \times \rangle R$ is the user-role assignment relation, $PA \quad R \times \rangle P$ is the role-permission assignment relation, $RH \quad R \times \rangle R$ is a partial order over R, which is called a *role hierarchy*, and $DUPA \quad U \times \rangle P$ is the direct

user-permission assignment relation. The RBAC state is *consistent* with $\langle U, P, UP$, if every user in U has the same set of authorized permissions in the RBAC state as in UP.

3.1.1 The weighted structural complexity

Given the same access control configuration, many RBAC states are consistent with it. There has to be a measurement of how good an RBAC state is in order to select among them. The measure used in [6] is the number of roles needed to explain all user-permission assignment, while the measure used in [8] is the total number of edges (and edges plus vertices) when an RBAC state is visualized as a graph. The intuition is that a major advantage of using RBAC is to simplify management. Given m permissions and n users, if we directly assign the permissions to users, and the number of permissions assigned to each user is large, then we need to maintain on the order of mn relationships. However, using RBAC, the number of relationships that we need to maintain could be reduced to the order of $(m+n)$. We generalize the previous measures and propose the notion of weighted structural complexity. This complexity sums up the number of relationships in an RBAC state, with possibly different weights for different kinds of relationships.

Definition 3.1.2 (Weighted Structural Complexity) Given $W = \langle w_r, w_u, w_p, w_h, w_d$, where $w_r, w_u, w_p, w_h, w_d \in \rangle \mathbb{Q}^+ \cup \rangle \{\infty\}$,[1], the *Weighted Structural Complexity* (WSC) of an RBAC state γ, which is denoted as $wsc(\gamma, W)$, is computed as follow.

$$wsc(\gamma, W) = w_r * |R| + w_u * |UA| + w_p * |PA| + w_h * |t\_reduce(RH)| + w_d * |DUPA| \quad (3.1)$$

where $|\cdot|$ denotes the size of the set or relation, and $t\_reduce(RH)$ denotes the transitive reduction of the role-hierarchy.

[1] $\mathbb{Q}^+$ is the set of all non-negative rational numbers.

A transitive reduction is the minimal set of relationships that encodes the same hierarchy. For example, $t\_reduce\left(\{(r_1, r_2), (r_2, r_3), (r_1, r_3)\}\right) = \{(r_1, r_2), (r_2, r_3)\}$, as (r_1, r_3) can be inferred. That is $(r_i, r_j) \in t\_reduce(RH) \Leftrightarrow r_i \succ\succ r_j$.

Arithmetics involving ∞ is defined as follows: $0 * \infty = 0$, $\forall_{x \in \mathbb{Q}^+}\, x * \infty = \infty$, $\forall_{x \in \mathbb{Q} \cup \{\ \}}\, x + \infty = \infty$.

Intuitively, in role mining, we would like to find an RBAC state that has the smallest weighted structural complexity. One can adjust the weights to limit the RBAC states to be considered and to meet different optimization objectives. By setting w_h to ∞, we can force a flat RBAC state since each role inheritance relation costs ∞. By setting w_d to ∞, we forbid direct user-permission assignment, and adhere to the RBAC standard. By setting $w_r = 1$, $w_u = w_p = 0$, and $w_h = w_d = \infty$, we aim at minimizing the number of roles. A list of common minimization objectives in WSC is presented in Table 3.1.

Weighted structural complexity is a flexible complexity notion for role mining, yet it does not fully capture all existing complexity measures in the literature. Specifically, WSC cannot capture δ-RMP or MinNoise-RMP.

WSC can capture the notion of over-assigned permissions; these are permission assignments missing from the standard RBAC relations (UP,PA,RH), but are captured by direct user-permission assignments. An RBAC state with direct user-permission assignments may be seen as a δ-consistent RBAC state for $\delta\sigma = |\, DUPA\,|$. However, even if we restrict noise to over-assignments, WSC cannot minimize the number of roles while allowing *at most* $\delta\sigma$under-assignments. Doing so would require $w_d = 0$ when $|\, DUPA\,| < \delta\sigma$and $w_d = \infty$ otherwise. There are similar problems with representing MinNoise-RMP. Because of the sensitive nature of over- and under-assigning permissions, this not believe to be a shortcoming of WSC as a complexity measure for RBAC states. Handling noisy input data will be considered separately in Chapter 4.

To use WSC, an administrator can select a weight vector based on their own experience with RBAC, or select a standard weight vector, such as role or edge minimization, or minimizing the total number of relations. An administrator should select

Table 3.1
The relationship between WSC and other role mining criteria

Criteria	WSC Weight Vector
Basic-RMP	$W = \langle 1, 0, 0, 0, \infty \rangle$
Edge-RMP	$W = \langle 0, 1, 1, \infty, \infty \rangle$
Edge-RMP with Hierarchy	$W = \langle 0, 1, 1, 1, \infty \rangle$
Zhang Graph Optimization	$W = \langle 1, 1, 1, 1, \infty \rangle$

a weight vector that estimates their administration costs. If roles remain relatively static and user-assignments more dynamic, they may wish to increase w_u relative to w_p. Conversely, permission-assignments may be viewed as more sensitive than user-assignments. Selecting an appropriate weight vector is impacted by real-world constraints, such as system designs and RBAC implementation details, and can be biased by the personal feelings of an administrator. As such, we assume an appropriate weight vector W is known.

3.1.2 The Weighted Structural Complexity Optimization Problem

We have introduced the notion of weighted structural complexity (WSC) as a complexity measure of RBAC systems. Given an access control configuration $\rho\sigma$ and weight vector W, the role mining problem can be defined as the problem of optimizing the weighted structural complexity. A natural question that arises is to determine the WSC value of an optimized RBAC system. In this section, we formally define the Weighted Structural Complexity Decision Problem and study its computational complexity.

Definition 3.1.3 (Weighted Structural Complexity Optimization) The Weighted Structural Complexity Optimization (WSCO) problem is as follows: Given an access control configuration $\rho\sigma = \langle U, P, UP$ and a weight vector

$W = \langle w_r, w_u, w_p, w_h, w_d$, find an RBAC state $\gamma\sigma$that is consistent with $\rho\sigma$and minimizes $wsc(\gamma, W)$.

We first observe that the WSCO problem can be trivially solved for certain weight vectors. In Table 3.2, we summarize certain cases in which trivial optimal RBAC systems exist, and note how each may be constructed. A more detailed account is given in Appendix A.1.

Table 3.2
Cases where trivial optimal RBAC systems exist with respect to a given weight vector. In the table, c_i $(i \in \langle\{ u, p, h \})$ denotes an arbitrary non-zero number. In a row, two cells having value x indicates that the two cells take the same non-zero value.

w_r	w_u	w_p	w_h	w_d	Optimal RBAC System
0	c_u	0	c_h	$\infty\rangle$	Create a role for each user
0	0	c_p	c_h	$\infty\rangle$	Create a role for each permission
0	c_u	c_p	0	$\infty\rangle$	Create a role for each user and for each permission. Simulate user-permission assignments with a role hierarchy.
x	x	0	c_h	$\infty\rangle$	Create a role for each unique user (permission set)
x	0	x	c_h	$\infty\rangle$	Create a role for each unique permission (user set)

However, more generally the WSCO problem is **NP**-hard. To show this, we will prove that the decision problem corresponding to the WSCO problem is **NP**-complete. The decision problem is defined below; it is no harder to solve than the WSCO problem.

Definition 3.1.4 (Weighted Structural Complexity Decision Problem) Given an access control configuration $\rho\sigma = \langle U, P, UP$, $W = \langle w_r, w_u, w_p, w_h, w_d$ and a number k, the Weighted Structural Complexity Decision Problem (denoted as $\mathsf{WDP}(\rho, W, k)$) asks whether there exists an RBAC system $\gamma\sigma$that is consistent with $\rho\sigma$and $wsc(\gamma, W) \leq \rangle k$.

Clearly, if one can solve the WSCO problem, then one can find the minimal structural complexity for an access configuration and can solve the WDP. Furthermore, if we have a WDP solver, then we can use binary search to find the minimal structural complexity for an access configuration by invoking the WDP solver a linear number of times, as follows. Given a configuration ρ, let ω be the cost of using a trivial RBAC state to reproduce ρ, e.g., creating one role for each permission and assigning each user roles corresponding to the user's permissions. The minimal complexity is in the range $[0, \omega]$. One then sets $k = \omega/2$ and ask whether $\mathsf{WDP}(\rho, W, k)$ is true. If the answer is "yes", one knows the minimal complexity is in the range $[0, \omega/2]$, and if the answer is "no", it is in the range $(\omega/2, \omega]$. One needs to invoke the WDP solver $O(\log \omega)$ times, which is linear in the representation of the input ρ.

We are interested in the computational complexity of WDP when the weight vector W is fixed. That is, we would like to view WDP as a family of problems, parameterized by the weight vector W, and understand the computational complexities for different instantiations of the problem. Some special cases in this family have been known to be **NP**-hard. Vaidya et al. [6] showed that the case of minimizing the number of roles (e.g., using a none-zero value for w_r, $w_d = \infty$, and zero for all other weights) is **NP**-complete. Edge concentration, which minimizes $|UA| + |PA|$ and can be equivalently represented by setting $w_u = w_p = 1$, $w_h = w_d = \infty$, and $w_r = 0$, has also been shown to be **NP**-complete [59]. Furthermore, it has been shown that no polynomial time approximation with factor n^δ, $\delta > 0$ exists unless $\mathbf{P} = \mathbf{NP}$ [19, 60], making even approximate solutions difficult.

The above results are for specific weight combinations; as Table 3.2 illustrates, not all weight combinations are **NP**-complete to solve. Further, none of the above results allow role hierarchies and/or direct assignments. The theorem below shows that WDP is **NP**-complete for many weight combinations.

Theorem 3.1.1 $\mathsf{WDP}(\rho, W, k)$ is **NP**-complete, for any $W = \langle w_r, w_u, w_p, w_h, w_d \rangle$ satisfying the following condition: $w_d, w_p, w_h \geq w_u > 0$.

We argue that most interesting combinations of weights will satisfy the conditions in the theorem. That $w_u > 0$ means that assigning a role to a user is not free. That $w_d, w_p, w_h \geq\rangle w_u$ means that operations involving permissions (such as assigning permission to a user or to a role, or assigning a role to a role) costs more than assigning a role to a user. These assumptions are consistent with the rationale of RBAC.

First, the WDP problem is in **NP**, as one can provide an RBAC system to a deterministic Turing machine and the Turing machine can verify the consistency of the system and compute its WSC value in polynomial-time.

To show the **NP**-hardness, we need the following lemma.

Lemma 1 Given weights $\langle w_r, w_u, w_p, w_h, w_d$ such that $w_d \geq\rangle w_u > 0$, a configuration $\rho\sigma = \langle U, P, UP$, and $P_1 \quad P$ where $|P_1| \geq\rangle 2$. Let U_1 be the set of users whose set of permissions is exactly P_1 in ρ. If

$$|U_1| > \frac{(w_r + w_p|P_1|)}{w_u},$$

then any optimal RBAC system for $\rho\sigma$must include a role that is assigned exactly permissions in P_1.

Proof This lemma says that when the number of users having exactly a set P_1 of at least two permissions is large enough (i.e., satisfying the condition), then it is better to create a role for the permission set P_1, so that each user in U_1 can be assigned a single role. To prove this, we compare the costs of two cases. Case 1 is to create a role for P_1 and to assign the role to all users in U_1. Case 2 is not to create a role for P_1, and hence users in U_1 cannot be covered by a single role assignment.

In Case 1, the cost of creating a role for P_1 is at most $w_r + w_p|P_1|$, which is the cost of creating a role plus the cost of assigning all permissions in P_1 directly to it. The cost of assigning the role to users in U_1 is $w_u|U_1|$. Hence the total cost to cover U_1 under Case 1 is *at most* $w_r + w_u|U_1| + w_p|P_1|$.

In Case 2, there is no role having exactly P_1, then for every user $u \in\rangle U_1$, one of the following cases applies:

- u is assigned at least two roles. The cost is at least $2w_u$.
- u is assigned one role plus at least one permission. The cost is at least $w_u + w_d$.
- u is directly assigned all permissions in P_1. The cost is $w_d|P_1|$.

The minimum cost of covering a user is $min(2w_u, w_u + w_d, w_d|P_1|)$. Because $w_d \geq\rangle$ w_u and $|P_1| \geq\rangle 2$, we have $min(2w_u, w_u + w_d, w_d|P_1|) = 2w_u$. Hence the total cost for Case 2 is *at least* $2w_u|U_1|$.

When the upperbound of the cost for Case 1 is less than the lowerbound of the cost for Case 2, Case 1 must be the case in any optimal state. We have $w_r + w_u|U_1| + w_p|P_1| < 2w_u|U_1|$ if and only if $|U_1| > \frac{(w_r + w_p|P_1|)}{w_u}$. This lemma thus holds. ■

With Lemma 1, we are ready to prove that $\mathsf{WDP}(\rho, W, k)$ is **NP**-hard, completing the proof for Theorem 3.1.1.

Proof We reduce the **NP**-complete Set Covering to WDP using a polynomial time Turing reduction, also known as a Cook reduction. Here, a Cook reduction is a polynomial time algorithm that uses an oracle that can solve WDP to solve a Set Covering instance. A Cook reduction differs from the more common Karp reduction (i.e., a polynomial-time many-one reduction) in that we can call the WDP oracle more than once.

In Set Covering, we are given a set $S = \{e_1, \cdots, e_m\}$, a family of S's subsets $F = \{S_1, \cdots, S_n\}$ where S_i S, and an integer k, and need to determine whether there exists no more than k subsets in F such that the union of these sets cover S completely. Without loss of generality, we assume that each element in S is covered by at least one subset in F. (Otherwise, the answer to the set covering problem is a trivial "no cover".)

For any $W = \{w_r, w_u, w_h, w_p, w_d\}$ such that $w_h, w_p, w_d \geq\rangle w_u > 0$ we construct an access control configuration $\rho\sigma = \langle U, P, UP$ as follows.

- P contains m permissions, denoted by $p_1, \cdots, p_m$, where m is the number of elements in S.

- U contains two kinds of users. Users of the first kind correspond to the subsets in F. For every $i \in]1, n]$, let t_i be the smallest integer greater than $\frac{(w_r + w_p|S_i|)}{w_u}$. There are t_i users that have exactly the permissions corresponding to S_i, that is, these users have a permission p_j if and only if $e_j \in S_i$.

- The second kind of users in U consists of a single u_s that corresponds to S, that is, u_s has all permissions in P.

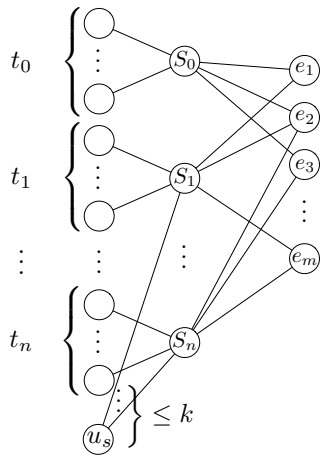


Figure 3.1. Reduction of Set Cover to WDP. User-Assignments for u_s solve Set Cover.

We can then compute c, the minimal weighted structural complexity necessary for the configuration $\rho'^{\infty} = \langle U \setminus \{ u_s \}, P, UP \rangle$ under W. Recall that we have explained that such a c can be computed by invoking a WDP oracle a linear number of times. Finally, we ask whether $\mathsf{WDP}(\rho, W, c + kw_u)$ is true or false. If it is true, then a set covering with no more than k subsets exist; if it is false, then such a set covering does not exist.

We now show that the above algorithm correctly solves the set-covering instance. Because there are t_i users having permissions corresponding to S_i, and $t_i > \frac{(w_r + w_p|S_i|)}{w_u}$, according to Lemma 1, to cover users of the first kind, an optimized RBAC system must contain a role r_i that has exactly the permissions corresponding to each S_i.

With the above argument, we show that there is an RBAC system with cost at most $c + kw_u$ if and only if the answer to the SET COVERING instance is "yes".

For the "if" direction, assume that the answer to the SET COVERING instance is "yes". There exist no more than k subsets that together cover S. Then from the RBAC state corresponding to ρ' that costs c, we can add additional assignments to cover u_s with cost $\leq\rangle kw_u$ by assigning to u_s roles corresponding to the subsets that cover S.

For the "only if" direction, assume that there is an RBAC state consistent with $\rho\sigma$while costing no more than $c + kw_u$, we need to show that a k-set covering of S exists.

First, we can assume, without loss of generality, that the RBAC state has no direct permission assignment to u_s. Because $w_d \geq\rangle w_u$, we can replace any direct permission assignment to u_s with assigning a role that contains the permission to u_s. Because any element in S is contained in at least one S_i, and any S_i is a subset of S, such replacement is feasible and results in the same permissions for u_s.

Second, we can assume, without loss of generality, that the RBAC state contains no roles that are not a subset of one or more S_is. Any such role can be assigned (directly or indirectly) only to u_s, and any such role can be removed, with sub-roles assigned to u_s (this does not increase cost because $w_h \geq\rangle w_u$), and permissions assigned to the role replaced by assigning appropriate roles to u_s (this does not increase cost because $w_p \geq\rangle w_u$). Hence every role in the RBAC state either corresponds to an S_i or is a strict subset of one or more S_is.

Third, we can assume, without loss of generality, that in the RBAC state, u_s is only assigned roles that correspond to S_is. If any role that is a strict subset of S_j is assigned to u_s, it can be replaced by assigning S_j to u_s instead. Finally, we note that u_s is assigned at most k roles. Hence, a k-set cover must exist. ■

3.2 Mining Roles with Low Complexity Using Concepts

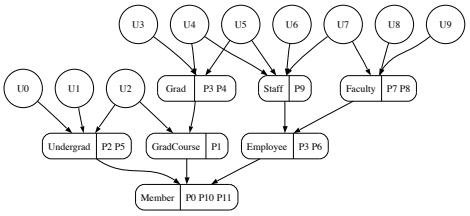
In this section, we describe a method for constructing an RBAC state with low cost and complexity, while maintaining semantic meaning. Given the role mining definition from Section 3.1.1, how should one discover roles with semantic meanings other than simply a set of permissions and a set of users that are associated with it? We examine the available techniques from data mining. The input we have is essentially a binary matrix with one dimension being the users and the other the permissions. One technique is clustering. One can cluster the users based on the similarity of their permissions, or cluster the permissions based on the similarity of the users assigned to them. These clusters can further be clustered together, resulting in a tree. One can also use co-clustering (also known as biclustering or two-mode clustering), which does simultaneous clustering of the rows and columns of a matrix. However, we believe that these techniques are not the most suitable ones for role mining. The main reason is that most clustering techniques seek to find mutually-disjoint groups. That is, each entity (user or permission) can belong to only one cluster (or appear in only one node in a tree with hierarchical clustering). But we may expect a user to be a member of at least two different roles.

Among the data mining and analysis techniques we examined, it appears that the most suitable one is formal concept analysis. (Some consider formal concept analysis to be one kind of co-clustering.) Formal concept analysis takes an input matrix specifying a set of objects and their properties, and aims to finding "concepts" in them. This is exactly the same problem as finding meaningful roles. In formal concept analysis, the concepts are arranged in a lattice. The relative relationships among concepts provide semantic information in addition to the users and permissions that are associated with them.

In this section, we give a brief introduction to formal concept analysis [20] and then develop an algorithm exploiting the connection between formal concept analysis

and mining roles with a role hierarchy. We will use the following running example in this section.

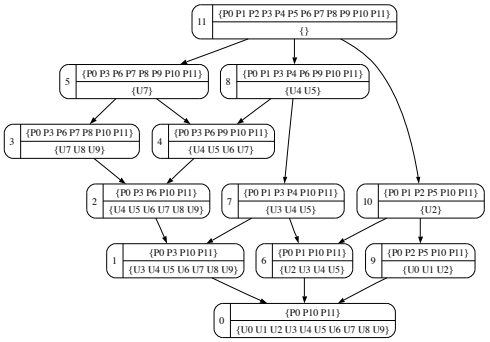
Example 1 The original RBAC state is given in Figure 3.2(a). There are 10 users, 12 permissions, and 7 roles in the original state. The user-permission relation resulted from the state is given in Figure 3.2(b).



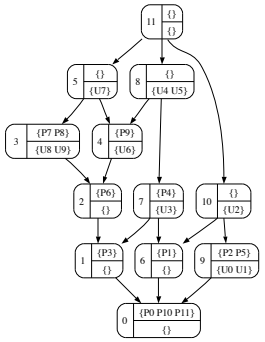
(a) Original Role Hierarchy

User	$P0$	$P1$	$P2$	$P3$	$P4$	$P5$	$P6$	$P7$	$P8$	$P9$	$P10$	$P11$
$U0$	1	0	1	0	0	1	0	0	0	0	1	1
$U1$	1	0	1	0	0	1	0	0	0	0	1	1
$U2$	1	1	1	0	0	1	0	0	0	0	1	1
$U3$	1	1	0	1	1	0	0	0	0	0	1	1
$U4$	1	1	0	1	1	0	1	0	0	1	1	1
$U5$	1	1	0	1	1	0	1	0	0	1	1	1
$U6$	1	0	0	1	0	0	1	0	0	1	1	1
$U7$	1	0	0	1	0	0	1	1	1	1	1	1
$U8$	1	0	0	1	0	0	1	1	1	0	1	1
$U9$	1	0	0	1	0	0	1	1	1	0	1	1

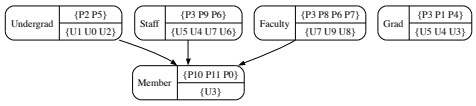
(b) User-Permission Relation



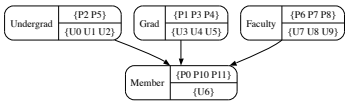
(c) The Concept Lattice



(d) Reduced Lattice



(e) Pruned role hierarchy $\langle 1, 1, 1, 1, 1$



(f) Optimal $\langle 1, 1, 1, 1, 1$

Figure 3.2. Running Example.

3.2.1 Formal Concept Analysis

The input to formal concept analysis is called a formal context.

Definition 3.2.1 (Formal Context) A *formal context* is a triple (G, M, I) where G and M are sets and $I \quad G \times \rangle M$ is a binary relation between G and M. We call the elements of G *objects* and the elements of M *attributes*. For $g \in \rangle G$ and $m \in \rangle M$, we write gIm when $(g, m) \in \rangle I$.

In role mining, the user-permission relation is a formal context, where G is the set of all users, M is the set of all permissions, and $(g, m) \in \rangle I$ if and only if the user corresponding to g has the permission corresponding to m. This is, I is the UP relation.

Definition 3.2.2 (Formal Concept) A *concept* of the context (G, M, I) is a pair (X, Y), where $X \quad G$ and $Y \quad M$ satisfy the following properties:

- $Y = \{m \in \rangle M \mid (\forall g \in \rangle X)\ gIm\}$, i.e., Y is the set of *all* properties shared by *all* objects in X.
- $X = \{g \in \rangle G \mid (\forall m \in \rangle Y)\ gIm\}$, i.e., X is the set of *all* objects that share *all* properties in Y.

We write $X^{\downarrow \infty =} Y$ and $Y^{\uparrow \infty =} X$. X is also called the *extent* and Y the *intent* of the concept (X, Y). The set of all concepts of the context is denoted by $\mathcal{B}(G, M, I)$. A concept (X_1, Y_1) is a subconcept of (X_2, Y_2), denoted as $(X_1, Y_1) \leq \rangle (X_2, Y_2)$ if and only if $X_1 \quad X_2$ (or, equivalently, $Y_1 \quad Y_2$).

For instance, in the running example, $(\{U_3, U_4\}, \{P_0, P_1, P_{10}, P_{11}\})$ is not a concept, because U_2, U_5 also have the permissions $\{P_0, P_1, P_{10}, P_{11}\}$. The pair $(\{U_2, U_3, U_4, U_5\}, \{P_0, P_1, P_{10}, P_{11}\})$ is a concept. See concept 6 in Figure 3.2(c).

The family of these concepts obeys the mathematical axioms defining a lattice, and is called a concept lattice or Galois lattice. The concept lattice for the running example is given in 3.2(c). In this concept lattice, each concept inherits all permissions associated with its subconcepts, and users are inherited in the other direction. Therefore, we can remove redundant permissions and users from each node. The

result is called the *reduced concept lattice* and is shown in Figure 3.2(d). The reduced concept lattice defines a complete RBAC state. Each concept is a role and the lattice can be viewed as the role hierarchy. In this RBAC state, each user is assigned exactly one role, and each permission is assigned to exactly one role. The subconcept relation corresponds to the role inheritance relation. When treating a concept as a role, we are assured that the permission set and the user set associated with a concept are maximum. This already has more meaning than a role with just a set of permissions and a set of users. The lattice hierarchy further illustrates the semantic relationships among concepts, helping people understand them.

Concepts are not an unnatural way to view roles. Ene et al. [19] show the role mining problem is equivalent to finding the minimum biclique cover for a bipartite graph $G(V, E)$ with $V = U \cup P$ and $E = UP$. Each concept is a maximal biclique [61, 62], and the concept lattice is the partial relation of all maximal bicliques. [19] finds the minimal set of maximal bicliques that cover UP. Is it easy to show that the set of formal concepts can be used to solve the role minimization problem.

Theorem 3.2.1 Given a set of roles R that covers the user-permission relation UP, there exists a subset R'^{∞} $\mathcal{B}(U, P, UP)$ that covers UP where $|\, R' \,| \leq\rangle |\, R \,|$.

Proof For each role r $\in\rangle$ R, we can define the concept $(\mathcal{U}_r, \mathcal{P}_r)$ which may not be maximal, i.e., not a formal concept and not in $\mathcal{B}(U, P, UP)$. We can replace r with a maximal formal concept $r'^{\infty}\in\rangle R'^{\infty}$where $r'^{\infty}= (\mathcal{P}_r^{\uparrow};\infty\mathcal{P}_r^{\uparrow\downarrow})$ $\in\rangle\mathcal{B}(U, P, UP)$. From Definition 3.2.2, we have that $\mathcal{U}_r$ $\mathcal{P}_r^{\uparrow}$, and $\mathcal{P}_r$ $\mathcal{P}_r^{\uparrow\downarrow}$. That is, for each role $r \in\rangle R$, we can make r maximal. Because two roles r_i, r_j may have the same maximal role, the total number of roles may be reduced. Thus, $|\, R' \,| \leq\rangle |\, R \,|$. ■

From Theorem 3.2.1, if R solves the role minimization problem, then there exists a set of formal concepts that also solves the role minimization problem.

Role Hierarchy and Concept Lattice Relationships

One natural question that arises is that suppose we generated a configuration from an RBAC state that contains a role hierarchy RH, and then generated the concept lattice from the configuration, how does the concept lattice relate to RH?

Observe that in the running example (Figure 3.2(a) and 3.2(c)), roles *Member*, *GradCourse*, and *Undergrad* correspond to concepts $0, 6, 9$, respectively. If users U_0 and U_1 were also assigned the *GradCourse* role, concept 9 would no longer be maximal and would not be present in the concept lattice. The extent $\{ U_0, U_1, U_2 \}$ (*Undergrad*) would have maximal intent $\{ P_0, P_1, P_{10}, P_{11} \}$ (*GradCourse* and *Undergrad*). Any role r not in the concept lattice with users $X = \mathcal{U}_r$ and permissions $Y = \mathcal{P}_r$ is either a subconcept of a role in $r'^{\infty} = (X', Y')$ where $X \subset \rangle X'^{\infty}$ and $Y \equiv \rangle Y'^{\infty}$ or is not maximal and $X \subset \rangle X'^{\infty}$ and $Y \quad Y'$. This implies that if r is not present, either every user in r is also assigned another set of permissions or there are users that are assigned the set of permissions assigned to r that are not members of r.

In the following we state some facts about the relationship between the original role hierarchy and the concept lattice.

1. For any role r in RH there is a concept (X, Y) such that $X \quad \mathcal{U}_r$ and $Y \quad \mathcal{P}_r$.

2. Given an RBAC state, if a role r has at least one unique user or one unique permission, i.e, there is a user who is assigned to r and no other role or there is a permission that is assigned to only r and no other role, then $(\mathcal{U}_r, \mathcal{P}_r)$ is a concept.

The second fact above shows that if a role is truly "unique", then it will manifest itself in the concept lattice, and one can find them by pruning the concept lattice.

The drawback of using the reduced concept lattice as the role hierarchy is that the role hierarchy may be excessively large. For example, in Figure 1(d) some concepts introduce no new users, some introduce no new permissions, and some introduce neither. However, it is incorrect to always remove all concepts with no new users

or permissions. We need to compare the desirability of different role hierarchies generated by choosing different sets of concepts from the reduced concept lattice. For this, we use weighted structural complexity.

3.2.2 The HierarchicalMiner

Our algorithm for generating a hierarchical RBAC state, which we refer to as *HierarchicalMiner*, is based on pruning the reduced concept lattice. We view the reduced concept lattice as the initial role hierarchy and heuristically optimize it based on the weighted structural complexity. Because the weighted structural complexity optimization problem in **NP**-complete, even to approximate, we choose a heuristic based solution. *HierarchicalMiner* is a greedy algorithm; it iterates over all of the roles and performs local pruning or restructuring operations if the change will decrease the cost of the RBAC state at the role. The algorithm stops when no more operations can be performed. The basic intuition behind *HierarchicalMiner* is to identify roles that add complexity to the RBAC state, and make changes that decrease this complexity. These changes include demoting user-role assignments (UA) down the hierarchy, promoting role-permission assignments (PA) up the lattice, adding and removing role hierarchy (RH) edges, and adding direct user-permission assignments ($DUPA$). Each operation ensures the RBAC state is consistent with the input access control configuration ρ, and every restructure reduces the WSC for the RBAC state.

Figure 3.2(e) shows the role hierarchy generated by *HierarchicalMiner* for our running example. *HierarchicalMiner* uses three rules to prune roles, and a forth rule to restructure the lattice.

Case 1. A role r does not have a new user or a permission assigned to it. In this case, the role is used solely as a connection point for other roles. Removing r reduces the cost for creating the role and the associated edges; however, we need to add back

some edges so that the inheritance relation remain correct. We remove r only if this is beneficial. More precisely, role r is removed when

$$w_h * \rangle (\mid Sen(r) \mid + \mid Jun(r) \mid) + w_r \geq \rangle w_h * \rangle \mid Thr(r) \mid$$

where $Sen(r)$ is the set of roles that are the immediate senior to r, $Jun(r)$ is the set of roles that are the immediate junior to r, and $Thr(r)$ is the set of pairs of roles (r_i, r_j) such that, without role r, r_i would no longer be senior to r_j.

Formally:

$$\begin{aligned} Sen(r) &= \{\, r_i \mid (r_i, r) \in \rangle RH \,\}, \text{ i.e., } r_i \succ \succ \rangle r \\ Jun(r) &= \{\, r_j \mid (r, r_j) \in \rangle RH \,\}, \text{ i.e., } r \succ \succ \rangle r_j \\ Thr(r) &= \{\, (r_i, r_j) \mid r_i \succeq \rangle r_j \wedge \rangle RH \setminus \{\, (r_i, r), (r, r_j) \,\} \Rightarrow \rangle r_i \not\succeq \rangle r_j \,\} \end{aligned}$$

Case 2. A role r has some users but no permissions assigned to it. If role r is removed, we need to assign each user in $\{u \mid (u, r) \in \rangle UA\}$ to each role in $Jun(r)$, and add $Thr(r)$ to RH to maintain the relationships among other roles. Thus role r is removed when

$$\begin{aligned} w_r + w_u * \rangle n + w_h * \rangle (\mid Sen(r) \mid + \mid Jun(r) \mid) \\ \geq \rangle w_u * \rangle n * \rangle \mid Jun(r) \mid + w_h * \rangle \mid Thr(r) \mid . \end{aligned}$$

Case 3. A role r has no users but some permissions assigned to it. If r is removed, we need to assign each permission in $\{p \mid (p, r) \in \rangle PA\}$ to each role in $Sen(r)$, and add $Thr(r)$ to RH. Thus role r is removed when

$$\begin{aligned} w_r + w_p * \rangle m + w_h * \rangle (\mid Sen(r) \mid + \mid Jun(r) \mid) \\ \geq \rangle w_p * \rangle m * \rangle \mid Sen(r) \mid + w_h * \rangle \mid Thr(r) \mid . \end{aligned}$$

In practice, we only need to propagate permissions up the role hierarchy and users down the role hierarchy when pruning a role r if the propagation does not create redundant assignments. For example, consider the role relations $\{\, (r_0 \rightarrow \rangle r_1), (r_0 \rightarrow \rangle r_2) \,\}$. If both r_1 and r_2 have a permission p_0, we do not need to assign p_0 to r_0 if we attempt to prune either r_1 or r_2, but not both.

The assignments that must be propagated to a role r' when pruning a role r is easily determined by aggregating all assignments inherited from all roles *except* r. This is easily found using a linear scan of all junior roles (in the case of permissions) and all senior roles (in the case of users) to r' in the original concept lattice before factoring. Our restructuring does not affect the users or permissions authorized to each role, but rather changes whether they are directly assigned to the role of are authorized via the role hierarchy. By only propagating the assignments that are necessary more aggressive restructuring that reduces the weighted complexity are possible without introducing redundancies that would later be removed. The restructuring benefits above may be trivially extended to include these additions.

It should be noted that the above restructuring rules do not allow for the creation of flat RBAC states or direct user-permission assignments. These shortcomings are resolved in the last case which may not prune the role.

Case 4. A role r has $n > 0$ users and $m > 0$ permissions. It may be advantageous to remove the role from the hierarchy or even remove it completely. To remove the role from the hierarchy, we perform the restructuring in both Case 2 and Case 3; permissions are propagated up the role hierarchy, users are propagated down, and $Thr(r)$ is added to RH. This restructuring is advantageous when

$$\begin{aligned} & w_h * \rangle \mid Jun(r) \mid + w_h * \rangle \mid Sen(r) \mid \\ \geq & \rangle w_h * \rangle \mid Thr(r) \mid + w_u * \rangle n * \rangle \mid Jun(r) \mid + w_p * \rangle m * \rangle \mid Sen(r) \mid . \end{aligned}$$

Additionally, we may consider removing the role completely and replace the role with direct user-permission assignments. In addition to the above restructuring to remove role r, we assign all m permissions to all n users. We may remove the role r when

$$\begin{aligned} & w_r + w_h * \rangle \mid Jun(r) \mid + w_h * \rangle \mid Sen(r) \mid \\ \geq & \rangle w_h * \rangle \mid Thr(r) \mid + w_u * \rangle n * \rangle \mid Jun(r) \mid + w_p * \rangle m * \rangle \mid Sen(r) \mid + w_d * \rangle m * \rangle n. \end{aligned}$$

For Case 4, we choose the strategy that produces the greatest improvement in WSC. When we use this last strategy we can consider all directly assigned permissions to be

a form of noise; either assignments that do not fit the RBAC model or are incorrect. These assignments may be discarded to produce an approximation of the input user-permission relation.

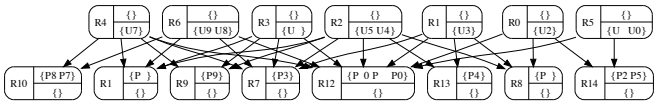
HierarchicalMiner begins by placing all roles in a queue, and processing each role according to the above four cases until the queue is empty. By removing a role r from the lattice, it may become beneficial to perform restructuring on neighboring roles that were previously overlooked because we may have modified user-, permission-, or role-assignments in the process. For each role r that is pruned or removed from the lattice, the roles $Sen(r) \cup \rangle Jun(r)$ must be added to the queue (if not already present). When the queue is empty we are assured no additional local restructuring may improve the complexity of the RBAC state.

3.2.3 Running Example

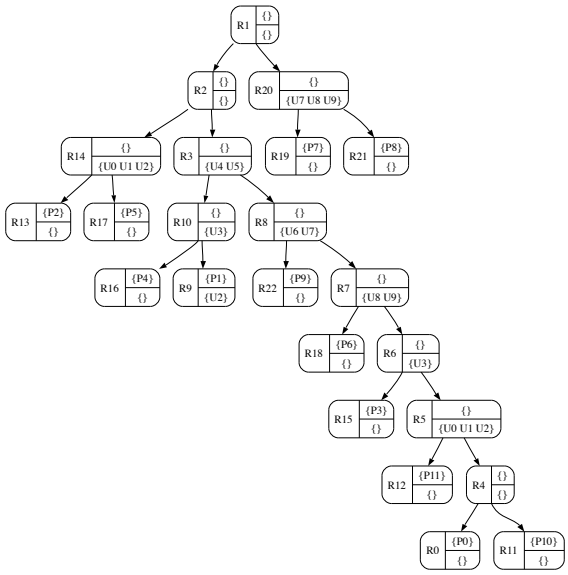
The output of *HierarchicalMiner* for the running example is given in Figure 3.2(e); and the optimal state with weight $\langle 1, 1, 1, 1, 1$ is given in Figure 3.2(f).

Using this weight vector, the original RBAC state (Figure 3.2(a)) has $wsc = 40$ and the one found by *HierarchicalMiner* (Figure 3.2(e)) has $wsc = 38$ (one direct assignment not shown), while the optimal solution (Figure 3.2(f)) has $wsc = 36$ (eight direct assignments not shown). This indicates the total number of relations that must be maintained to store the given RBAC state.

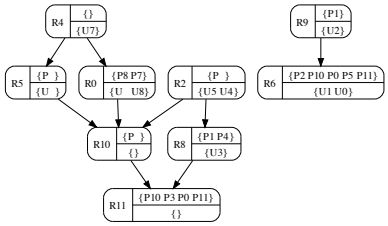
For comparison, we provide the results when using several algorithms from the literature that produce role hierarchies on our running example. Because most role mining algorithms produce a complete covering, for these examples we prevent direct user-permission assignments ($w_d = \infty$), and use $W = \langle 1, 1, 1, 1, \infty \rangle$. First, we use the constraints of the disjoint decomposition model [13] but not their probabilistic model, allowing us to assume the data is clean. This corresponds to minimizing $W = \langle 0, 1, 1, 0, \infty \rangle$, one of the trivial cases noted in Table 3.2, and has $wsc = 65$. Second, ORCA will always generate a tree based structure, assigning each permission



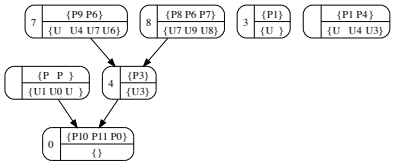
(a) Disjoint Decomposition ($wsc = 65$)



(b) ORCA ($wsc = 75$)



(c) Graph Optimization ($wsc = 45$)



(d) HierarchicalMiner ($wsc = 40$)

Figure 3.3. Role hierarchies generated from other algorithms in the literature. $W = \langle 1, 1, 1, 1, \infty \rangle$. Graph optimization has be augmented to use WSC and Disjoint Decomposition assumes clean data. RBAM created 19 roles and only two role-hierarchy relations and is omitted. Note the difference between Figures 3.2(e) and 3.3(d) when $w_d = \infty$.

to a single role. The full role hierarchy is presented in Figure 3.3(b), and has $wsc = 75$; note that it is possible to prune roles $R1$ and $R2$, leaving a forest of three trees and reducing the complexity by $w_r *\rangle 2 + w_h *\rangle 4 = 6$ to $wsc = 69$. Next, the graph optimization algorithm [8] has an interesting byproduct where each user is assigned to only a single role. The final hierarchy is shown in Figure 3.3(c), and has $wsc = 45$. Note that the intersection of roles $R6$ and $R11$, $\{ P0, P10, P11 \}$, can be identified as a new role without increasing the complexity of the RBAC state[2]. Finally we use *HierarchicalMiner*. In contrast to Figure 3.2(e), we do not allow direct user-permission assignments. We can see how this weight affects roles $R3$, $R4$, and $R5$ in Figure 3.3(d). The final state has $wsc = 40$, an increase of two. The RBAM algorithm [10] created a mostly flat RBAC hierarchy, with 19 roles and only two role-hierarchy edges. There were 81 user-assignments and 44 permission-assignments, for a total $wsc = 146$. Due to the flat shape of the hierarchy, we omit it from Figure 3.3.

3.3 Using User Attributes

We have studied how to mine roles when the available information is limited to the user-permission relation. In this section, we study mining roles when user-attribute information is also available. Examples of user-attributes include job positions, work departments, and job responsibilities. For instance, if Alice is a lecturer in the Math Department, who teaches MA101 this semester, then she has at least three attributes: *Lecturer*, *MathDepartment*, and *MA101*. Almost all organizations maintain attribute information of their employees for the purposes of administration, payroll, etc. Many organizations even display a portion of their employee's attribute information on publicly accessible websites.

[2] $6 *\rangle w_p = 3 *\rangle w_p + w_r + 2 *\rangle w_h$.

3.3.1 Semantic Meanings of Roles

We use $\mathcal{A}$ to denote the set of all attributes. The input data to role mining is modeled as a *configuration*, given by $\rho\sigma = \langle U, P, UP, UAT \rangle$, where UP is the user-permission relation, and $UAT \quad U \times \rangle\mathcal{A}$ is the user-attribute information. Here, each attribute has only a binary value of 0 or 1. While this cannot easily model numerical attributes such as age, it is general enough to model most non-numerical attributes. For example, if an attribute takes values in a tree structure (such as the division attribute) of n nodes, then this can be encoded using n binary attributes, each corresponding to a node in the tree.

Intuitively, a semantically meaningful role should correspond to a real-world concept, and a real-world concept can be described by an expression of user-attributes.

Definition 3.3.1 (Attribute Expression) An *attribute expression* e can take one of two forms:

- $e = All$: Any user satisfies e.
- $e = a_1 \wedge \rangle \cdots \wedge \rangle a_k$ $(\forall \rangle i \in \rangle [1, k] \wedge \rangle a_i \in \rangle \mathcal{A})$: A user u satisfies e under configuration $\rho\sigma$ if and only if $\forall \rangle i \in \rangle [1, k] \quad (u, a_i) \in \rangle UAT$, i.e., u has all attributes in e.

We use $\mathcal{U}_e^\rho$ to denote all the users that satisfy e.

For example, an expression $(CS \wedge \rangle Faculty)$ describes all faculty members in the CS department, while $(Programer \wedge \rangle NY \wedge \rangle Project101)$ describes all programmers in the New York branch who are working on Project 101. To incorporate the attribute information into an RBAC state, we use expressions to define the memberships of some roles. First we define the attribute compliance of a role.

Definition 3.3.2 (Attribute Compliance) For a given role r and a set of attributes A, the *attribute compliance* is the percentage of users having attribute a that are authorized to the permissions in r, i.e.,

$$\frac{|\{ u \mid \forall_{a \in A}(u, a) \in \rangle UAT \}|}{|\{ u \mid \mathcal{P}_r \quad \mathcal{P}_u \}|}.$$

Definition 3.3.3 (Membership Definition) Given a configuration $\rho\sigma = \langle U, P, UP, UAT$ and a consistent RBAC state γ, an expression e is a *membership definition* of role r if and only if $\mathcal{U}_r^\gamma = \mathcal{U}_e^\rho$.

Intuitively, a membership definition of a role represents a real-world concept the role corresponds to. However, not every role in an RBAC state has a membership definition; and a role may have more than one membership definition, as the same set of users may satisfy different expressions. A role with a membership definition has one-hundred percent attribute compliance for the attributes defined by the membership definition e.

Definition 3.3.4 (Most-General Definition) An expression e is a *most-general definition* of a role r if and only if e is a membership definition of r and there does not exist a strict sub-expression e' of e such that e' is also a membership definition of r. We assume that *All* is a strict sub-expression of any other expressions.

For example, assume that both expressions $e_1 = (NY \wedge \rangle RegularEmployee)$ and $e_2 = NY$ are membership definitions of role r (this indicates that everyone working in the New York branch is a regular employee), and *All* is not a membership definition of r. By definition, e_2 is a most-general definition of r; while e_1 is not, because e_2 is a strict sub-expression of e_1. Again, a role may have more than one most-general definitions.

If a role does not have a membership definition, we may conclude that it does not correspond to a real-world concept that can be identified by the user-attribute relation in the given configuration. In this case, we may try to see if the role describes a portion of a real-world concept.

Definition 3.3.5 (Membership Upperbound) Given a configuration $\rho\sigma = \langle U, P, UP, UAT$, let $\gamma\sigma$be an RBAC state that is consistent with ρ. An expression e is a *membership upperbound* of role r if and only if $\mathcal{U}_r^\gamma \quad \mathcal{U}_e^\rho$.

In other words, given an RBAC state and a user-attribute relation, all members of role r satisfy its membership upperbound e; but there may exist users who satisfy e but are not members of r. Note that every role has at least one membership upperbound, All. Furthermore, if a role has a membership definition, then the membership definition is also an upperbound of the role.

Intuitively, a role r corresponds to a portion of the real-world concept described by its upperbound. For example, r having an upperbound $(CS \wedge\rangle Faculty)$ indicates that members of r is a portion of all faculty members in the CS department.

Definition 3.3.6 (Least Upperbound) An expression e is the *least upperbound* of a role r if and only if e is a membership upperbound of r and there does not exist a strict super-expression e' of e such that $e'^{\infty}$is also an upperbound of r.

Given an RBAC state and a user-attribute relation, there is a unique least upperbound for each role. To prove this, assume, for the purpose of contradiction, that e_1 and e_2 are two different least upperbounds of role r. Let a_1 be an attribute that is in e_1 but not in e_2. Then, by Definition 3.3.5, $e_3 = a_1 \wedge\rangle e_2$, being a strict super-expression of e_2, is also an upperbound of r. Hence, e_2 is not a least upperbound, which is a contradiction.

Definition 3.3.7 (Least-Common Upperbound) Given a set S of roles, an expression e is the *least-common upperbound* of S if and only if e is an upperbound of every role in S, while no strict super-expression of e is.

Similar to the case of least upperbound, we can prove that there is a unique least-common upperbound for a given set of roles.

3.3.2 Attribute Miner

We now study how to construct an RBAC state exploiting the attribute information. Intuitively, during the construction of a consistent RBAC state, we would like

to use roles with membership definition whenever possible, as these roles correspond to real-world semantic concepts. Also, in order to keep the resulting state simple, we would like to minimize the number of roles that are used.

We allow two types of roles. Roles of the first type do not use attribute information, and we call these roles *normal roles*. Roles of the second type are called *attribute roles*. Every attribute role r has a membership definition $D(r)$ $\mathcal{A}$, and every user who satisfies the membership definition is assigned the role. That is to say, $\forall u \in\rangle U\ \forall r \in\rangle R$ (u satisfies $D(r)$ $\Longleftrightarrow\rangle$ $(u, r) \in\rangle UA$). Users cannot be directly assigned to attribute rules. There may be multiple attribute roles that have the same set of permissions, as those roles represent different real-world concepts that have the same permissions.

To guide our algorithm to choose attribute roles that use simple membership definitions to assign many users and to choose normal roles that have tighter upperbound constraints, we define the following complexity measure as an optimization objective.

Definition 3.3.8 (WSC with Attributes) Suppose in the RBAC state the set of normal roles is R_n and the set of attribute role is R_a. For every $r \in\rangle R_n$, the least upperbound of r is $B(r)$. For every $r \in\rangle R_a$, the membership definition of r is $D(r)$. Given $W_a = \langle w_r, w_u, w_p, w_h, w_d, w_e$, the Weighted Structural Complexity with Attribute (WSCA) of an RBAC state γ is denoted by $wsca(\gamma, W_a)$, and

$$\begin{aligned} wsca(\gamma, W_a) =& w_r *\rangle(|R_n| + |R_a|) + w_p *\rangle|PA| + w_h *\rangle|t\_reduce(RH)| + \\ & w_d *\rangle|DUPA| + w_e *\rangle\sum_{r \in R_a} |D(r)| + \\ & w_u *\rangle\sum_{r \in R_n} \left(\sqrt{|\{u \in\rangle U \mid (u, r) \in\rangle UA\}|} *\rangle\sqrt{|\{u \in\rangle U \mid u \in\rangle U(B(r))\}|}\right)\sum \end{aligned}$$

The cost of creating roles, permission assignment, role hierarchy and direct user permission assignments are the same as in Definition 3.1.2. The cost of user assignment for each normal role is determined by both the number of users authorized for this role, and the upperbound of this role. The intuition is that each user-role assignment has a cost, and the larger the upperbound, the less desirable the role is. We

choose the geometric mean of the two number, as we feel that an arithmetic mean penalizes too much.

Table 3.3
Calculating benefits and costs in AttributeMiner

Choose an attribute candidate role r	benefits	$\lvert\{(u,p) \mid u \in r.\text{users} \wedge p \in r.\text{perms} \wedge (u,p) \in UP'\}\rvert$
	cost	$w_r + w_e * \lvert r.\text{attrs}\rvert + w_p * \lvert r.\text{perms}\rvert$
Choose a normal candidate role r and a user set U_0	benefits	$\lvert\{(u,p) \mid u \in U_0 \wedge p \in r.\text{perms} \wedge (u,p) \in UP'\}\rvert$
	cost	$w_u * \lvert U_0\rvert + \begin{cases} w_r + w_p * \lvert r.\text{perms}\rvert & r.\text{users} = \phi \\ \infty & r.\text{users} \neq \phi \end{cases}$
Choose a direct assignment (u,p)	benefits	1
	cost	w_d

Our *AttributeMiner* algorithm takes a configuration as well as a list of permission sets as input. These permission sets are candidates for the algorithm to generate roles. They can be computed using HierarchicalMiner, CompleteMiner, or frequent permission set mining. The algorithm has two phases. The first phase is to identify candidate roles. For each permission set P in input, we create a candidate normal role r, and for every most general membership definition a, create a candidate attribute role using a as definition, if all users that satisfy the definition have permissions in P. The second phase selects roles and assigns them to users. Here, we use a greedy approach, and tradeoff the number of user-permission relations a role will cover (the benefit), but the increase in attribute weighted structural complexity to create such a role (the cost). This is similar to the *HierarchicalMiner* algorithm in reverse. Instead of reducing the complexity by removing roles, we select roles that have the largest benefits-cost ratio, or a return on investment. The *benefits* is the number of edges in UA'^{∞} that the selection can cover, and the *cost* is the complexity cost we need to pay to choose the selection. They are calculated as in table 3.3.

3.4 Hybrid Role Mining

In the previous sections we have illustrated how to mine an RBAC state with low complexity, and how to ensure that the roles have semantic meaning when a user-attribute relation is available. In this section we discuss how the existing techniques can be applied to instances where some roles already exist.

There are several settings where some roles already exist yet role mining is beneficial. In one setting, role mining may be used to alleviate the role proliferation problem, which may be caused by an administrator creating many unnecessary and redundant roles, increasing the complexity of the RBAC system. Role proliferation may also arise when two or more RBAC systems are merged, such as after a merger or acquisition. In Section 3.4.1, we show how role mining techniques can be used to optimize an existing deployed RBAC system that has become too large and complex.

In another setting, role mining can be used in combination with a top-down role engineering effort, to achieve a hybrid method for role engineering. Here, administrators, business professionals, and domain experts, may have already performed a cursory role engineering effort to "sketch" a set of predefined roles, e.g., defining permission sets. Role mining techniques can be then used to "fill in the gaps." In Section 3.4.2, we discuss how this can be done.

3.4.1 Optimizing an Existing Set of Roles

In this setting, the goal is to optimize the definitions of existing roles in an already deployed RBAC state while preserving the permissions they have, by merging and/or splitting roles, finding more efficient ways of assigning permissions to roles, or removing redundant role hierarchy relationships.

This can be accomplished by applying role mining techniques to the permission-assignment (PA) relation; that is, one can treat the roles as users, and mine the resulting configuration. More precisely, we create a new access configuration by creating n_r fake users for each role r such that these fake users all have exactly the

same permissions as r. We recommend choosing the number n_r based on the number of users that have the role r in the RBAC state. This way, currently widely used roles are likely to be preserved intact. From this configuration, the *HierarchicalMiner* produces a role hierarchy that provides a low-cost way to cover existing roles. For each original role r, if n_r is large enough, then a role for r is likely to also exist in the hierarchy, although the way r is defined may be optimized. If n_r is small, the role hierarchy may not have a role corresponding to r. Instead, fake users corresponding to r are covered using two or more roles, or a combination of roles and permissions. The administrator can then decide whether to remove these roles or keep them. We can then construct an RBAC state $\gamma\sigma$consistent with the original RBAC state using the optimized role definition from ξ. This approach optimizes the RBAC state without affecting role semantics or interpretation.

3.4.2 A Hybrid Approach to Role Engineering

Hybrid role engineering combines the top-down and the bottom-up approaches to role engineering. In particular, we consider the case that a top-down role engineering process already created definitions of some roles. These roles may represent the most critical or sensitive job functions and are manually specified. We would like to use role mining techniques to find the remaining roles. Furthermore, while the roles constructed through the top-down role engineering process typically have useful semantic information, a few of the permissions may be overlooked. The role mining process may be able to discover these missing permissions and complete the definitions of these roles.

We informally define the hybrid role mining problem as: Given a set of predefined roles R and access control configuration ρ, find an *RBAC state* $\gamma\sigma$that is consistent with ρ, minimizes $wsc(\gamma, W)$ for weight vector W, and includes the roles in R. Here, including "a role r in R" can be interpreted either strictly, in which case $\gamma\sigma$must contains a role with exactly the same permissions as r, or loosely, in which case $\gamma\sigma$is

required to contain a role that are semantically equivalent to r while maybe having a few more permissions missing from the definition of r. Our method also allows the flexibility of specifying some pre-defined roles as strict and others as loose.

Our approach combines traditional role mining over the user-permission relation and the approach in Section 3.4.1 for mining the permission-assignment relation, creating a hybrid method. We begin by flattening the permission-assignment relation and treat each predefined role as a user as before. To ensure the creation of each role $r \in R$, we leverage Lemma 1 and create $\left\lceil \frac{w_r + \max(w_p \mid \mathcal{P}_r \mid, w_h \mid P \mid)}{w_u} \right\rceil \sum$ users[3] with the permissions $\mathcal{P}_r$. Let $R^{*\infty}$ and $PA^{*\infty}$ be the corresponding set of users and permission assignments taken from the set of roles R with duplicated users. The final input access control configuration is $\rho'^{\infty} = \langle U \cup R^*, P, UP \cup PA^{*\infty} \rangle$. We can then apply any existing role mining algorithm to ρ'^{∞} and obtain an RBAC state, yielding the RBAC state $\gamma\sigma$ as usual.

The RBAC configuration $\gamma\sigma$ will contain both candidate roles R'^{∞} and the set of predefined roles R. If all predefined roles must be included strictly, then the artificially added users can be removed from $\gamma\sigma$ and returned as the final RBAC state, i.e., $\gamma\sigma = \langle R' \cup R, UA \setminus \{ (u, r) \mid u \in R^* \}, PA', RH, DUPA \setminus \{ (u, p) \mid u \in R^* \} \rangle$.

When some predefined roles may be incomplete, the following steps attempt to find other permissions that should also belong to the same role. For example, a predefined role may be missing permissions and is not a formal concept of the formal context $\langle U, P, UP \rangle$. By the definition of a formal concept, we know that $\mathcal{P}_r \quad \mathcal{P}_r^{\uparrow\downarrow\infty}$; $\mathcal{U}_r \quad \mathcal{P}_r^{\uparrow}$, and $(\mathcal{P}_r^{\uparrow\infty}, \mathcal{P}_r^{\uparrow\downarrow})$ is a formal concept [63]. To do this, the administrator can remove the artificially added users and continue to optimize the RBAC state, possibly pruning the predefined role and returning only its closure. If some predefined roles are favorable to their closure, e.g., the closure loses semantic meaning, may contain mutually exclusive permissions, etc., then the users assigned to these roles can be maintained during the additional optimizations. The number of artificial users can

[3]Because *HierarchicalMiner* is a heuristic algorithm, we cannot rely on Lemma 1 alone. Pruning is prevented by analyzing Cases 1–4, yielding $\frac{w_r + w_h \mid P \mid}{w_u} < n$. The proof is given in Lemma 2, Appendix A.2.

be used to turn how favorable the predefined roles are compared to the maximal formal concepts. This process is likely interactive with inputs from administrator. We will evaluate how well this approach works at completing a predefined role in Section 5.4.3.

3.5 Conclusions

This chapter studied how to mine roles and generate a role-based access control system using different types of information under a wide variety of administrative objectives. Our approaches take into account both the semantics of the roles and the complexity of the resulting access control system. We have shown that formal concept analysis provides a solid theoretical foundation for mining roles when the only user-permission information is available, and how to easily extend these methods to include other information such as user-attributes and predefined roles. We also presented the weighted structural complexity measure as a general objective that models the administrative costs associated with managing an RBAC system. The weighted structural complexity optimization and weighted structural complexity decision problems were defined, and the computational complexity for a wide variety of weights was analyzed. We developed the *HierarchicalMiner* role mining algorithm based on formal concept analysis and weighted structural complexity, and the *AttributeMiner* miner algorithm and extension to increase the semantics of the resulting RBAC states.

The approaches to role engineering presented in this chapter assume the input user-permission and user-attribute relations are *clean*. This is, we assume the input data is correct, i.e., it represents a safe state, and attempt to transition it into a role-based access control state *exactly*. In the next chapter we relax this assumption, and consider how to apply role mining to input data that may contain errors.

4 HANDLING NOISY OR MISSING ACCESS CONTROL DATA

Many role mining approaches assume the input data is clean, including *HierarchicalMiner* and *AttributeMiner*, and attempt to optimize the RBAC state. In this chapter we examine role mining with noisy input data and suggest dividing the problem into two steps: noise removal and candidate role generation. We introduce an approach to use (non-binary) rank reduced matrix factorization to identify noise and experimentally show that it is effective at identifying noise in access control data and illustrate how to use user- and permission-attributes to improve accuracy. Next, we show that our two-step approach of mining noisy data is able to find candidate roles that are close to the roles mined from noise-less data. This method performs better than the approach of mining noisy data directly and offering the administrator increased control in the noise removal and candidate role generation phases. These methods may be used to predict missing values in the input relations, allowing an administrator to input a partial specification of access control decisions. Such a solution is beneficial when access control usage logs are available indicating which permissions a subject has used, which may be a subset of the permissions they are authorized to use.

4.1 Introduction

Most role mining algorithms, such as [6, 7, 16, 19], find a set of candidate roles and a user-assignment relation such that the RBAC state represents exactly the same level of access as the input user-permission relation. Such an approach implicitly assumes clean data and uses roles to achieve a complete covering of the user-permission relation.

In reality, access control configurations in any large organization are noisy in at least two senses: first, they might contain undesirable over- and under-assignments; and second, not all permissions are assigned due to roles—some assignments may be exceptions. Hence it is necessary for role mining to deal with noisy data. Conceptually there are two ways to mine roles from noisy data. One could first clean the data and then perform role mining, or directly mine the noisy data without a cleaning step. For the latter approach, one might adapt existing role mining algorithms to prioritize the roles being selected and only use the top roles by assuming these roles are minimally affected by noise.

This chapter presents the first approach, i.e., cleaning noisy data first. In practice, role engineering often requires an organization to perform a costly data cleaning and verification step where all changes must be approved by management. It is desirable to develop techniques to partially automate the process. This chapter illustrates how to use machine learning techniques, namely various matrix decomposition models, to identify noise in access control data. The security sensitive nature of role mining exacerbates the existing challenges of using matrix decomposition for any large machine learning task. Challenges include: determining the rank of the decomposition; converting a real-valued matrix into a binary matrix; handling the risk of permission over- and under-assignment; and isolating the assignments applicable to RBAC.

This chapter presents several findings and contributions. First, singular value decomposition, non-negative matrix factorization, and logistic PCA have good performance at identifying errors in a user-permission relation and excellent predictive performance for missing values with low false positive and negative rates.

Second, the inclusion of a small amount of noise can cause significant differences in the candidate roles produced when minimizing the number of roles required for an exact solution. Our contributions include a number of new techniques for mining roles with noisy data, including how to use non-binary matrix decomposition in role mining, how to use security relevant attribute information about users and permissions to improve prediction, and selecting an appropriate rank in matrix decomposition. We

also provide a new distance measure between two candidate role sets that addresses shortcomings of methods previously used in the literature, and use it to measure the stability of roles given noise. Finally, we refine a measure for attribute importance in role mining and use attributes to improve prediction.

The rest of this chapter is outlined as follows. In Section 4.2 we provide a background on matrix decomposition, describe several models, and discuss related work. We define noise in role mining, describe our approach, and evaluate several matrix decomposition models in Section 4.3. In Section 4.4 we suggest a new distance measure for candidate role sets and discuss how noise affects two popular algorithms. Section 4.6 describes how attributes can be leveraged in noise detection and provides an evaluation for a real dataset from a large organization. Section 4.7 concludes.

4.2 Matrix Decomposition

This chapter discusses how to use approximate matrix decomposition to identify noise at varying levels. Boolean matrix decomposition (also known as matrix factorization) and rank reduction are fundamental machine learning techniques that have been applied to role mining [6, 7, 15]. In this work, given n users and m permissions, a matrix $X \in \{0,1\}^{n \times m}$ is decomposed into two matrices $A \in \mathbb{R}^{n \times k}$ and $B \in \mathbb{R}^{m \times k}$ such that $X \approx AB^T$, i.e., the distance between X and AB^T, denoted by $D(X \| AB^T)$, is minimized. One can obtain different matrix factorization models by selecting a different distance measure $D(X \| AB^T)$ and by placing restrictions on A and B [64]. Restrictions include making A and B to be non-negative, binary, or all rows sum to 1.0, etc.

The rank of matrix X is the maximum number of linearly independent columns (or rows). In matrix decomposition, when X is decomposed into two matrices $A \in \mathbb{R}^{n \times k}$ and $B \in \mathbb{R}^{m \times k}$, the rank of A and B (and hence of AB^T) is typically k. When k is large, e.g., $k = min(n, m)$, it is possible that $X = AB^T$, and $D(X \| AB^T) = 0$. Otherwise, the decomposition results in AB^T, which has lower rank than X and

is an approximation of X. Given a lower-rank approximate decomposition AB^T of X, intuitively AB^T captures the information in X that is structured and can be explained by some underlying succinct model. Hence the differences between X and AB^T identifies assignments in X that may correspond to noise.

4.2.1 Decomposition Models

We describe the matrix decomposition models evaluated in this work below, and indicate how the results can be interpreted in the context of RBAC.

Singular Value Decomposition

One of the simplest and best known matrix decompositions is the singular value decomposition (SVD). The singular value decomposition is $X = A\Sigma B^T$, where $\Sigma = \mathsf{diag}(s_1, s_2, \cdots, s_n)$ is a diagonal matrix with nonnegative real numbers on the diagonal, and A and B satisfy the following conditions: $A^T A = I_n$ and $B^T B = I_m$ where I_n and I_m are identity matrices, and A and B have orthogonal columns. The values in Σ are the singular values of the matrix X, and are uniquely determined by X. They are typically sorted in non-increasing order. Additionally, the square of each singular value (the eigenvalue) is relative to the amount of variance in X that the eigenvalue explains.

To interpret the results of a decomposition in the context of role mining, one can view it as creating pseudo-roles (or eigenroles). The rows of A give the degrees that each user is a member of the pseudo-roles, and the columns of B^T give the degrees that each permission is assigned to the pseudo-roles. In SVD, Σ gives the global importance of each pseudo-role in reconstructing the complete user-permission relation. Because these degrees are real numbers, and possibly negative, prevents the results from being directly interpreted as an RBAC state.

The singular value decomposition is a popular algorithm for performing the rank reduction of a matrix. If $\Sigma_k = \mathsf{diag}(s_1, s_2, \cdots, s_k, 0, \cdots, 0)$, then $\hat{X} = A\Sigma_k B^T$ is the

closest rank-k approximation to X minimizing the Frobenius norm, $D(X \;\|\rangle\, \hat{X}) = \left|\left|X - \rangle \hat{X}\right|\right|_{\sum}^{2} = \sqrt{\sum_{ij}(X_{ij} - \rangle \hat{X}_{ij})^2}$.

Non-Negative Matrix Factorization

The singular value-decomposition makes it difficult to interpret the A and B matrices, especially in the context of RBAC. Instead, we can constrain A and B to be non-negative, resulting in non-negative matrix factorization (NMF) [21]. To contrast models with negative assignments, NMF can be viewed as learning a decomposition as a sum of parts. This is more consistent with RBAC, which lacks negative assignments and a user's set of authorized permissions is the disjunction of the permissions of their authorized roles.

We use an implementation of NMF from Matlab (`nnmf`) that minimizes the root-mean-squared error and Bayesian NMF (BNMF) from Schmidt et al. [22] that maximizes the probability $\Pr[\, X \mid A, B\,]$ under the assumption that the data is normally distributed.

It is known that many NMF implementations are sensitive to initial conditions and may converge to local optimal solutions. Because of this we run NMF ten times and select the best decomposition.

Binary Non-Orthogonal Decomposition

We can further constrain A and B to be boolean matrices, allowing us to directly interpret the results in the context of RBAC. Binary non-orthogonal matrix decomposition [65] decomposes a matrix X by successive rank-one decompositions $X = ab^T$ where a and b are binary vectors that minimize the Hamming distance. That is, it finds a single role that most closely approximates the remaining uncovered user-permission relation. If ab^T is a satisfactory approximation (as measured by the Hamming radius, the maximum hamming distance between a vector and all columns of a matrix), then ab^T is returned as the factorization. Otherwise, the vector a is

used to partition X into two sub-matrices of disjoint users and a and b are discarded. To partition X using a, row i of X is added to sub-matrix X_1 if $a_i = 1$, otherwise it is added to sub-matrix X_0.

The decomposition recurses on each sub-matrix X_0, X_1 until the Hamming radius drops below threshold e. Finally, the rank-one approximations for each sub-matrix are combined into a final approximation $X \approx AB^T$.

Logistic PCA

The above matrix factorization models are linear models where $X = AB^T$. There exist matrix factorization models that use a non-linear prediction link where $X = f(AB^T)$. Because an access control system is a binary relation, we use logistic PCA [23], where $f(\theta) = (1 + e^{-\theta})^{-1}$ and the loss function is $D(X \| \hat{X} = f(AB^T)) = \sum_{ij} \left(X_{ij} \log \frac{X_{ij}}{\hat{X}_{ij}} + (1 - X_{ij}) \log \frac{1-X_{ij}}{1-\hat{X}_{ij}} \right)$. Instead of assuming the data is normally distributed, logistic PCA assumed a multivariate Bernoulli distribution. The Bernoulli distribution for $x \in \{0, 1\}$ with mean p is given by $\Pr[\, x \mid p \,] = p^x (1 - p)^{1-x}$.

Collective Matrix Factorization

So far we have described several methods to decompose a single user-permission relation X. This can make detecting noise difficult when the user-permission relation is sparse, or when an administrator forgets to assign or revoke many permissions from a user, such as when a user changes positions within the organization. We can often improve our accuracy if attributes are available.

Singh and Gordon [24] describe a collective matrix factorization method that we can use to improve noise detection and predictive accuracy when we have attributes about users or permissions. In the collective matrix factorization framework we have two or more matrices that share common domains, and we wish to leverage one relation to improve the predictive accuracy of another. Assume we have ℓ attributes about users, and a user-attribute matrix $Y \in \mathbb{R}^{\ell \times n}$. Matrix Y could be real-valued, for

example containing salary information, ranking, etc., or binary, indicating which attributes a user has. We can collectively factor the user-attribute and user-permission matrices such that $Y = CA^T$ and $X = AB^T$ and minimize a linear combination of their losses, $\alpha D(Y \| CA^T) + (1 - \alpha) D(X \| AB^T)$. Here α is a mixing parameter of the importance of reconstructing X versus Y. We can view A as a matrix that collectively groups users by both attributes and permission assignments, and by combining these two pieces of information we can improve accuracy and reduce the number of false positives and false negatives. We can also include a permission-attribute matrix $Z \in \mathbb{R}^{m \times o}$ such that $Z = BD^T$. We only consider the former case in the remainder of this work.

4.3 Identifying Noise

In this section we will define noise and present our approach to use the matrix decomposition models described in the previous section to identify noise prior to role mining.

4.3.1 What is Noise?

The term *noise* in access control is ambiguous and may have different meanings to different readers. We consider two possible meanings: correctness, and RBAC applicable.

Correctness Noise

There are some user-permission assignments that many would consider errors. These are Type I errors (false positives) and Type II errors (false negatives) that impact the security of the applications they are intended to protect.

A Type I error occurs when a user has been over-assigned permissions and has gained additional privileges. Type I errors are often the result of users changing

job functions in the organization. As an employee remains with an organization the jobs they perform and the projects they work on change over time. When the employee moves from one project to the next, not all of their old permissions are correctly revoked. This provides the employee with an increasing set of permissions and entitlements and poses a security threat. The security risks associated with not correctly revoking privileges can often be great when an employee's new and old jobs contain mutually exclusive permissions. These can also be leaked rights.

Type II errors are often the result of employees joining the organization or joining a new project, division of the company, etc. If the employee is not granted all the permissions required for their job, there may be a loss of availability, a reduction in productivity, and often resulting in costly calls to support. There are several works that seek to reduce the costs associated with incorrectly denying requests [51, 66].

Balancing the allowed and denied actions is a very delicate process with many tradeoffs. Reducing Type I errors too aggressively will likely result in an increase in Type II errors, and vice versa. For some algorithms, one can carefully tune the parameters to balance how many Type I and Type II errors to output.

Correctness errors can be viewed as gaps between layers on the Unifying Policy Hierarchy [27] from Figure 2.2. The input user-permission relation is a security policy at the configured policy level, while a correct policy is at the model level that is closest to the oracle policy. The distance between the configured policy and the model feasible policy depends on a distance measure that balances Type I and Type II errors.

Any permission assignment represents a balance of the risks of assigning a permission to a user with the trust the user will perform correctly and honestly. The classification of a user-permission pair as a Type I or Type II error depends on the interpretation of risk and trust and by the administrator. Here, *risk* is defined as the expected negative outcome. For example, if there is a probability of 0.1% that granting permission p_1 to user u_1 will leak information, resulting in \$5,000 in damage, the expected harm is \$5.00. *Trust* is a belief the system (or user) will behave honestly and correctly with a given amount of risk.

RBAC Applicability Noise

RBAC is not a panacea for access control, and it may not be suitable for all access control needs, such as when sharing is low. We now consider how to identify assignments that are applicable for roles in RBAC.

One can view RBAC as a compression of the access control matrix into the a set of roles and user-assignment relations. A role mining algorithm is then a (typically lossless) compression algorithm; the more compressible the data, the more applicable it is for RBAC. There is also a *law of diminishing returns* for RBAC; a small number of roles can typically cover a larger percentage of the user-permission relation, while complete coverage requires significantly more roles. Each new role contributes a smaller portion of the user-permission relation [67].

The diminishing returns of adding more roles is known to professional role engineers who often speak of an *80–20 rule* for RBAC. This could be loosely interpreted as 80% of the assignments need to be covered via roles while the remaining 20% are *exceptions*. These exceptions do not "fit the mold" of the RBAC model and may or may not be correct. Many datasets we have analyzed, such as the one we use in Section 5.5.1, have a very long tailed distribution, where many permissions are assigned to one-or-two users. For example, the **anonymous** dataset contains over three-thousand users and permissions, and a density of only 0.74%. Over two-thousand permissions are assigned to fewer than ten users and over twelve-hundred are assigned to fewer than five. Four-hundred nine permissions are assigned to a single user. The sparsity and long tail, shown in Figure 4.1, motivate the exception models like the *80–20 rule*. We differentiate an exception from an error by the intention of the administrator.

Noise and Safety

As mentioned in Section 2.5.2, one objective of noise removal is to place the input user-permission relation into a safe state. One advantage of the two-step approach presented is it allows an administrator to validate, approve, or reject the noise removal

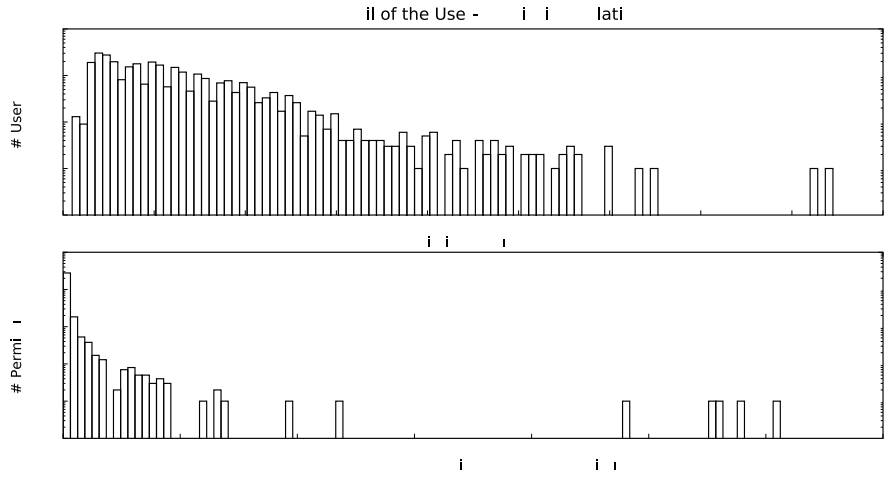


Figure 4.1. Long tail of the user-permission relation for the **anonymous** dataset. Note the y-axis is on a log-scale.

results. Once the results have been validated one can assume the input data is in a safe state and an exact role-mining algorithm may be used. Inexact role mining algorithms, if nondeterministic, may make different noise removal suggestions for each iteration. This makes noise removal and role mining difficult when attempting to ensure safety; the results from one role mining experiment may be validated and considered safe while another with identical parameters may be unsafe.

Missing and Unknown Values

In many instances an incomplete view of the user-permission relation is available, making role mining difficult due to the increased uncertainty. For example, log data indicating what actions a user has performed in the past, and whether these actions were allowed or denied, may be available. If the access control state has not been correctly maintained, e.g., a user's permission set is monotonically increasing, the access control logs will be a strictly smaller set of permissions. In these settings, some user-permission assignments may be known allow-deny decisions, while user-permission

pairs remain unknown. Here, the noise removal phase must perform prediction of the unknown values to identify a complete binary relation to input for role mining.

4.3.2 Using the Decomposition

The two meanings of noise above imply the user-permission relation can be factored into three disjoint relations: assignments for roles, assignments that are exceptions, and assignments that are errors.

If we begin with roles, intuitively these are the dominant patterns in the data and are strongly reinforced, i.e., there are many users that have identical or similar permission assignment patterns. Users share a common set of job duties within an organization and the reinforced patterns imply roles. This is the underlying assumption for all bottom up role engineering; if it does not hold, role mining cannot be performed.

Exceptions, on the other hand, are user-permission assignments that are slightly abnormal. Perhaps only *some* of the users sharing a common permission set have these extra assignments, or they have a subset of the additional permissions that similar users have. These assignments are weakly reinforced in the data such that their patterns are weaker than the user-permission patterns of roles.

Finally, non-systematic errors are likely to be rarely reinforced assignments. If an error is a strong pattern in the user-permission relation, for example roles that are incorrectly assigned or revoked, then we will not be able to identify such errors using just the user-permission relation.

The above analysis implies that the three sets of permission assignments can be differentiated by the strength of the patterns in the user-permission relation. The stronger the pattern, the more likely the user-permission assignment is correct. Low rank matrix decompositions are well known techniques for identifying dominant patterns where the rank is inversely proportional to the granularity of the patterns. They

have frequently been applied in other domains for compression and noise removal [68], and as we will show, can be successfully applied to the domain of role mining.

To identify noise in the original matrix X, we can compare X with AB^T. Where they differ indicates potential noise. However AB^T is not a binary matrix. There are several strategies that can be used to convert AB^T into a binary relation. First, we are not interested in interpreting the results of the decomposition, so we only need to make AB^T binary, and not A or B^T. Let $g : \mathbb{R}^{n \times m} \rightarrow \rangle \{ 0, 1 \}^{n \times m}$ be a function that converts a real-valued matrix into a binary matrix. A simple example of a function g is to set all user-permission relations greater than a threshold t, for example 0.5, to 1, and 0 otherwise. This is the approach we take in our experiments. As we increase t there will be fewer user-permission assignments in $\hat{X}$, resulting in a more conservative approximation (fewer assignments), and a sparser user-permission relation. Because of the sensitive nature of the access control matrix, we suggest several alternatives for g that integrate permission sensitivities next. In Section 4.6.3 a method that leverages risk and trust is presented.

Sensitivity of Users and Permissions

Security is a critical concern when attempting to detect noise in an access control system. One of the advantages of the two-step method we propose is that an administrator can inspect, and possibly reject, the noise correction. One shortcoming of all existing works on handling noisy data in role mining is that all permissions, and over- and under-assignments, are treated identically. There are several ways we can treat users and permissions uniquely.

In our experiments we use a single threshold t for all items. Increasing t will result in a more conservative user-permission relation, while decreasing t will increase permission assignments. An alternative is to select a threshold t_p for each permission. This allows an administrator to increase the threshold for more security sensitive permissions, or lower the threshold for permissions that are less sensitive, or are

needed for availability. Similarly we can use a unique threshold t_u for each user based on their trust.

If such measures are not available, we can estimate them using permission frequency. For permissions, we assume more sensitive permissions are assigned infrequently. We can define the sensitivity of permission j as the probability a user is not assigned it, $S(j) = 1 - \Pr[(u, p_j) \in X]$, where $\Pr[(u, p_j) \in X]$ is the fraction of users assigned p_j. This weights rare permissions more heavily than frequent permissions, but has low variance. Because of the sparsity of X, most permissions have sensitivity close to 1.

To resolve this issue, we must normalize the results into a more usable range $[0 \leq t_0, t_1 \leq 1]$ and use a unique threshold $t_j = S(j)$ for each permission j. When $t_0 = t_1 = 0.5$ we have the naïve solution. To be more lenient on assigning common permissions, we lower t_0, and to be more conservative for rare permissions, we increase t_1.

The objective is to estimate a prior probability the user needs the permission that will be compared to the posterior probability calculated by the matrix decomposition. We then assume a user who does not require a permission has a greater likelihood of misuse. This was called a *temptation* in [69].

As an alternative to selecting distinct thresholds for discretizing the real-valued matrix, we can leverage this information in our matrix factorization model. Using the unified view on matrix factorization [64], a weight matrix W can be applied to the distance measure, $D(W, X \parallel \hat{X})$. By applying a greater weight to more sensitive permissions one can ensure they are more accurately reconstructed. For example, the loss function for SVD can be defined as $\parallel W \odot (X - \hat{X}) \parallel_F^2$, where $\odot$ is element-wise matrix multiplication.

Determining the Correctness Noise

Identifying and removing noise is likely to be a human intensive task, even with tools provided to a role engineer. Assigning new permissions to a user or revoking permissions from a user is not taken lightly; the administrator must ensure policies are maintained and balance the requirements of compliance and separation of duty policies, against job requirements. There are many ways an administrator could proceed using a matrix decomposition to identify noise and select an appropriate rank k that will determine the separation between the noise and signal.

First, if the administrator has an estimate on the number of errors ϵ, we can select the smallest k such that $||X - g(X_k)||_1 \leq \epsilon$. Several prior works [7, 13] assume the number of errors is known a priori.

Second, an administrator can begin with a high-rank, fully reconstructed dataset, and slowly reduce the rank of the approximation, observing the changes to the user-permission relation. The administrator continues to reduce the rank of the decomposition until they are satisfied with the clean representation. While this is still a very human intensive task, it significantly reduces the work from $O(|U| * |P|)$.

An alternative approach is based on the decrease in a distance function $D(\ \|\rangle)$ observed from increasing the rank of $\hat{X}$, equivalent to adding a new pseudo-role to the RBAC state, and thus may be comfortable for system administrators. Roles that represent correct assignments are often assigned to a wide number of users. A role should represent a significant user-permission assignment pattern in an organization. Errors, on the other hand, should be rare; pseudo-roles that capture noisy assignments will cut small holes in the existing user-permission relation or add a small number of assignments.

Our approach is to slowly increase the rank of the decomposition and measure the number of changes made to the user-permission relation between the rank k and rank $k+1$ approximations, that is $\left|\left|g(\hat{X}_k) - g(\hat{X}_{k+1})\right|\right|_1$. When these gains drop below

a given threshold, we believe adding more roles recovers more noise than accurate user-permission assignments, and we can stop.

Low rank approximations are largely independent of small amounts of noise. We can correlate the results from many variations of the noisy dataset (e.g., employing the approach from [7]), and obtain a rank k that should work well for a given dataset. For example, see Figure 4.2. The top figure plots the L1 norm of the difference of the rank-k approximation and the clean data; the vertical lines illustrate the optimal rank decomposition for each noisy dataset. The second figure plots the normalized differences between the rank k and rank $k+1$ approximation; the vertical lines indicates when $||g(X_k) - \rangle g(X_{k+1})||_1 \,/\, ||X||_1$ drops below 0.1%.

Finally, a common approach in machine learning to select the rank k of a decomposition is to evaluate how well the decomposition recovers known values. A small amount of the data can be held out, and assumed unknown. The rank k that provides the best approximation of the missing values indicates the appropriate rank of the decomposition. We evaluate the predictive nature of several matrix decomposition methods against real datasets in Section 5.5.

Determining the Most Applicable Assignments

Most research in role mining has been concerned with migrating an *entire* user-permission relation to RBAC, typically trying to minimize some complexity measure. In practice, this is rarely required, and often not desirable. In Section 4.3.1 we noted that professional role engineers often speak of an 80–20 rule for role engineering. The intention is that in practice only 80% of the user-permission relation needs to be covered by roles.

Because only 80% of the user-permission relation needs to be represented by roles, we can be rather lax with our reconstruction, and propose several possible solutions. When using SVD, a common practice is to discard all singular values that are less

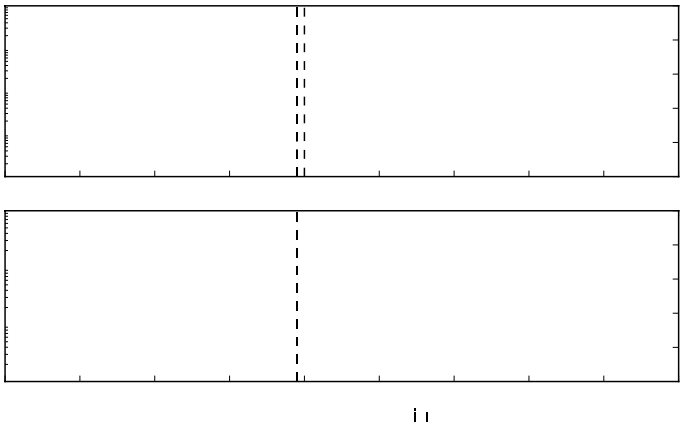


Figure 4.2. Observing the consistency of the optimal rank k decomposition using multiple levels of noise.

than 1.0, or to keep the singular values that represent most of the variance in the *UP* relation, e.g., the first 80%:

$$\frac{1}{\sum_{i=1}^{\ell} s_i^2} \sum_{i=0}^{k} s_i^2 \geq \alpha.$$

We will evaluate these approaches in the next section.

In general, we can use the 80–20 principle to select the significant 80% (or more) of the data using an appropriate rank decomposition. Simply stated, if we only require 80% of the data to be covered via roles, we can select k such that

$$\frac{\left|\left|X - g(\hat{X}_k)\right|\right|_1}{||X||_1} \quad 80\%.$$

Because we are selecting a more coarse approximation, it may be desirable to select a function g that favors under-assignment, for example by setting a threshold $t > 0.5$.

4.3.3 Performing Prediction

Prediction of unknown values can be performed using the matrix decomposition methods described in this chapter. The described decomposition methods require a complete real-valued or binary matrix, and do not allow for null or unknown values. Before these decomposition methods can be used, the null values must be replaced.

Two common approaches are to replace all unknown values with zero or the column means. To prevent biasing the prediction, the decomposition should be invariant to the replacement chosen. That is, the unknown values cannot impact the decomposition's fitness or distance measure. Given a matrix X with missing values, let $X'^{\propto}$ and $X''^{\propto}$ be the matrix X with unknown values replaced using two different methods. The distance measure is invariant to the unknown values if $D(X'^{\propto} \| \hat{X}) \equiv D(X''^{\propto} \| \hat{X})$. This can be accomplished by masking the unknown values such that they do not contribute to the distance, i.e., $D(W, X \| \hat{X})$. This method was described in Section 4.3.2.

4.4 Mining Noisy Data

The goal of role mining has always been to discover meaningful and useful roles. Next we consider how well a role mining algorithm can perform this task in the presence of noise. There are three basic solutions to the problem: clean the data first; use a role mining algorithm that can identify noise; or mine the noisy data and only return the top roles. We mine a noisy data before and after cleaning, and compare the results to the candidate roles obtained from mining the clean data.

A good solution will produce roles that are similar to the roles mined from clean data regardless of the presence of noise. We introduce a distance measure between two sets of roles and use it to evaluate how well two role mining algorithms reproduce meaningful roles when mining both noisy and cleaned data. One result obtained from this analysis is that the role mining objective can impact the susceptibility of the candidate roles to overfit noisy user-assignments. In particular RMP, minimizing the number of roles required, is especially susceptible to noise.

4.4.1 Role Quality

System administrators are often skeptical of candidate roles generated via role mining, believing the roles are overfitting to the data. Some candidate roles may

only be generated when the input data contains noise, making deployment and administration difficult. The goal of a role mining algorithm should be to produce a similar set of candidate roles regardless of the presence of noise. This is similar to the intuitive definition of role stability from Colantonio et al. [70]. Roles that overfit a noisy dataset will be different from roles mined from a clean dataset. The greater the difference, the more unstable and overfit the roles are.

To mine stable roles, Colantonio et al. [70] define the notion of role weight, and present a method to mine roles satisfying a minimum weight. The weight of a role in a function of the number of users and permissions assigned to the role, such as a weighted sum or product. Unfortunately, role stability is not formally defined, and role weight does not satisfy the intuitive definition of a stable role. A proof is presented in Appendix A.3.

A stable role is one that is invariant to small changes in the input data, and an RBAC state that can generalize to an independent set of users drawn from the same distribution. In the context of role mining, this independent set of users should come from the same organization. A role mining result isn't required to generalize to other organizations. We now define role stability.

Definition 4.4.1 (Role Stability) Let $\rho_0 = \langle U_0, P_0, UP_0 \rangle$ and $\rho_1 = \langle U_1, P_1, UP_1 \rangle$ be two access control configurations where $U_0 \cap U_1 = \emptyset$ and ρ_0 and ρ_1 are drawn from the same distribution. Next, let R_0 and R_1 be the candidate roles output from ρ_0 and ρ_1 by role mining algorithm $\mathcal{A}$ respectively. The role mining algorithm $\mathcal{A}$ is stable if R_0 and R_1 are similar, and the distance $D(R_0 \| R_1)$ is small.

In the presence of noise ρ_1 is a noisy version of ρ_0, and the restriction $U_0 \cap U_1 = \emptyset$ is relaxed. Here stability is defined as the invariance to noise.

To measure how overfit or stable a set of roles is, we first need a distance measure $D(\gamma\sigma \| \beta)$ between two RBAC states $\gamma\sigma$ and β. We start with a distance measure between two roles and expand it into a distance measure between two sets of roles.

4.5 A Distance Metric for RBAC

Before we can measure the stability of an RBAC state, the impact noise has on a set of candidate roles, or compare the roles resulting from two different role mining algorithms or weight vectors W and W', we need a distance metric between the set of candidate roles. First, we need to define a distance metric.

Definition 4.5.1 (Metric) A distance metric must have the following properties:

1. $d(x, y) \geq 0$ [Non-negativity]
2. $d(x, y) = 0 \Leftrightarrow x = y$ [Identity of Indiscernible]
3. $d(x, y) = d(y, x)$ [Symmetry]
4. $d(x, z) \leq d(x, y) + d(y, z)$ [Triangle Inequality]

We will start with a distance measure between two roles, and expand it into a distance measure between two sets of roles that represent RBAC states.

4.5.1 Distance Measure for Roles

In order to define a distance metric between two roles, we must determine the domain of a role. Roles are defined by the RBAC user-assignment (UA), permission-assignment (PA), and role-hierarchy (RH) relations. In the context of role mining, it is useful to think of a role as a set of authorized users and permissions, and the role-hierarchy as reducing the absolute size of UA and PA. In many organizations the job functions a user performs change, while the permissions required for a job are relatively stable. This is one of the main motivating factors in using RBAC. As a result, we define a role by its set of authorized permissions, resulting in the following definition for a role distance metric,

$$d_R : 2^P \times 2^P \rightarrow \mathbb{R}. \tag{4.1}$$

We can then use the popular Hamming distance as one possible distance measure

$$\mathsf{Hamming}(X, Y) = |\, X \cup\rangle Y \,| - \rangle|\, X \cap\rangle Y \,|\,. \tag{4.2}$$

One disadvantage of the Hamming distance is that it measures the total number of differences between two roles, and not the relative number of differences. For example, a Hamming distance of one represents a higher degree of dissimilarity when $|\, X \,|$ is small (e.g., two or three permissions), than when $|\, X \,|$ is large (e.g., hundreds of permissions).

A common measure of similarity between two roles that addresses this issue is the Jaccard similarity coefficient

$$\mathsf{J}(X, Y) = \frac{|\, X \cap\rangle Y \,|}{|\, X \cup\rangle Y \,|}, \tag{4.3}$$

and the related Jaccard distance, $\mathsf{J}_\delta(X, Y) = 1 - \mathsf{J}(X, Y)$, which satisfies the required properties of a metric [71]. Note that the Jaccard distance is the normalized Hamming distance, and is bound by zero and one inclusive. The Jaccard similarity measure has been extended to a similarity measure between two sets of roles [17] using the average maximum Jaccard,

$$\mathsf{AMJ}(X, Y) = \operatorname*{avg}_{x \in X} \max_{y \in Y} \mathsf{J}(x, y), \tag{4.4}$$

where X and Y are sets of roles and x and y are sets of permissions. This measure of role set similarity was proposed by Vaidya et al. [17].

The average maximum Jaccard can easily be converted into a measure of dissimilarity, $1 - \rangle\mathsf{AMJ}(X, Y)$. Equivalently we can define the average minimal Jaccard dissimilarity measure, $\mathsf{AMJ}_\delta(X, Y) = \operatorname{avg}_{x \in X} \min_{y \in Y} \mathsf{J}_\delta(x, y)$.

There are several issues with the average maximum Jaccard. First, it is not a metric because it does not satisfy the symmetric property: $\mathsf{AMJ}(X, Y) \neq \mathsf{AMJ}(Y, X)$. This can clearly be seen using the following example. Consider two candidate role sets, X and $Y = \wp(P)$, the powerset of the set of permissions, where $X \neq Y$. It is clear that $AMJ(X, Y) = 1$, while $AMJ(Y, X) < 1$. Second, excessive role creation can artificially improve the measure, as shown above. A more realistic example is

the following. Consider a role $R = \{ p_j, \cdots, p_n \}$ taken from a set of permission $P = \{ p_0, \cdots, p_m \}$. Many roles "close" to R can be generated as $R_i = R \setminus \{ i \}$, or $R_\ell = R \cup \{ p_\ell \}$ where $p_\ell \in P \wedge p_\ell \notin R$.

4.5.2 Role Matching Distance

We can view the average minimum Jaccard distance, $\mathsf{AMJ}_\delta(X, Y)$, as finding a mapping from the set of roles X to the set of roles Y. The main flaw is that it is not an injective mapping; each role in Y may be the closest match for multiple roles in X. We resolve this by requiring the closest pairing of roles between two sets where each role can only appear in only one pair.

Given two RBAC states γ and β, let R_γ be the roles in γ and R_β be the roles in β. The pairing is an injective mapping $m : R_\gamma \rightarrow R_\beta$. We force the mapping to be a bijection by requiring $| R_\gamma | = | R_\beta |$. Without loss of generality (the distance measure must be symmetric), assume $| R_\beta | \leq | R_\gamma |$. We add $| R_\gamma | - | R_\beta |$ roles to R_β whose permission set is the empty set. Finally, denote the set of all bijections from R_γ to R_β as $\mathcal{M}(R_\gamma, R_\beta)$, and let d be a distance metric between two roles, as defined by their permission sets (Definition 4.5.1 and Equation 4.1). The distance between γ and β is

$$D_d(\gamma \| \beta) = \min_{m \in \mathcal{M}(R_\gamma, R_\beta)} \sum_{r_i \in R_\gamma} d(r_i, m(r_i)). \tag{4.5}$$

This problem is known as minimum weighted bipartite matching (MWBM). The minimum weighted bipartite matching problem is defined as follows:

Definition 4.5.2 (Minimum Weighted Bipartite Matching) Given a bipartite graph $G = \langle V = (X, Y), E \rangle$ where $X \cap Y = \emptyset$ and E is a set of undirected edges from vertices in X to vertices in Y, and a weight function over the set of edges, $w : E \rightarrow \mathbb{R}$, find a set of pairwise adjacent edges $M \subseteq E$ such that the sum of the weights of the edges is minimal, i.e., $\arg\min_{M \subseteq E} \sum_{e \in M} w(e)$.

The minimum weighted bipartite matching problem can be solved in $O(V^3)$ using the Hungarian algorithm [72]. A $O(V^2 \log V)$ algorithm that computes the optimal

solution with probability $1 - o(n^{-1})$ also exists [73]. In practice the number of roles is expected to be small because they are typically managed by human administrators.

We convert the RBAC distance problem into an instance of the minimum weighted bipartite matching where we desire a perfect matching, $|M| = \min(|X|, |Y|)$, as follows. Given two RBAC states γ and β, we create one vertex in X for each role in γ and one vertex in Y for each role in β. For each pair of roles (x, y), $x \in X$ and $y \in Y$, add an edge (x, y) with weight $d(x, y)$. The solution to the minimum weighted bipartite matching is the distance between the two RBAC states, which we denote $D_d(X \| Y)$.

4.5.3 Minimum Matching is a Distance Metric

We now prove the minimum weighted bipartite matching is a metric if the edge weights are measured by a distance metric.

Theorem 4.5.1 $D_d(X \| Y)$, as defined in Equation 4.5, defines a metric when d satisfies the properties of a metric.

Proof $D_d(X \| Y)$ satisfies each of the four properties of a metric in Definition 4.5.1.

- $D_d(X \| Y)$ *satisfies non-negativity, i.e.,* $D_d(X \| Y) \geq 0$. The distance measure is the sum of individual distance measures defined by edge weights, which are strictly non-negative. Therefore, the distance measure must be non-negative.

- $D_d(X \| Y)$ *satisfies* $D_d(X \| Y) = 0 \Leftrightarrow X = Y$. $D_d(X \| Y)$ is the sum of the distances of its parts. If $D_d(X \| Y) = 0$, then by definition $\forall_{x \in X} \exists_{y \in Y} d(x, y) = 0$, and therefore $\forall_{x \in X} x \in Y$ by the metric property of d. Finally, because $|X| = |Y|$, we know that $\forall_{y \in Y} y \in X$, and therefore $X = Y$. The reverse direction follows directly from the definition of $D_d(X \| Y)$ and the metric property of d.

- $D_d(X \| Y)$ *is symmetric, i.e.,* $D_d(X \| Y) = D_d(Y \| X)$. Each edge in the bipartite graph G is undirected, and each edge weight is symmetric by defini-

tion, thus $G = \langle V = (X, Y), E \rangle \equiv \langle V = (Y, X), E \rangle$, and the minimum weighted matching must be identical.

- $D_d(X \parallel Y)$, *satisfies the triangle inequality, i.e.,* $D_d(X \parallel Z) \leq D_d(X \parallel Y) + D_d(Y \parallel Z)$. Let M_{XZ}, M_{XY}, and M_{YZ} be the matchings for $D_d(X \parallel Z)$, $D_d(X \parallel Y)$, and $D_d(Y \parallel Z)$ respectively. Then we can rewrite the above as

$$\begin{aligned} \sum_{(x,z)\in M_{XZ}} \sum d(x,z) &\leq \sum_{(x,y)\in M_{XY}} \sum d(x,y) + \sum_{(y,z)\in M_{YZ}} \sum d(y,z) \\ \sum_{(x,z)\in M_{XZ}} \sum d(x,z) &\leq \sum_{y\in Y} d(x,y) + d(y,z); \quad (x,y) \in M_{XY},\ (y,z) \in M_{YZ} \end{aligned}$$

  Because the mappings $M_{*\infty}$ are bijections, (x, z) is uniquely determined by each $y \in Y$. Because the triangle inequality holds for d for each unique x, y, z, and $D_d(X \parallel Y)$ is the sum of its parts, the triangle property holds.

Finally, the proof of Theorem 4.5.1 follows directly from the above four required properties. Therefore the Theorem holds, and the minimum weighted bipartite matching defines a distance metric for RBAC roles when the edge weights are defined by a distance metric. ■

4.6 Leveraging Attributes

We have shown how to leverage the structure of the user-permission relation to identify errors and perform predictive analysis on several real-world datasets. We now illustrate how attributes can be leveraged to improve predictive accuracy using a real dataset from a large organization.

Attributes have been used to increase the semantic meaning of roles [12, 14], but not for noise detection. Colantonio [12] use an organization hierarchy to define roles assigned to closely related users. Frank et al. [14] select roles with high attribute compliance[1] from otherwise similar roles generated with MAC [15] by maximizing the number of attributes shared by users assigned to a role. Their approach reduces

[1]Number of attributes shared by users assigned to a role.

permission assignment accuracy to increase attribute compliance, increasing the number of errors in order to gain semantic meaning.

Frank et al. also provide a method to identify attributes that are applicable to role mining. They suggest a measure based on the entropy reduction $I(p_j, A)$ of a permission p_j's assignments given knowledge of a user's attribute A:

$$I(p_j, A) = \frac{h(p_j) - h(p_j \mid A)}{h(p_j)}$$

where $h(p_j)$ is the entropy of the permission p_j,

$$h(p_j) = -\sum_{s \in \{\,0,1\,\}} \Pr\left[\, p_j = s \,\right] \log_2 \Pr\left[\, p_j = s \,\right]$$

and $h(p_j \mid A)$ is the entropy of p_j conditional on the value of the attribute A

$$h(p_j \mid A) = -\sum_{a \in A} \Pr\left[\, A = a \,\right] \sum_{s \in \{\,0,1\,\}} \Pr\left[\, p_j = s \mid A = a \,\right] * \log_2 \Pr\left[\, p_j = s \mid A = s \,\right].$$

In the rest of this section we identify a shortcoming of the above entropy-based approach on attribute selection. We then use the collective matrix factorization framework from Section 4.2.1 to illustrate how attributes can be used to improve the accuracy of a decomposition.

4.6.1 Analysis of the Organization Dataset

We analyze how well attributes can be used on a real dataset from a large anonymous organization, which will be referred to as the **anonymous** dataset. The dataset contains over three thousand users and permissions, and around seventy-two thousand permission assignments, making it exceptionally sparse compared to the datasets used elsewhere [13, 19]. Sparsity, combined with a long tail distribution, make the data difficult to predict held out values. The dataset also contains eight attributes per user.

We have discovered that the granularity of each attribute, the number of values it may take, often has a significant impact on the entropy reduction. The conditional

entropy above divides the user-permission relation into k disjoint sets of users. If k is equal to the number of users, then each user is in their own user-permission relation, and the total conditional entropy is zero. This granularity bias causes the entropy reduction metric to favor finer grained attributes.

Table 4.1 sums the total conditional entropy over all permissions for each attribute, $\sum_{p_j \in P} h(p_j \mid A)$, and the total entropy of all permissions, $\sum_{p_j \in P} h(p_j)$. It should not be concluded that a user's *Last Name* is significant in predicting their permissions given it has the least total entropy, nor should we consider the *Contractor* attribute useless. We do believe that the *Department* attribute may be more significant than *Title* because it produces a greater reduction in the entropy and has a smaller cardinality. We propose a partial order over the attributes by $\langle$*Cardinality*, *TotalEntropy* where we wish to minimize both cardinality and total entropy. The partial order is shown in Figure 4.3.

Table 4.1
The total uncertainty reduction of the user-permission relation given knowledge of a user's attribute. Total entropy of the user-permission relation is 107.34 bits.

Attribute	Cardinality	$\sum_{p \in P} I(p, A)$
Last Name	2224	2769.39
Manager	298	2186.03
Department	192	1931.95
Title	527	1878.51
Location	53	1316.92
Organization	12	789.46
Level	17	170.34
Contractor	2	78.44

Figure 4.3. Partial order over attributes significance.

4.6.2 Using the Collective Matrix Factorization

The most applicable user-attributes are selected using the cardinality and total entropy minimization partial order. Categorical attributes are converted into binary relations UAA_A where $\forall_{u \in U} \mid \{\, a \mid (u, a) \in \rangle UAA_A \,\} \mid = 1$. Multiple user-attribute assignment relations may be combined into larger relations such that a user is assigned only one attribute for each category.

The same techniques may be applied to permission attributes when applicable. For instance, attributes indicating the application a permission applies to, such as "DB2", "PeopleSoft", "ActiveDirectory", etc., may be available. This results in a permission-attribute relation PAA.

The three binary relations, UP, UAA, and PAA, are collectively factored such that

$$\begin{array}{cc} UP & AB^T \\ UAA & CA^T \\ PAA & BD^T \end{array}$$

and minimizing the weighted loss

$$\alpha D(UAA \,\|\rangle CA^T) + \beta D(PAA \,\|\rangle BD^T) + (1 - \rangle\alpha - \rangle\beta) D(UP \,\|\rangle AB^T).$$

The parameters $\alpha\sigma$and $\beta\sigma$are chosen that minimize the predictive error on the UP relation. Evaluation for the **anonymous** dataset is given in Section 5.5.1.

4.6.3 Collective Matrix Factorization of Risk and Trust

Section 4.3.2 illustrated how to individually weight users and permissions when performing decomposition to integrate sensitivity semantics. Collective matrix factorization provides more powerful tools by allowing actuarial data, attributes about users and permissions, to be integrated in the factorization directly.

The permission attribute relation PAA may be a binary relation (for example, permission p_1 grants access to application a_1). However, it may represent actuarial data on the impact from misuse of permissions across multiple vectors such as monetary loss (e.g., fines, lawsuits, stock value), loss of reputation (e.g., expected decrease in future sales), loss in human life, etc. The loss function for PAA can also assume a non-Bernoulli distribution, such a normal or lognormal, and it does not need to be the same loss function as the user-permission relation. For example, a logistic function can be used for user-permissions, and the Forbenius for expected harm.

4.7 Conclusions

The majority of role mining algorithms assume the provided input data is clean and correct. In reality, however, in any large organization this data is likely to contain noise.

We advocate a two-step process to role mining with noisy data: first clean the data, then mine. We propose non-binary matrix decomposition as a solution to cleaning the user-permission relation. Our experiments indicate that singular value decomposition, non-negative matrix factorization, and logistic PCA perform well, while several other techniques, including two state-of-the-art probabilistic algorithms, yield high false negative rates. The ability of the top matrix decomposition models to predict missing values is evaluated on real datasets and we illustrate how attributes can be leveraged to improve predictive performance.

To evaluate the impact noise has on the quality of the candidate roles we introduce a new distance measure between two role sets; this measure avoids the pitfalls of the

average maximum Jaccard that has been used previously. Using our distance measure, we find that first cleaning the noisy data produces candidate roles that more closely resemble the candidate roles mined from the clean data than alternatives such as selecting the top roles or allowing exceptions. Finally, we found that trying to find roles to exactly cover the data can reduce the quality of the resulting candidate roles, causing them to overfit the noisy data.

5 EXPERIMENTAL EVALUATION

This chapter presents experimental analysis illustrating the effectiveness of the role mining techniques proposed in this dissertation, and compares the results to role mining algorithms from the literature. Because of the diversity of role mining algorithms from the literature, algorithms are categorized into two classes based on their output; Class 1 algorithms output a sequence of prioritized roles while Class 2 algorithms output complete RBAC states. Methods to convert Class 1 output into Class 2 and vice versa are discussed. A subclass of Class 2 algorithms that produces an RBAC state that is inconsistent with the input user-permission relation is considered.

Using synthetic as well as real datasets, eight role mining algorithms are compared. Two new algorithms for generating user-permission data with role hierarchies are also presented: the tree-based data generator and the ERBAC data generator. The tree-based data generator outputs a tree-based role hierarchy while ERBAC data generator outputs a two-level role hierarchy in the Enterprise RBAC model. Both data generators are parameterized.

Performance of role mining algorithms is evaluated in several ways. First, the RBAC states generated by each role mining algorithm at satisfying a given role mining objective, as defined by the weighted structural complexity, are directly compared. Next, several new role quality measures are described in this chapter to compare the RBAC states generated from candidate roles mined from algorithms of different classes. These measures, which may discard candidate roles, are used. This allows one to compare a Class 1 and Class 2 role mining algorithm. Third, the noise removal techniques proposed in this dissertation are evaluated against three techniques proposed in the literature. Finally, the ability of rank reduced matrix decomposition to predict unknown access control decisions is evaluated on three real-world datasets. The improvement in predictive performance gained from attributes using the collec-

tive matrix factorization framework is measured. Prediction is useful for granting exceptional access or mining roles from incomplete data, such as access history logs.

The next section describes a role mining algorithm classification system and algorithms for converting between algorithms of one class to another. Section 5.1 presents additional RBAC role quality measures used for evaluation, and describes the real, synthetic, and algorithmically generated datasets used for evaluation. Evaluation results are then presented: Class 2 algorithms and Class 1 algorithms converted to Class 2 output are directly compared in Section 5.3; and noise removal and prediction methods are evaluated in Section 5.5.

5.1 Classification of Role Mining Algorithms

Existing role mining algorithms in the literature can be divided into two main classes based on their output. This section presents a classification system for role mining algorithms, and presents methods for converting between the two main classes.

5.1.1 Class 1 Algorithms: Outputting prioritized roles

Algorithms in the first class output a prioritized list of candidate roles, each of which is a set of permissions. These algorithms output a vector of candidate roles $\vec{\mathbf{C}}$ ordered by their priority. Examples include *CompleteMiner* and *FastMiner* in [5].

These algorithms can be divided into two phases: *candidate role generation* and *candidate role prioritization*. The candidate role generation phase identifies a set of candidate roles from the user-permission assignment data. This phase usually outputs a large number of candidate roles. The candidate role prioritization phase assigns a priority value to each candidate role; roles with a larger priority value are believed to be important and useful by the role mining algorithm.

5.1.2 Class 2 Algorithms: Outputting RBAC states

Algorithms in the second class output a complete RBAC state. Examples include ORCA [4], graph optimization [8], role/edge minimization algorithms [19], and *HierarchicalMiner* [16].

These algorithms take as input a configuration $\rho\sigma = \langle U, P, UP$ and output an *RBAC state* $\gamma\sigma = \langle R, UA, PA, RH, DUPA$ that is consistent with ρ. In the state, R is a set of roles, $UA \quad U \times\rangle R$ is the user-role assignment relation, $PA \quad R \times\rangle P$ is the role-permission assignment relation, $RH \quad R \times\rangle R$ is a partial order over R, which is called a *role hierarchy*, and $DUPA \quad U \times\rangle P$ is the direct user-permission assignment relation. The RBAC state is *consistent* with $\langle U, P, UP$, if every user in U has the same set of authorized permissions in the RBAC state as in UP.

Some of these algorithms [6–8, 10, 16, 19] aim at generating an RBAC state that minimizes some cost measure, such as minimizing the number of roles or the number of user-assignments and permission-assignments. Weighted structural complexity is a general parameterized optimization objective that subsumes the objectives used in the above algorithms. Different weight vectors encode different mining objectives and minimization goals. For example, by setting $W = \langle 1, 0, 0, 0, \infty \rangle$ one can minimize the number of roles. *HierarchicalMiner* [16] takes both a configuration $\rho\sigma$and a weight vector $\langle w_r, w_u, w_p, w_h, w_d$ and aims at outputting an RBAC state with low WSC. Other algorithms, such as graph optimization [8], are often designed to minimize a specific metric, such as the number of edges. These algorithms can be generalized to take a weight vector as input. In the experiments in this dissertation, all algorithms are modified when appropriate to minimize a generalized WSC weight W.

5.1.3 Class 1 Versus Class 2 Algorithms

RBAC states are easy to compare against a common weight vector W, however it is important to compare role sequences too. Outputting a list of candidate roles can be more useful in practice. A typical role engineering process is not completely

automatic, because the input data is often noisy or incomplete. It is unlikely that one will adopt a complete RBAC state output by a role mining program. The most likely usage scenario of a role mining tool is as follows: the administrator examines the role mining results and determine whether to adopt some part of it. Outputting a sequence of candidate roles will allow the administrator to examine the sequence of candidate roles one by one and determine whether these roles should be created. In short, this dissertation argues that an important metric that is relevant in practice for role mining algorithms is whether they can suggest the best candidate roles, and suggest these first.

5.1.4 Class 3 Algorithms: Outputting an Inconsistent Covering

Class 2 algorithms output a complete RBAC state that is consistent with the input user-permission relation. If the input user-permission relation contains errors (incorrect assignments described in Section 4.3.1), or missing or unknown values (as described in Section 4.3.3), a consistent state may be undesirable or insecure. Some role mining algorithms [7, 13–15], and even *HierarchicalMiner* and *AttributeMiner*[1], produce complete RBAC states that are an inexact covering. Class 3 algorithms are a special case of Class 2 role mining algorithms, providing some users with a different set of authorized permissions. A Class 1 algorithm cannot be consistent with ρd itself; a user-assignment relation is needed to authorize roles to users resulting in over- and under-assign permissions.

5.1.5 Converting Class 1 Algorithms to Class 2 Algorithms

A method to convert outputs of one class into the other is required to compare Class 1 algorithms and Class 2 algorithms. Class 3 algorithms will be handled separately in Section 5.5.

[1]If $w_d \neq \infty$⟩and the direct user-permission assignments are discarded.

Given a sequence of candidate roles $\vec{\mathbf{C}}$ generated by a Class 1 algorithm and a weight vector W, these roles are used to construct an RBAC state that minimizes the WSC. For each k from 1 to some suitable upperbound (with a maximum of $\left|\vec{\mathbf{C}}\right|$), consider the optimal RBAC state using the top k roles in the sequence, denoted $\gamma^{*\infty}_{[\rho,\vec{\mathbf{C}},k,W]}$, and chooses the best among them.

When the exact set of roles R created is known, the optimal RBAC state can be obtained by finding the optimal way to cover each user and each role with permissions and the created roles. Each user and each role can be considered separately from how other users and roles are covered. Note that the resulting RBAC state will often be hierarchical.

For each user and each role, this creates an instance of the set cover problem: given a permission set P_i and a family $\mathcal{R}$ of roles that are subsets of P_i, a cover is a subfamily R_i $\mathcal{R}$ whose union is P_i; the set cover optimizing problem is to find a cover which uses the fewest roles. An optimal set cover algorithm is to consider all subfamilies of $\mathcal{R}$, check whether they cover P_i, and select the subfamily that uses the fewest sets.

When constructing an RBAC state, users and roles are covered with roles. Covering a user with a role R_i adds a user-assignment (UA), at a cost of w_u, and covering a role with R_i adds a role hierarchy assignment (RH) at a cost of w_h. Individual items from P_i may also be assigned, adding to PA when P_i is a role, and $DUPA$ when P_i is a user. As the optimal set cover algorithm runs in exponential time, the experiments in this dissertation use a dynamic combination of the optimal algorithm and a heuristic algorithm. If the size of $\mathcal{R}$ is at most fifteen, apply the optimal algorithm to cover P_i; otherwise, use the following heuristic algorithm. At step j, the algorithm chooses $r_j \in \rangle\mathcal{R}$ such that r_j covers the most elements that haven't been covered in P_i yet with minimal cost (from W). This procedure terminates when all elements in P_i have been covered in at least one of the steps and R_i is the union of all the r_js. Because the same dynamic combination is used for all algorithms, the comparison of different algorithms remains fair.

Note that this is a quite expensive procedure. However, this is not part of any role mining algorithm, but rather a step used to compare different algorithms.

5.1.6 Converting Class 2 Algorithms to Class 1 Algorithms

Given an RBAC state, output a prioritized sequence of roles as follows. First, compute a score for each role. The score of a role r is the difference in the WSC of the mined RBAC state and the WSC when r is removed. Roles are ordered in decreasing order. The intuition is that if a role is important, removing the role would result in an RBAC system with a large structural complexity.

To calculate the score of a role r, compute the structural complexity of the resulting RBAC system when r is removed. When role r is removed from the role hierarchy, all of r's children are assigned to all of r's parents. There are some permissions in r that do not appear in any of its children. Add these permissions to all of role r's parents. Finally, cover all users of role r by applying an optimal set cover algorithm using the other roles and direct user-permission assignment and weights W. An RBAC system with role r removed has now been constructed, and the structural complexity difference is computed.

5.2 Methodology

5.2.1 Metrics for Comparing Algorithms

Role mining algorithms will be compared on both the complexity of the RBAC state, and on the quality of the roles output. Algorithms will not be evaluated on their efficiency or running time. However, the RBAM algorithm will not be evaluated due to its time complexity.

Quality of RBAC States

Using the WSC, one can evaluate how well each algorithm performs under a variety of mining objectives. For each algorithm and dataset, the data will be mined using a variety of weight vectors that tune the algorithms for each objective, e.g., role minimization or edge minimization. Algorithms that do not accept WSC inputs, such as *CompleteMiner*, are mined only once.

Weights Used

There is a countably infinite[2] number of possible WSC weights we could use to evaluate the role mining algorithms. A small representative set of weights that depict what is commonly used in the literature is selected.

1. $W = \langle 1, 0, 0, 0, \infty$. This is the basic-RMP objective. It will minimize the number of roles.

2. $W = \langle 0, 1, 1, \infty, \infty$. This is edge-RMP (edge-concentration). It will minimize the number of UA and PA relations.

3. $W = \langle 1, 1, 1, 1, \infty$. This is the Zhang-variant [8] of edge-RMP with a role hierarchy. It will minimize the number of roles and edges without direct assignments.

4. $W = \langle 1, 1, 1, 1, 1$. This will minimize the size of the RBAC state representation, and allows direct assignments.

5. $W = \langle 1, 1, 5, 1, 5$. Similar to the above, this will minimize the size of the relation while treating permission assignments (PA and $DUPA$) as more important.

When $w_d = \infty$, the described set cover solution will maximize the coverage of UP.

[2]The set of weights is countably infinite when $W \in \rangle (\mathbb{Q} \cup \infty)^5$, and uncountably infinite when $W \in (\mathbb{R} \cup \infty)^5$.

Prioritized Role Quality

As discussed in Section 5.1, outputting a prioritized sequence of roles can be more useful in practice, and one needs a way to compare the quality of this sequence generated by different role mining algorithms. This is not an easy task. While one can define the quality of a set of roles using some measures, the order in which the roles are output is important to consider. For example, when there are 100 roles, outputting the most useful roles early in the sequence is much better than outputting them last, as the administrators are likely to consider only the roles output earlier. To address this challenge, the following approach is used. Once the prioritized roles for each algorithm are obtained, compute the optimal RBAC state using the top k roles for $k = 1$ to some upper-bound commensurate with the size of the dataset (possibly exhausting the set of candidate roles). For each of the k RBAC states are then evaluated against a common criteria. The ability of the RBAC state to quickly (in few roles) optimize (minimize or maximize) the objective using is considered.

This analysis allows us to generate simple k vs. criteria plots, such as k vs. cost and k vs. coverage.

1. Among the top k roles, how quickly do the mined roles reduce the complexity of $\gamma\sigma($compared to $UP)$? For each weight vector W, we evaluate the complexity of the optimal RBAC state using only the top k roles.

2. Among the top k roles, how quickly do the mined roles cover the UP relation?

3. Among the top k roles, how well do they "resemble" the original roles? In [5], Vaidya et al. consider the number of original roles recovered as an evaluation criteria. One advantage of role-mining is to improve the RBAC state by finding better roles, a problem directly addressed in [17]. Vaidya et al. [17] use the Jaccard coefficient as a similarity metric between two roles r_1 and r_2

$$\mathsf{Jaccard}(r_1, r_2) = \frac{|\, r_1 \cap r_2 \,|}{|\, r_1 \cup r_2 \,|}$$

and define the similarity between two sets of roles R_1 and R_2 as the average maximum Jaccard between each role $r_i \in \rangle R_1$ and all roles $r_j \in \rangle R_2$

$$\mathsf{sim}(R_1, R_2) = \text{avg}_{r_i \in R_1} \max_{r_j \in R_2} \mathsf{Jaccard}(r_i, r_j). \tag{5.1}$$

Here, R_1 is the set of mined roles in $\gamma\sigma$and R_2 is the set of roles in the original data.

When original roles are available, such as in generated data, one can calculate the similarity between original and mined role sets. This is primarily useful for evaluation purposes only.

Measuring "how quickly," or "how well" the candidate roles satisfy the evaluation criteria has not been addressed. For example, consider the goal of reducing the WSC of $\gamma\sigma$and the task of comparing two sets of candidate roles A and B. Set A has only one role that greatly reduces the complexity while set B requires more roles but converges to a less costly solution. From an administrative point of view, set A may to easier to understand (fewer roles) while set B may be easier to manage (lower cost).

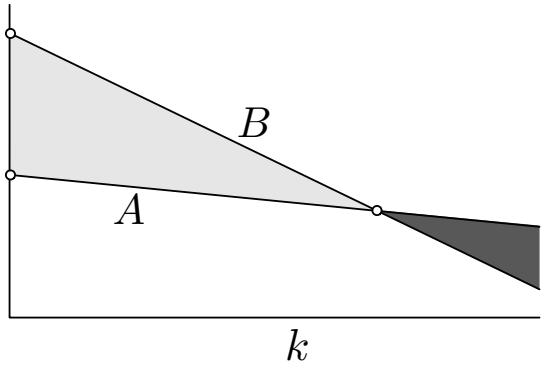


Figure 5.1. Illustrating the difference between quickly and optimally optimizing a role mining objective as the number of roles changes. Selecting the best algorithm depends on the area of the shaded regions.

To balance these two criteria (localized and global improvements), one can integrate the evaluation criteria over the number of roles, creating quality metrics. For example, the quality of the candidate roles $\vec{\mathbf{C}}$ at reducing the WSC of ρ is

$$\mathsf{Q}_{wsc}(\rho, \vec{\mathbf{C}}, k, W) = \int_0^k wsc(\gamma^{*\infty}_{[\rho,\vec{\mathbf{C}},x,W]}, W)\, dx. \tag{5.2}$$

Similar quality metrics for coverage and similarity can be defined. This strategy is useful for evaluating any utility measure that one wishes to minimize or maximize.

5.2.2 Input Data Type

All role mining algorithms evaluated use *user permission* information as the input data. That is, the input to a role mining algorithm is an access control configuration as defined in Definition 3.1.1. That is, an access control *configuration* ρ is given by a tuple $\langle U, P, UP$, where U is a set of all users, P is a set of all permissions, and $UP \quad U \times \rangle P$ is the user-permission relation. Exceptions include *AttributeMiner*, which also uses user attribute information such as a user's job title, department, and location. *AttributeMiner* will be evaluated separately for its ability to generate compact, and semantically meaningful roles.

Datasets from the Literature

Both real-world and newly generated data will be used to evaluate *HierarchicalMiner* and *AttributeMiner* and other role mining algorithms from the literature. Datasets that have been used previously are shown in Table 5.1.

The **university** data was generated based on a template[3] used in [74]. Researchers from Stony Brook University generated a template for an RBAC system in a university setting, presumably through a process similar to top-down role engineering. They created this template for the purpose of studying security analysis in role based access control, rather than role engineering. Thus, the main consideration was to make the

[3] `http://www.cs.sunysb.edu/~stoller/ccs2007/university-policy.txt`

Table 5.1
Sizes of the real-world datasets presented.

Dataset	$\mid Users \mid$	$\mid Perm \mid$	$\mid UP \mid$	Density
Americas	3477	1587	105205	0.019
Firewall 1	365	709	31951	0.123
Firewall 2	325	590	36428	0.190
APJ	2044	1146	6841	0.003
Domino	79	231	730	0.040
EMEA	35	3046	7220	0.068
Healthcare	46	46	1486	0.702
University	493	56	3955	0.143

RBAC system as realistic as possible. This template specifies roles, permissions, the role hierarchy, and the role permission assignment relation. A dataset is created using the template by creating users and assigning them to roles. The final dataset contains 493 users, and 56 permissions. Five attributes that are likely to be maintained in a typical university data system are created: *Undergrad*, *Grad*, *HonorsStudent*, *TA*, *Faculty*. To obtain a user-permission relation, the RBAC state is flattened, directly assigning users their full set of authorized permissions.

The other seven datasets were obtained from researchers at HP Labs and used for evaluation in [19]. The **healthcare** data was from the US Veteran's Administration; the **domino** data was from a Lotus Domino server; **americas** (referring to **americas_small** in [19]), **emea**, and **apj** data were from Cisco firewalls used to provide external users access to HP resources. We also use their **firewall1** and **firewall2** policies.

Generated Data

In addition to real-world and top-down datasets, synthetic datasets generated from three test data generators are used. The first data generator is the random data generator from Vaidya et al. [5]. Two other data generators, tree-based data generator and ERBAC data generator, that produce more realistic RBAC datasets, are proposed. Due to the difficulty in obtaining real-world data, especially containing complete RBAC states, synthetic data generation is still a useful tool for role-mining evaluation.

Random Data Generator The random data generator was used in [5]; it takes five parameters $\{n_u, n_r, n_p, m_r, m_p\}$ where n_u, n_r, n_p are the number of users, roles, and permissions, respectively, and m_r, m_p are the maximum number of roles a user can have and the maximum number of permissions a role can have. The algorithm consists of three steps. First, for each role, a random number of permissions up to m_p are chosen to form the role. Second, for each user, a random number of roles up to m_r are assigned to the user. Finally, for each user, the user-permission assignments are computed based on user-role assignments and role-permission assignments.

The data generated by the random data generator does not contain any structure and treats each user, role, and permission as statistically independent. Two data generation algorithms that consider different structures and role hierarchies are also considered.

Tree-Based Data Generator The tree-based data generator assumes the following scenario. A company consists of a number of departments and each department has several offices. There are company-wide permissions that are shared by all employees. Different departments have their own department-wide permissions, which are assigned only to employees within the department. Also, different offices in a department have different job functions and thus each office has certain permissions that are assigned only to employees in that office. For example, an employee working

in the Business Office of Department A may have certain company-wide permissions, some permissions associated to Department A, and a number of permissions specific to the office she is working in. In general, department-wide permissions are never shared by users from different departments, while permissions specific to an office are never shared by users from different offices.

The tree-based data generator takes five parameters $\{n_u, n_p, h, b_0, b_1\}$, where n_u, n_p are the number of users and permissions respectively, h is the height of the tree, and b_0 and b_1 are the lower-bound and upper-bound of the number of children for each internal node of the tree respectively. The data generation algorithm consists of three steps. First, randomly generate a tree T of height h such that each internal node has $b \in\rangle[b_0, b_1]$ children. Let m be the number of nodes in T. Second, divide the set of permissions $\{p_1, \cdots, p_{n_p}\}$ into m disjoint sets $P_1, \cdots, P_m$. For every node n_i ($i \in\rangle[1, m]$) in T, associate P_i with n_i. Let $\{n_j, \cdots, n_m\}$ be the set of leaf-nodes in T. For every $i \in\rangle[j, m]$, compute P_i' such that P_i' contains all permissions associated with n_i or n_i's ancestors in T. Finally, divide the set of users $\{u_1, \cdots, u_{n_u}\}$ into $(m + 1 -\rangle j)$ disjoint sets $U_j, \cdots, U_m$. For every $i \in\rangle [j, m]$, use the random data generator to generate user-permission assignment UP_i between U_i and P_i'. Return $UP = \bigcup_{i=j}^{m} UP_i$.

ERBAC Data Generator Experiences from deploying RBAC systems in the real world suggested the Enterprise RBAC model, which uses a two-level layered role hierarchy [75]. In such a role hierarchy, there are two types of roles: functional roles and business roles. Permissions are only assigned to functional roles. Business roles are connected to functional roles and inherit all permissions from the connected functional roles. Finally, users are only assigned business roles and inherit all permissions from the assigned business roles.

The ERBAC data generator takes seven parameters: $\{n_u, n_{br}, n_{fr}, n_p, m_{br}, m_{fr}, m_p\}$ where n_u, n_{br}, n_{fr}, n_p are the number of users, business roles, functional roles, and permissions, respectively, and m_{br}, m_{fr}, m_p are

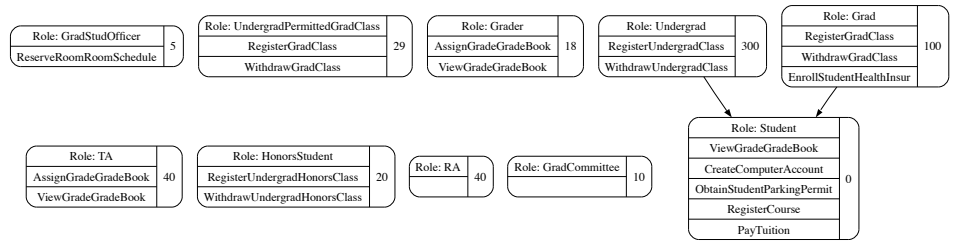
the maximum number of business roles a user can have, the maximum number of functional roles a business role can have, and the maximum number of permissions a functional role can have, respectively. The algorithm consists of four steps. First, for each functional role, a random number of permissions, up to m_p, are chosen to form the functional role. Second, for each business role, a random number of functional roles, up to m_{fr}, are assigned to the business role. Third, for each user, a random number of business roles, up to m_{br}, are assigned to the user. Finally, for each user, the user-permission assignments are computed.

5.3 Evaluation of *HierarchicalMiner* and *AttributeMiner* against Class 2 Algorithms

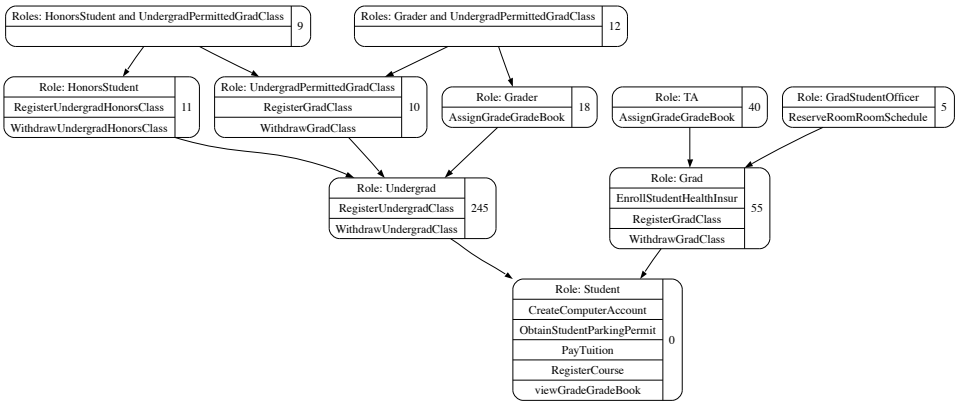
In this section, the effectiveness of *HierarchicalMiner* and *AttributeMiner* are evaluated. As the aim is to construct roles with semantic meanings, the resulting RBAC states are analyzed in addition to the WSC. Both *HierarchicalMiner* and *AttributeMiner* are evaluated against the **university** dataset and the semantic meaning of the resulting roles is manually analyzed. Next *HierarchicalMiner* is evaluated against several role mining algorithms from the literature on six real-world datasets without attributes on minimizing the structural complexity and other measures. Finally, the hybrid role mining techniques on the synthetic dataset are evaluated. The results show that *HierarchicalMiner* and *AttributeMiner* are able to generate RBAC states that have lower complexities than the original RBAC state, while preserving roles with semantic meanings and discovering some new roles with semantic meaning. Further, *HierarchicalMiner* reduces the complexity of RBAC states when allowing a role hierarchy better than the existing algorithms in the literature.

5.3.1 Synthetic Dataset with Attributes

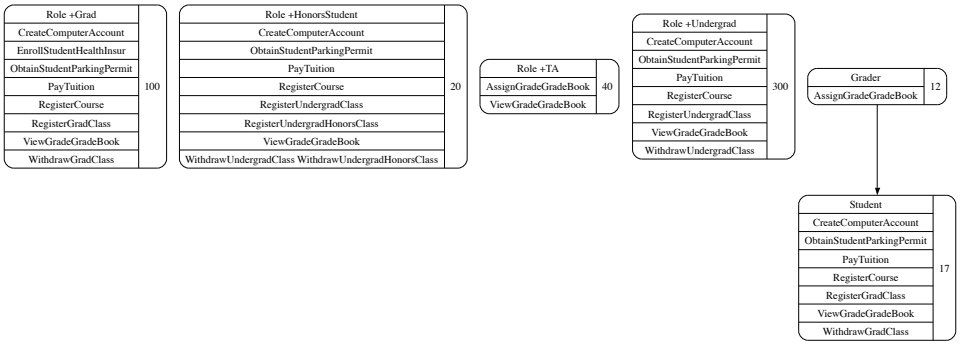
The complexity of the RBAC states generated using *HierarchicalMiner* and *AttributeMiner* using the **university** dataset are considered first. Table 5.2 shows the weighted structure complexities of the original role-engineered RBAC state, and the



(a) Original



(b) HierarchicalMiner



(c) AttributeMiner

Figure 5.2. Graphical Representation of roles in the student part of the university dataset: The original roles are shown in 5.2(a), the roles generated by *HierarchicalMiner* are shown in 5.2(b), and the roles generated by *AttributeMiner* are shown in 5.2(c). In the figures, the first line in a role is the name, the other lines are the permissions; the number to the right indicates the number of users assigned each role.)

Table 5.2
Mining results for the University dataset.

	$W = \{1,1,1,1,1\}$							$W = \{1,1,2,2,2\}$						
	R	UA	PA	RH	DUPA	CR	Total Cost	R	UA	PA	RH	DUPA	CR	Total Cost
Original	22	799	65	19	0	0	875	22	799	65	19	0	0	959
Hierarchical	18	499	57	14	12	0	600	19	500	66	15	2	0	685
Attribute	14	142	73	5	35	4	273	15	175	73	5	17	5	385

states generated by *HierarchicalMiner* and *AttributeMiner*. In the table, W is the weight vector defined in Definition 3.1.2. Two sets of weights are used to evaluate this dataset. Costs in columns show a breakdown of the total cost. The columns R, UA, PA, RH, $DUPA$, and CR represents number of role, user-assignment, permission-assignment, role hierarchy, direct user-permission assignment, and role membership (only used in attribute miner), respectively. The *Total Cost* column represents the total cost of the Weighted Structural Complexity from Equation 3.1.

For *AttributeMiner*, attributes that are likely to be maintained in a typical university data system are used, e.g., *Undergrad*, *Grad*, *HonorsStudent*, *TA*, *Faculty*. From the results, one can see that *HierarchicalMiner* generates significantly fewer roles and fewer user-role assignments than the original state. In fact, *HierarchicalMiner* reduced the size of each relation except the direct user-permission assignment relation. In our first test case, the total weighted structural complexity is reduced by 31%, while in the second test it is reduced by 29%. *AttributeMiner* is able to further decrease the complexity by replacing regular roles with attribute roles, resulting in a large number of user-role assignments being replaced by a single attribute-based role assignment.

Next consider the semantics of the roles. Figure 5.2 shows a portion of the original and generated states graphically. For compactness, the roles relating to students are presented; these contain more attributes. Figure 5.2(a) presents the original roles. There are ten roles and two role-hierarchy edges. By analyzing the resulting roles, ob-

serve that *HierarchicalMiner*, Figure 5.2(b), finds semantically meaningful roles. All except for two roles are from the original state. In some cases, the mined roles have more permissions than in the original state. For example, the role TA inherits the Grad role in the mined state. This is because in the original state all users assigned to the TA role are also assigned to the Grad role. *HierarchicalMiner* found this implicit semantic relationship, while also reducing the complexity. *HierarchicalMiner* finds two roles that do not exist in the original state; they are the composite roles *HonorsStudent and UndergradPermittedGradClass*, which has 9 users, and *Grader and UndergradPermittedGradClass*, which has 12 users. They represent meaningful concepts. Names for the roles generated by *HierarchicalMiner* have provided manually for comparison.

AttributeMiner, Figure 5.2(c), finds four attribute roles corresponding to *Grad*, *UnderGrad*, *HonorStudent*, and *TA*, having 100, 300, 20, and 40 users respectively. *AttributeMiner* uses fewer roles and has more direct user-permission assignments. It does not create those roles that have a smaller number of users. The attributes provide a role name automatically, easing administrator comprehension.

This illustrates how our algorithms *HierarchicalMiner* and *AttributeMiner* can be used to generate RBAC states with low complexity while creating semantically meaningful roles on a synthetic dataset. The next section will evaluate how well *HierarchicalMiner* performs at minimizing the structural complexity on six real-world datasets over a variety of weighted structural complexity vectors compared to other algorithms in the literature. Because *HierarchicalMiner* can be extended using *AttributeMiner* to handle attributes, these results generalize to *AttributeMiner*.

5.3.2 Evaluation on Real-World Datasets

In the previous section we evaluated *HierarchicalMiner* and *AttributeMiner* on a single, synthetic dataset representing a university. In this section we compare the results of *HierarchicalMiner* against three algorithms from the literature at reducing

the complexity of six real-world datasets, and the university dataset from the previous section.

For comparison graph optimization (GO) [8], and the heuristic role minimization (HPr) and edge-concentration (HPe) algorithms [19] are used. RBAM [10] is also omitted based on poor results on our running example and excessive running time on the example datasets. RBAM traverses up a lattice of candidate roles $\wp(P)$, the power set of the permissions P, and prunes when certain conditions are met. The number of roles it considers is either excessively large, or the minimum support must be set such that it eliminates a large number of rare permissions [11].

The graph optimization and edge-concentration (HPe) algorithms are modified to use weighted structural complexity to produce a more fair comparison. For example, the edge-concentration algorithm may increase the number user-assignments if it causes a greater decrease in the number of permission-assignments (at the expense of creating an additional role). The same techniques used to apply a weight vector to *HierarchicalMiner* are applied to these algorithms when evaluating such a tradeoff.

Two Class 1 algorithms are also considered: *CompleteMiner* (CM); and *PairCount* (PC), which uses the number of pairs of roles as the prioritization method and the *FastMiner* algorithm for role generation [67]. All Class 1 algorithms are converted to Class 2 output. For each objective, the number of roles k is selected that minimizes (or maximizes) the desired measure. Because the ORCA algorithm cannot be readily modified to minimize the WSC, it is first converted to Class 1 output, and treated as a Class 1 algorithm. All experiments are run five times, and non-zero standard deviations are given in parentheses.

The results for a variety of representative weight vectors are presented in Tables 5.3–5.7. For each objective and dataset, the minimum solution generated by the representative algorithms is highlighted in bold. First, consider the benchmark criteria, minimizing the number of roles. As shown in Table 5.3, the role and edge minimization algorithms (HPr and HPe) produce the minimum number of roles among

the selected algorithms[4], while the *HierarchicalMiner* actually performs the worst (excluding the Class 1 algorithms). HPr finds maximal bicliques (concepts) to cover the user-permission relation, allowing one to implement this algorithm using the formal concept lattice with minimal effort. This illustrates one shortcoming of the *HierarchicalMiner* algorithm, but not the foundation of formal concept analysis. It was illustrated in Theorem 3.2.1 that the optimal set of candidate roles required to minimize the number of roles are a subset of the set of formal concepts.

The priority of each role in a Class 1 algorithm is independent, making them ill suited at selecting a minimal set of candidate roles when selecting the top k. They consistently perform poorly across all experiments.

Table 5.3
Overview of the datasets and mining results when $W = \langle 1, 0, 0, 0, \infty \rangle$.
Non-zero standard deviation in parentheses.

	Class 2				Class 1	
Source	HM	GO	HPr	HPe	PC	CM
APJ	542 (4.6)	564	**454 (0.44)**	**454 (0.44)**	779	764
Domino	27 (2.28)	23	**20**	**20**	64	62
EMEA	106 (6.1)	**34**	**34**	**34**	242	674
Firewall 1	91 (3.34)	90	**66**	**66**	248	278
Firewall 2	**10 (0.89)**	11	**10**	**10**	14	21
Healthcare	17 (1.8)	18	**14**	**14**	24	31
University	23 (1.3)	22	**17**	**17**	31	32
Rank	3.43	3.29	**1.64**	**1.64**	5.29	5.71
Average	116.57	108.86	**87.86**	**87.86**	200.28	266

Next, consider the problem of edge-RMP, or minimizing the number of user- and permission-assignments, in Table 5.4. This is an optimization HPe was explicitly designed for, and predictably performs well at. With this objective, *HierarchicalMiner*

[4]Note that with this weight vector these algorithms are identical.

produces more compact results than the graph optimization and role minimizations. This represents a tradeoff between their edge-RMP algorithm and our *HierarchicalMiner*. *HierarchicalMiner* prunes the formal concept lattice, and the initial set of candidate roles have a large degree of overlap, which is useful when a role hierarchy is allowed. However, without the role hierarchy, it leads to an increase in user- and permission-assignments. To remedy this, one could apply the post processing optimizations of HPe to the candidate roles of our *HierarchicalMiner*, however we believe this impacts the semantics of the resulting roles, making them no longer concepts, and potentially impacting interpretation.

Table 5.4
Total WSC when $W = \langle 0, 1, 1, \infty, \infty \rangle$. Standard deviation in parentheses.

	Class 2				Class 1	
Source	HM	GO	HPr	HPe	PC	CM
APJ	4299 (3.1)	5565	4875 (3.1)	**4016 (7.6)**	7163	6946
Domino	539	716	741	**408 (3.1)**	1498	1136
EMEA	7920 (39.4)	7246	7246	**3963 (38.0)**	21983	43051
Firewall 1	2866 (0.55)	7100	3413	**1861 (72.0)**	16633	19788
Firewall 2	1364	1499	1554	**1146**	1525	1559
Healthcare	232	545	369	**216 (5.9)**	589	706
University	725	752	887	**665**	786	786
Rank	2.29	3.36	3.79	**1.00**	5.07	5.50
Average	2563.57	3346.14	2727.43	**1095**	7168.14	10567.43

The main benefits of *HierarchicalMiner* can be observed when a role hierarchy is allowed. In the following experiments we augment edge-RMP to allow a role hierarchy and include the number of roles ($W = \langle 1, 1, 1, 1, \infty \rangle$), the optimization criteria for graph optimization [8]. The results are shown in Table 5.5. Here *HierarchicalMiner* algorithm outperforms the other role mining algorithms from the literature on all but

Table 5.5
$W = \langle 1, 1, 1, 1, \infty \rangle$. Standard deviation in parentheses.

	Class 2				Class 1	
Source	HM	GO	HPr	HPe	PC	CM
APJ	**4270 (7.4)**	4600 (42.3)	5337 (3.6)	4475 (12.6)	6496	6815
Domino	419 (2.5)	**413 (5.4)**	761	437 (6.5)	637	611
EMEA	**3790 (29.9)**	3888 (18.6)	7280	4222 (42.8)	10464	12728 (1.6)
Firewall 1	**1425 (5.1)**	1543 (28.6)	3478	2020 (20.0)	3057	2422
Firewall 2	**948 (1.1)**	960 (3.2)	1564	1161 (9.4)	1000	1021
Healthcare	**151 (1.6)**	168 (10.2)	383	240 (2.8)	213	261
University	**605 (1.1)**	607 (2.5)	904	703 (8.2)	626	627
Rank	**1.14**	2.00	5.43	3.57	4.14	4.71
Average	**1658.29**	1739.86	2815.29	1894.00	3213.28	3497.85

one dataset. On the **domino** dataset, graph optimization produces the most compact RBAC state.

Finally, direct user-permission assignments are allowed (Table 5.6) and permission assignments are more heavily weighted (Table 5.7). In both of these instances *HierarchicalMiner* produces a more compact RBAC state as measured by weighted structural complexity. These tests indicate the *HierarchicalMiner* algorithm is highly adaptable to a wide variety of role mining criteria, and produces compact RBAC states that maintain some semantic meaning when concepts can be interpreted. Further, the role hierarchy allows an administrator to make additional inferences not allowed by the other algorithms.

Two role mining objectives, role minimization and edge concentration, have been identified where other algorithms from the literature produce a more compact RBAC state. In each instance, the algorithm producing the most compact state was designed to minimize the objective specifically; these algorithms can also be implemented using

the formal concept lattice. This illustrates the advantages of building upon formal concept analysis as a foundation for many role mining algorithms.

Table 5.6
Total WSC when $W = \langle 1, 1, 1, 1, 1 \rangle$. Standard deviation in parentheses.

	Class 2				Class 1	
Source	HM	GO	HPr	HPe	PC	CM
APJ	**3863 (3.9)**	4600 (17.8)	5337 (3.6)	4489 (22.7)	4733	4985
Domino	**381 (0.9)**	413 (5.5)	761	437 (1.7)	501	495
EMEA	**3706 (12.8)**	3888 (23.8)	7280	4208 (44.8)	5811	6364
Firewall 1	**1355 (6.3)**	1543 (26.6)	3478	2005 (16.0)	2258	2678
Firewall 2	**945**	960	1564 (2.9)	1161	992 (1.3)	995
Healthcare	**144 (0.4)**	168 (11.3)	383	244 (2.5)	148	164
University	**600 (1.1)**	607 (5.8)	904	688 (5.2)	619	620
Rank	**1.00**	2.43	6.00	3.71	3.57	4.29
Average	**1570.57**	1739.86	2815.29	1890.29	2151.71	2328.71

5.4 Evaluations using Class Conversions

The previous section compared *HierarchicalMiner*, a Class 2 algorithm against other Class 2 algorithms and Class 1 output converted to Class 2. This section presents results from evaluating role mining algorithms where all output is first converted to Class 1. The ability to quickly and globally optimize an RBAC state is evaluated.

The ability of the role mining algorithms to minimize the weighted structural complexity for the weight vector $W = \langle 1, 1, 1, 1, 1$ has already been evaluated. If the roles are a poor fit for the user-permission relation, many assignments will be covered via direct user-permission assignments and are exceptions to the RBAC state. The fitness of these roles can be measured by evaluating the fraction of assignments that

Table 5.7
$W = \langle 1, 1, 5, 1, 5$. Standard deviation in parentheses.

	Class 2				Class 1	
Source	HM	GO	HPr	HPe	PC	CM
APJ	**8995 (3.7)**	9793 (18.9)	10877 (1.8)	10038 (10.1)	11300	11799
Domino	**1346 (1.0)**	1408 (6.2)	3017	1419 (5.2)	1568	1553
EMEA	**16146 (4.1)**	16424 (25.8)	36124	16966 (60.2)	46484	27448 (0.8)
Firewall 1	**4258 (1.0)**	4532 (66.5)	6926	5286 (42.8)	7515	7887
Firewall 2	**3309**	3323	3928 (6.1)	3618	3576	3527
Healthcare	**334**	386	627 (5.2)	533	432	461
University	**837 (0.9)**	843 (15.5)	1176	1074	867	868
Rank	**1.00**	2.00	5.14	3.86	4.29	4.71
Average	**5032.14**	5244.14	8953.57	5562.00	10248.86	7649.00

are exceptions, i.e., $1 - \mathsf{coverage}$, where $\mathsf{coverage}$ is the fraction of the user-permission relation covered by roles. The results are presented in Table 5.4.

It can be seen that *HierarchicalMiner* consistently uses a low number of direct user-permission assignments, despite the ability to effectively reduce the total weighted structural complexity (see Table 5.6). However, it can also be observed that the solution to the role minimization problem, as represented by *HPr*, produces roles where many (the majority in some datasets) of the user-permission assignments are exceptions. Consider the **emea** dataset; the *HPr*, *CompleteMiner*, and *PC* states use direct user-permission assignments to cover over 75–100% of *UP*, while *HierarchicalMiner*, *GO*, and *HPe* use significantly less (33% and 34% for *HierarchicalMiner* and *GO*). Similar results are seen in the **domino** dataset. This indicates role minimization may be an undesirable objective.

Table 5.8
$\frac{|DUPA|}{|UP|}$ for $W = \langle 1, 1, 1, 1, 1$

	Class 2					Class 1	
	HM	GO	ORCA	HPr	HPe	PC	CM
Americas	**0.00**	0.01	0.36	**0.00**	**0.00**	0.08	0.22
APJ	0.32	0.33	0.45	0.41	**0.31**	0.36	0.66
Domino	0.26	0.24	0.60	0.97	0.25	**0.18**	0.22
EMEA	0.34	**0.33**	0.94	1.00	0.40	0.75	0.86
Firewall 1	0.01	0.01	0.48	0.02	**0.00**	0.03	0.03
Firewall 2	**0.00**	**0.00**	0.77	**0.00**	**0.00**	**0.00**	**0.00**
Healthcare	0.01	0.01	**0.00**	0.04	0.04	0.02	0.02
University	**0.00**	0.01	0.34	0.04	0.01	**0.00**	**0.00**
Average	**0.12**	**0.12**	0.49	0.31	0.13	0.18	0.25

5.4.1 Prioritized Role Quality

Section 5.2.1 proposed a metric that integrates an evaluation criteria, such as cost or coverage, over the number of top roles selected. To perform this analysis, all output must first be prioritized (strictly ordered) by converting to Class 1. The top k roles are then converted to Class 2 and evaluated for each k, and integrated.

Table 5.4.1 lists the Q_{wsc} metric normalized across the selected algorithms to a [0,1] range. *HierarchicalMiner* minimizes the Q_{wsc} for all datasets except the **university** dataset. The next best role mining algorithms are *HPe* and *GO*. This indicates *HierarchicalMiner* is capable of minimizing the total weighted structural complexity, and maintains a low complexity for a reduced number of roles.

Table 5.4.1 list the $\mathsf{Q}_{\mathsf{coverage}}$ metric normalized by the best result (maximal coverage) for each dataset. *HPr* performed well at quickly covering the *UP* relation (this metric is consistent with role-minimization). *GO* and *HPe* each performed con-

Table 5.9
Normalized Q_{wsc} for $W = \langle 1, 1, 1, 1, 1$

	Class 2						Class 1	
	k	HM	GO	ORCA	HPr	HPe	PC	CM
Americas	150	**0.00**	0.12	1.00	0.24	0.03	0.30	0.54
APJ	150	**0.00**	0.11	0.67	0.94	0.06	0.72	1.00
Domino	35	**0.00**	0.51	0.95	0.52	0.47	0.98	1.00
EMEA	34	**0.00**	0.33	0.89	1.00	0.20	0.52	0.72
Firewall 1	30	**0.00**	0.08	1.00	0.06	0.01	0.09	0.17
Firewall 2	10	**0.00**	0.01	1.00	0.08	0.04	0.28	0.36
Healthcare	15	**0.00**	0.07	1.00	0.27	0.10	0.05	0.07
University	15	0.04	0.09	1.00	0.16	**0.00**	0.25	0.32
Average		**0.00**	0.16	0.94	0.41	0.11	0.40	0.52

sistently with previous tests where both algorithms have proven capable at minimizing cost and the number of roles. *HierarchicalMiner*'s performance was consistent with role-minimization. The Class 1 algorithms continued to perform poorly.

5.4.2 Discovering Original Roles

The above analysis is based on real-world data without an existing RBAC state. This section evaluates the similarity of the mined roles to a set of original roles. Because real-world data containing original roles is not available, the analysis is performed on generated data. Five datasets are generated using each data generating method. All results indicate the average over the five samples.

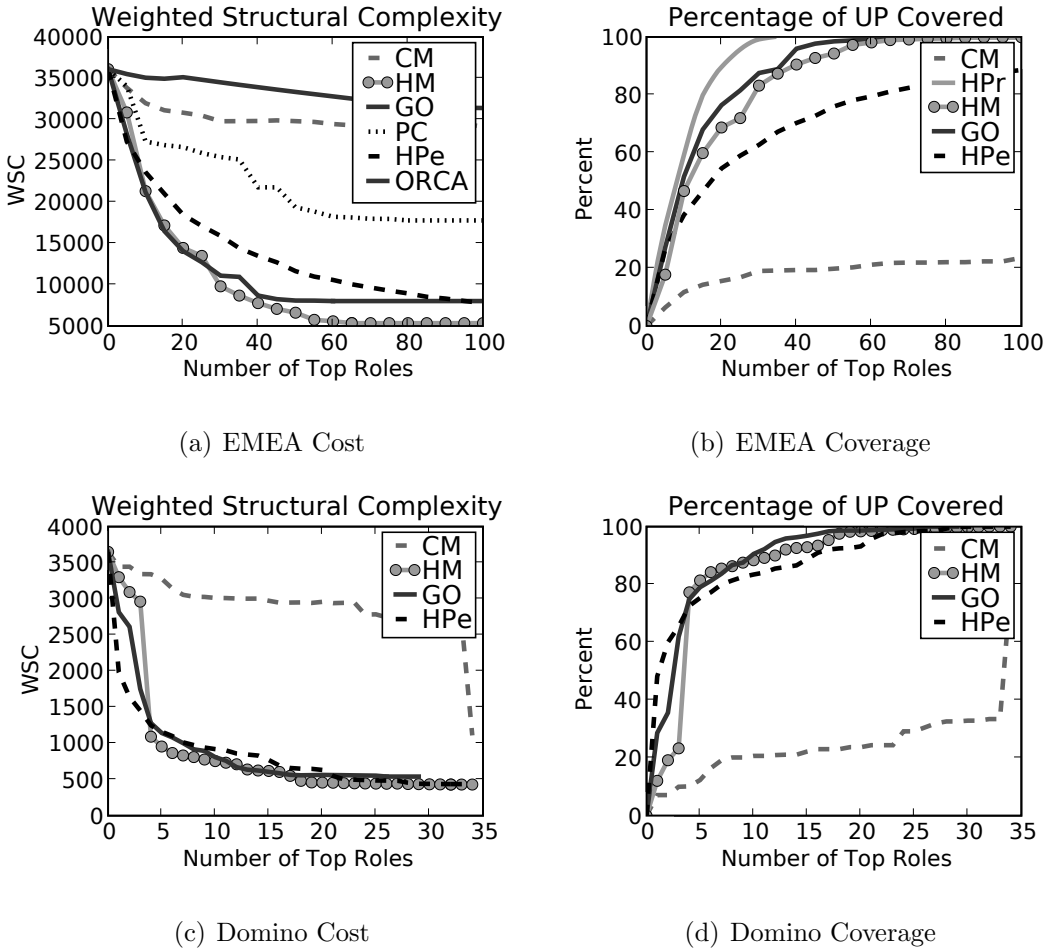
To evaluate the similarity of the mined roles to the original roles, the average maximal Jaccard is used. The results are plotted in Figure 5.4. A good role mining algorithm should produce familiar (high Jaccard) roles first.

Table 5.10
Normalized $\mathsf{Q}_{\mathsf{coverage}}$ for $W = \langle 1, 1, 1, 1, \infty \rangle$

	Class 2						Class 1	
	k	HM	GO	ORCA	HPr	HPe	PC	CM
Americas	150	0.90	0.91	0.00	0.90	**1.00**	0.90	0.85
APJ	150	0.39	0.80	0.00	0.86	0.41	**1.00**	0.33
Domino	35	0.84	0.95	0.02	0.94	**1.00**	0.04	0.00
EMEA	34	0.57	0.83	0.00	**1.00**	0.74	0.18	0.11
Firewall 1	30	0.96	0.81	0.00	**1.00**	0.89	0.99	0.98
Firewall 2	10	**1.00**	0.99	0.00	0.93	0.96	0.74	0.66
Healthcare	15	**1.00**	0.93	0.00	0.86	0.90	0.99	0.98
University	15	0.94	0.90	0.00	**1.00**	0.84	0.82	0.73
Average		0.82	0.89	0.00	**0.94**	0.84	0.71	0.58

Two conclusions may be drawn from the figures. In the ERBAC data for *HM*, it can be seen that the top 40 roles have a Jaccard close to one, and the Jaccard quickly begins to fall for subsequent roles. This means that the top 40 roles are very similar to the ones created by the data generator. By comparing these results to the coverage in Figure 5.4, one can see that this drop correlates to around 98% coverage. While *GO* does not select as many roles that so closely approximate the original RBAC data, it more consistently selects roles with a high Jaccard compared to the other role-mining algorithms. Excluding *PC* (and the related algorithms), *HPr* performed the worst, generating roles farthest from the original data.

The results for the tree data are very different. *HierarchicalMiner* still performs slightly better than *GO*, but they are among the worst performers, and *HPr* and *HPe* perform the best. Regardless of the algorithm used, the mined roles generated from tree-based data did not resemble the original roles as closely when compared to the ERBAC data.



(a) EMEA Cost

(b) EMEA Coverage

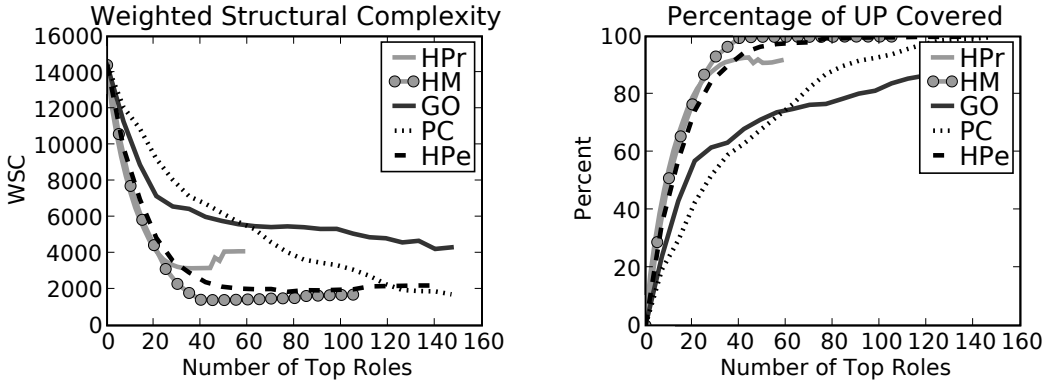
(c) Domino Cost

(d) Domino Coverage

Figure 5.3. Plots of the minimal WSC and maximal coverage for several algorithms and datasets.

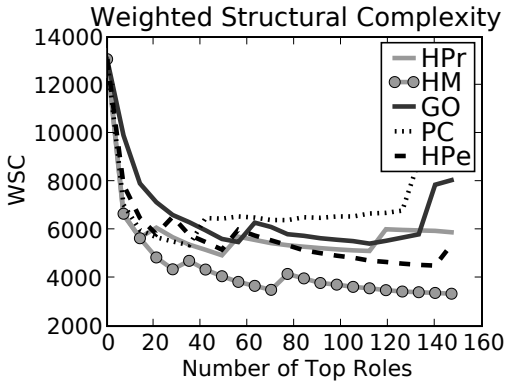
5.4.3 Hybrid Role Mining the University Dataset

The hybrid role mining technique presented in Section 3.4 is evaluated using the **university** dataset. Table 5.4.3 gives the size of each relation in the original and optimized versions of each state. It can be seen that optimizing the permission-assignment relation with $W = \langle 1, 1, 1, 0, \infty \rangle$ produces a more compact RBAC state, reducing the WSC by over 10% when compared to the original input. This indicates the top-down RBAC state can be optimized without considering the set of users. Using just the user-permission relation produced the most compact RBAC state.

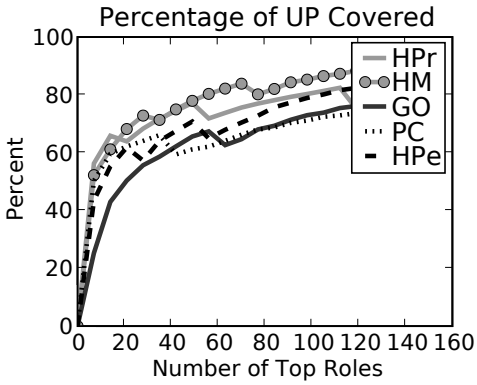


(a) ERBAC Cost

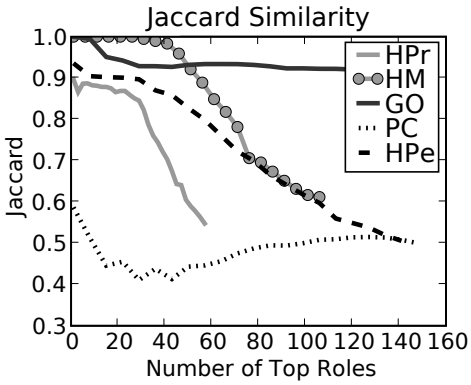
(b) ERBAC Coverage



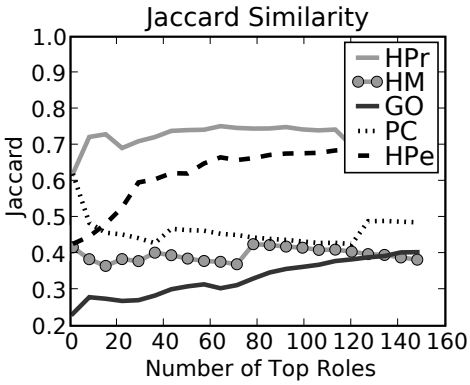
(c) Tree Cost



(d) Tree Coverage



(e) ERBAC Jaccard



(f) Tree Jaccard

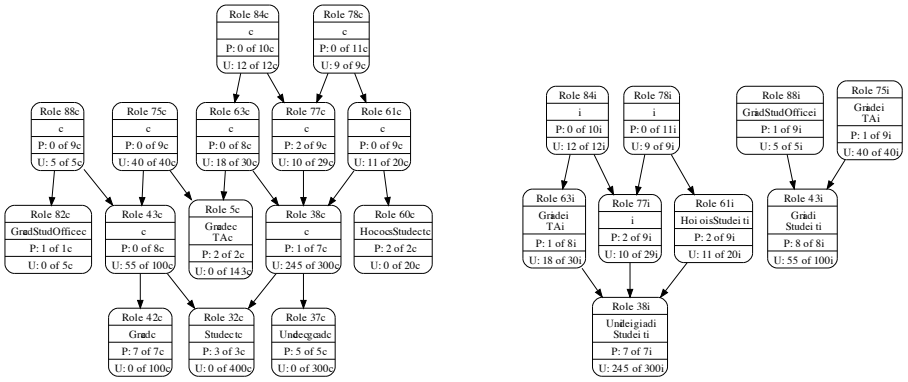
Figure 5.4. Role similarity for generated datasets.

To test the hybrid role mining approach, 80% of the permissions assigned are randomly selected for the "sketch" of each role; this defines the preliminary *PA* relation. When the predefined roles are forced to be created, an increase in the WSC can be observed. Removing the set of users representing predefined roles and further optimizing the RBAC state, eliminating most of the complexity caused by introducing the predefined roles.

The predefined roles aid in adding semantic meaning to the candidate roles, and the user-permission relation is used to complete the permissions assigned to the predefined roles. Figure 5.5 shows the candidate roles pertaining to students when the incomplete predefined roles are forced, and when we allow them to be pruned. By comparing Figure 5.5(a) (forced) to Figure 5.5(b) (may be pruned), one can see how the user-permission relation is used to complete the predefined roles. For example, the *Undergrad* role in Figure 5.5(a) is missing two permissions (see Figure 5.2(a)), and is not senior to the *Student* role. These omissions are corrected automatically in Figure 5.5(b), and the correct set of permissions for the *Undergrad* role is the closure, $\mathcal{P}^{\uparrow\downarrow}_{Undergrad}$.

Table 5.11
Mining the University dataset with $W = \langle 1, 1, 1, 1, \infty \rangle$. The permission-assignment relation is mined with $W = \langle 1, 1, 1, 0, \infty \rangle$.

Approach	$\lvert R \rvert$	$\lvert UA \rvert$	$\lvert PA \rvert$	$\lvert RH \rvert$	$\lvert DUPA \rvert$	Total
Original	32	799	64	19	0	914
PA Only	20	722	58	12	0	812
UA Only	18	499	57	14	12	600
Force R	44	502	96	49	0	691
Force and Remove R	22	504	79	12	0	617



(a) Forcing predefined roles R.

(b) Removing artificial users from 5.5(a) and optimizing.

Figure 5.5. The resulting RBAC states using hybrid role mining. Predefined roles are promoted by the closure $\uparrow\downarrow$.

5.5 Evaluating Mining Noisy Data

The performance of several matrix decomposition models and algorithms from the literature is evaluated using four representative datasets: **university**, **ERBAC**, **Random**, and **Tree**. ERBAC has 500 users, 347 permissions, and a density of 8.5%; Random has 500 users, 100 permissions, and a density of 22%; and Tree has 500 users, 190 permissions, and a density of 4%.

As opposed to [7, 13], noise is not added uniformly. First, more over-assignments are added than under-assignments. This based on the noise observed in real-world datasets and is consistent with the logic behind the principle of least privilege (any under-assignments will be discovered when access is denied and corrected while over-assignments often go unnoticed). Next, users and permissions are selected from a multinomial distribution with normally distributed probabilities, causing some permissions to be incorrectly assigned or revoked more than others. For each dataset five noisy representations are generated, and the results represent the average over all five datasets. Noise amounts range from 5% to 15% of the size of the user-permission

relation $||UP||_1$, and not $|U|*|P|$. All algorithms know the total number of errors $\delta\sigma$ (Type I plus Type II errors) and are run on the noisy datasets and selects users and permissions uniformly.

The δ-RMP algorithm is run 1000 times, and the solution that minimizes the number of roles is returned. In the case of a tie (two or more solutions have the same minimum rank), the mean over is returned. The δ-RMP algorithm does not know the distribution of over- and under-assignments.

The binary non-orthogonal decomposition (BND) algorithm is run with $e = 1$ (Hamming radius) and the number of resulting patterns p (sub matrices) is noted. No constraints are placed on the minimum size of each cluster.

The disjoint decomposition model (DDM) determines the appropriate number of users and permission clusters using the infinite relational model (IRM) and only uses the noise level for assigning business roles to technical roles. An implementation of the Infinite Relational Model[5] from Charles Kemp [56] is used to implement DDM. Subroles in which the DDM algorithm is re-run on sub-matrices as described in [13] is not considered.

For the remaining algorithms, when possible, multiple rank decompositions are performed and the lowest rank k is selected such that the L1-norm of the approximation and the noisy input data is less than δ. This uses the method to select k described in Section 4.3.2.

The code for MAC[6] was supplied by the authors [15]. Because the unconstrained version of MAC allows a user to be assigned any item in the power-set of $[0, k]$, the complexity is exponential. Based on recommendations from the authors, users are constrained to be assigned at most 2 of the k clusters. Even given this constraint the running time of MAC was prohibitive compared to the other matrix decomposition methods (days for MAC compared to seconds or minutes for SVD, BNMF, etc.). The

[5] `http://www.psy.cmu.edu/~ckemp/code/irm.html`

[6] `http://www.inf.ethz.ch/personal/mafrank/paper/MAC_1.0.zip`

Table 5.12
University Dataset: 197 Type I errors and 39 Type II errors.

Method	Options	False Pos.		False Neg.		Total	
δ-RMP	$k = 58.1$ ($= 1.01$)	393.6	6.35	72.9	4.65	466.6	3.55
BND	$p = 23.2$ ($= 2.2$)	94.2	9.12	65.2	9.36	159.4	16.83
BNMF	$k = 13.05$ ($= 0.05$)	35.48	6.07	29.1	8.72	64.58	8.84
DDM	BR=14.2, TR=14.8	**2.8**	1.64	1641	521.02	1643	491.48
LPCA	$k = 3$ ($= 0$)	49.28	4.24	50	7.24	99.28	8.71
MAC	$k = 12$ ($= 0$)	147	60.32	79.8	22.35	226.8	41.19
NMF	$k = 18.25$ ($= 2.09$)	43.05	11.78	63.55	4.22	106.6	13.81
SVD	$k = 10.25$ ($= 0.25$)	28.5	4.80	**26**	7.62	**54.5**	6.61

Table 5.13
ERBAC Dataset: 740 Type I errors and 148 Type II errors.

Method	Options	False Pos.		False Neg.		Total	
δ-RMP	$k = 579$ ($= 19.5$)	1529.6	14.30	226.4	11.48	1756	7.48
BND	$p = 353.8$ ($= 45.7$)	683.4	16.33	136.8	6.83	820.2	22.21
BNMF	$k = 34.8$ ($= 1.08$)	7.78	4.60	6	3.48	13.78	6.05
DDM	BR=49.2, TR=116.2	**1**	1	6602	368.33	6603	367.54
LPCA	$k = 9$ ($= 0$)	314.3	23.38	357.6	38.86	671	50.30
MAC	$k = 60$ ($= 0$)	92.2	23.12	250	44.40	342.4	8.84
NMF	$k = 37.9$ ($= 4.0$)	8.2	5.45	9	6.72	17.2	8.84
SVD	$k = 31$ ($= 1.3$)	3	0.84	**5.8**	4.78	**8.8**	4.11

appropriate rank k is selected empirically based on the reconstruction error and we thank the authors for their assistance in this task.

The results are shown in Tables 5.12–5.15. The average number of false positives and false negatives and the variance is given.

Table 5.14
Random Dataset: 573 Type I errors and 114 Type II errors.

Method	Options	False Pos.		False Neg.		Total	
δ-RMP	$k = 108.6$ ($= 10.3$)	1071.6	5.72	269.6	4.98	1341.2	8.56
BND	$p = 136.8$ ($= 42.2$)	493.8	19.93	123.4	9.71	617.2	27.60
BNMF	$k = 17.5$ ($= 0.82$)	28.9	12.20	6.4	8.78	35.3	17.52
DDM	BR=21.6, TR=22.6	**0.2**	0.44	3344	13.44	3344	424.89
LPCA	$k = 5$ ($= 0$)	140.55	16.28	134.5	23.87	275	34.68
MAC	$k = 30$ ($= 0$)	685.4	4.16	132.8	3.67	818	22.60
NMF	$k = 19.1$ ($= 1.95$)	17.7	9.45	6.3	3.67	24.0	10.35
SVD	$k = 14.8$ ($= 0.91$)	9.5	7.04	**2.25**	1.89	**11.8**	6.23

Table 5.15
Tree Dataset: 198 Type I errors and 38 Type II errors.

Method	Options	False Pos.		False Neg.		Total	
δ-RMP	$k = 207.7$ ($= 27.5$)	422.8	5.97	48.5	5.75	471.3	2.07
BND	$p = 238.2$ ($= 21.7$)	137.2	6.61	25.6	3.38	162.8	5.12
BNMF	$k = 41.6$ ($= 1.32$)	16.6	5.48	**4.6**	1.67	**21.3**	5.15
DDM	BR=42.4, TR=50	**0**	0	2413	197.60	2413	6.25
LPCA	$k = 7$ ($= 0$)	122.4	10.51	67.7	14.38	190	17.70
MAC	$k = 50$ ($= 0$)	29.2	10.82	6	2.12	35.2	10.94
NMF	$k = 41.7$ ($= 0.69$)	14.55	4.21	5.05	3.29	19.6	4.22
SVD	$k = 39$ ($= 0$)	15.8	3.40	10.5	3.70	26.2	6.30

It can be seen that in the four representative samples, the singular value decomposition, non-negative matrix factorization, and Bayesian non-negative matrix factorization had excellent performance. The logistic PCA finds a lower rank approximation than the competing algorithms and is able to remove a smaller amount of noise. The next section will illustrate that logistic PCA has better predictive performance when values are held out. All four algorithms were capable of removing the noise without excessive under-assignments of permissions.

The large number of false negatives in the DDM model is consistent with the observation that it provides clusters that are too coarse to capture the user-permission relation. Multi-assignment clustering also produces a large number of false negatives at comparable ranks. For example, in the Random dataset (Table 5.14), using 30 clusters MAC produced 818 total errors on average and takes 5,715 seconds while SVD produced 11.8 errors in only 0.02 seconds. These experiments were performed on a 2.0 GHz Core 2 Duo running MATLAB 7.8.

The MAC model appears to be better suited to datasets with fewer clusters that are large and dense, and not the sparse datasets that appear in many real-world access control datasets evaluated in this dissertation. Both DDM and MAC may be better suited to identifying applicable assignments, but may be ill suited at identifying errors in the user-permission relation, and simpler and less costly methods may be better suited for those purposes.

Table 5.16
Using SVD for all eigenvalues greater than or equal to 1.0 and threshold 0.5.

		Errors		> 1.0			$\alpha\sigma > 80\%$		
Source	UP	Type I	Type II	FP	FN	Recall	FP	FN	Recall
University	4129	197	23	37	216	93.9%	4	308	87.9%
ERBAC	15399	736	154	258	479	95.2%	86	1501	85.78%
Random	11844	568	190	73	1654	85.4%	87	1905	79.4%
Tree	4172	194	5	0	3350	19.7%	3	676	79.2%

Table 5.16 provides the results when using the variance and singular values in SVD to determine the rank. When compared to the results in Tables 5.12–5.15, one can see this approach provides a more coarse reconstruction, typically resulting in more under-assignments and lower recall. From the *Tree* dataset it can be seen that many permission assignment patterns may be weak and small, possibly indicating exceptions and explaining the performance of DDM.

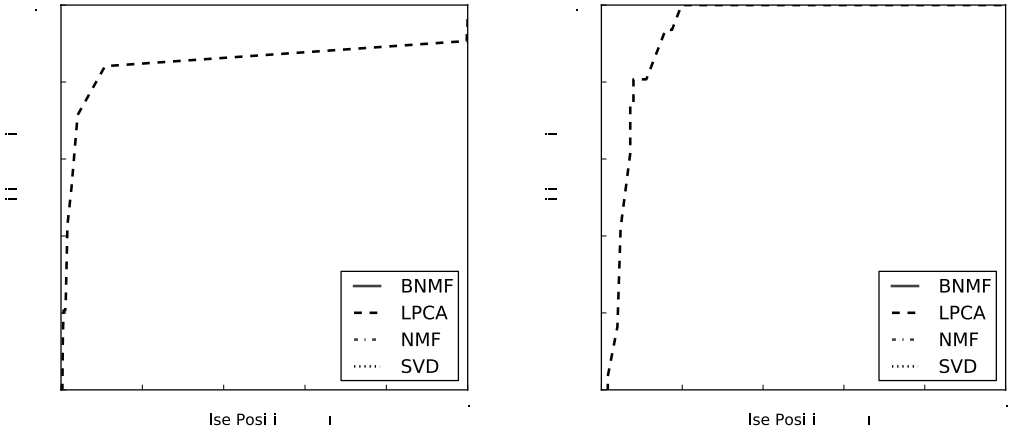
Prediction Accuracy

The previous evaluation used artificial datasets providing a ground truth that the noise removal methods can be evaluated against. In this section the performance of the top matrix decomposition models, SVD, NMF, BNMF, and logistic PCA, is evaluated using real datasets from [19, 67].

It is not known whether the real datasets contain any noise and false positives cannot be separated from true positives (and similar for true and false negatives). As a result, noise is not added to the real datasets for evaluation. Instead the ability of the models to predict several known values is measured. Twenty percent of the user-permission matrix is selected and held out, and the accuracy of the prediction is measured.

Table 5.17
AUC for LPCA, SVD, NMF and BNMF on three real datasets.

	Domino		Firewall		Healthcare	
Algorithm	k	AUC	k	AUC	k	AUC
LPCA	20	**.962**	19	**.999**	6	.990
SVD	4	.952	7	.995	3	.993
NMF	4	.954	5	.995	4	**.994**
BNMF	3	.953	13	.995	3	.993



(a) Domino (b) Healthcare

Figure 5.6. ROC curve for real datasets.

Any prediction method (in our case, deciding whether or not a user should be assigned a permission) must make a tradeoff between false positives and false negatives. In our case, this is done by selecting a threshold t. This tradeoff is often illustrated with a receiver operating characteristic (ROC) curve, plotting the true positive rate (ratio of true positives predicted) against the false positive rate (false positives divided by negatives). Two ROC curves can be compared by evaluating the area under the curve (AUC), where 1.0 implies a perfect predictor. The ROC curves for two datasets is presented in Figure 5.6, and Table 5.5 lists the rank of each decomposition given the best AUC. The predictive accuracy for the held out values only is measured. These methods all have excellent performance at predicting the held out values, with logistic PCA performing the best in general. These results confirm the previous experiments.

5.5.1 Prediction Using Attributes

Section 4.6 used a measure of the entropy reduction from Frank et al. [14] to predict which attributes will provide the best predictive performance on the **anonymous**

dataset. This section evaluates how well these attributes improve the actual predictive performance. Logistic PCA is used to reconstruct both the user-attribute relation Y (which has been converted to a binary relation) and the user-permission relation X. Similar to the previous section, 20% of the data is held out and the mean absolute error (MAE) of the zero-one loss for the binary relation is measured. All experiments are performed 10 times and the rank of the decomposition is ten. The parameter $\alpha\sigma$ trading off the loss of X and Y is varied from zero to one. The resulting MAE loss given $\alpha\sigma$is shown in Figure 5.7. Each attributes is evaluated based on how well it reduces the predictive error. A summary of these results in given in Table 5.18.

Table 5.18
The total entropy of the user-permission relation given knowledge of a user's attribute.

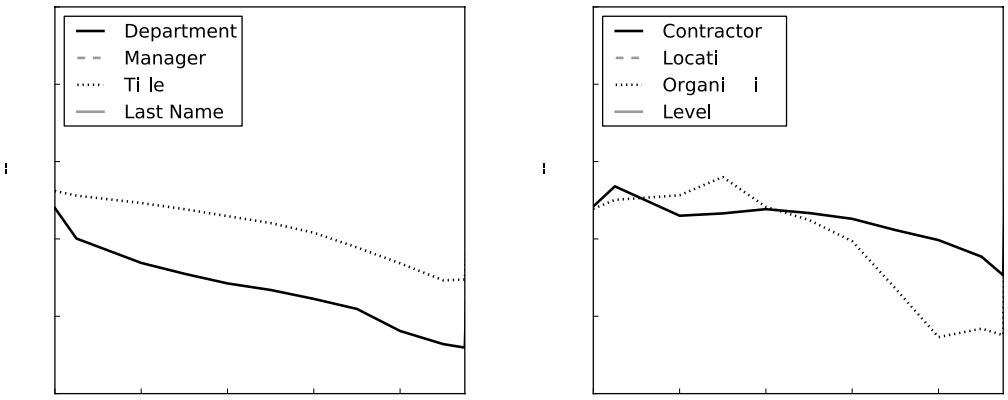
Attribute	Cardinality	$\sum_{p \in P} I(p, A)$	Predictive Improvement	Std. Dev.
Last Name	2224	2769.39	8.50%	0.24
Manager	298	2186.03	20.34%	0.14
Department	192	1931.95	25.41%	0.24
Title	527	1878.51	15.03%	0.02
Location	53	1316.92	21.44%	0.14
Organization	12	789.46	18.53%	0.51
Level	17	170.34	18.59%	0.1
Contractor	2	78.44	12.25%	0.24

The results illustrate that entropy reduction alone is not a good predictor of an attribute's ability to reduce the prediction error. This can clearly be seen from the *Last Name* attribute which has the smallest total permission entropy but the highest order, but only reduces prediction error by 8.5%. These experiments do show that attributes with smaller total entropy and lower cardinality have better predictive performance than the attributes they dominate. For example, *Manager* and *Depart-*

ment have statistically significant better predictive performance compared to *Title* (see Figure 5.7(a)) and while *Organization* has better predictive performance than *Level* (see Figure 5.7(b)), it is not statistically significant. Overall, the *Organization* and *Department* attributes are the most applicable attributes for the **anonymous** dataset.

5.5.2 Evaluating Role Quality

We measure how noise causes candidate roles to overfit the data using the distance measure from Section 4.5 on several synthetic and real datasets. Varying amounts of noise are added to the datasets using the method described in Section 5.5. Roles are mined from the clean, noisy, and cleaned data. For simplicity, SVD is used for cleaning. The distance between roles mined from clean and noisy data is used to measure how overfit the mined roles are. The process is repeated between roles mined from clean and cleaned data to evaluate how well cleaning stabilizes candidate roles. If the former distance is great, then noise causes overfit roles.



(a) (b)

Figure 5.7. The MAE for the user-permission relation X given several attributes.

Noise can have a significant impact on the number of candidate roles, causing the unmatched roles to dominate the distance measure. Instead, the top k candidate roles are selected (by the reduction in weighted structural complexity using the Class 2 to Class 1 prioritization method) such that all sets of roles are the same size.

A summary of the results from our experiments is shown in Table 5.19. *HierarchicalMiner* is used with weight vector $W = \langle 1, 1, 1, 1, 1$ which allows exceptions via direct user-permission assignments, and *HPr* [19], a solution to the role mining problem. The number of roles generated when mining the clean data, the number of candidate roles when mining the noisy and cleaned data, and their dissimilarity from the roles mined from clean data is provided. Unmatched roles are not penalized.

It can be observed that noise causes the number of roles generated from each role mining criteria to increase, often significantly. This increase is more apparent when attempting to mine roles that represent an exact cover of the user-permission relation, as *HPr* does. For example, the **Random** dataset with 4% noise added causes an almost 15-time increase in the number of candidate roles.

The second observation from Table 5.19 is that the candidate roles generated by *HPr* are more unstable than roles generated by *HierarchicalMiner*. This can be seen by comparing the dissimilarity between noisy and clean roles for each algorithm. The dissimilarity between clean and noisy candidate roles is 2.5-time greater for *HPr* than *HierarchicalMiner*, and the average Jaccard distance for each matching pair of roles is 0.16 for *HierarchicalMiner* and 0.50 for *HPr*. This implies the matching roles only share half of the permissions on average. This analysis supports the hypothesis that solutions to the role minimization problem overfit the roles to the user-permission relation, leading to an unstable RBAC state.

5.6 Role Similarity

This section uses the distance measure between sets of candidate roles to evaluate how similar the roles generated from *HierarchicalMiner* are compared to other algo-

Table 5.19
The distance between roles mined from clean data with roles mined from noisy and cleaned data and the effectiveness of noise removal at stabilizing candidate roles and the susceptibility of RMP.

		HM					*HPr*				
		Clean	Noisy		Cleaned		Clean	Noisy		Cleaned	
Data	Noise	$\|R\|$	Diss.	$\|R\|$	Diss.	$\|R\|$	$\|R\|$	Diss.	$\|R\|$	Diss.	$\|R\|$
ERBAC	1%	74	16.22	67	15.39	50	80	19.56	145	15.76	56
	2%	72	18.80	82	15.41	70	80	23.50	230	18.65	49
Random	1%	43	2.62	78	0.00	44	40	23.43	240	5.70	47
	2%	43	3.74	105	1.11	40	40	24.58	296	7.49	56
	3%	44	7.03	128	1.00	55	40	19.03	506	11.51	72
	4%	43	9.28	113	0.53	62	40	15.80	595	12.78	84
University	1%	18	1.20	19	0.00	18	17	11.21	45	7.42	21
	2%	18	0.80	21	0.58	16	17	9.37	58	5.66	20
	5%	18	0.79	38	0.00	16	17	11.25	62	5.81	17
	10%	19	2.90	56	0.53	14	17	7.70	67	2.68	8
Domino	15%	11	3.20	12	2.20	11	20	12.09	66	0.10	15
Firewall	1%	48	14.77	59	8.81	53	69	26.72	94	13.59	68
Healthcare	8%	11	2.40	17	2.42	11	15	8.43	38	5.48	12

rithms from the literature. The total distance between the sets of candidate roles is given in Table 5.20 and the normalized distance for each matched candidate role (unmatched roles, or role matching an empty role are discarded), is given in Table 5.21. From this analysis it can be seen that *HierarchicalMiner* produces sets of candidate roles that more closely resembles the *HPe* algorithm, however the individual matched roles are closer to *HPr*. This is expected; all initial candidate roles for *HPr* are bicliques (concepts), which are processed such that no role is the superset of another. The *HPe* algorithm further processes (permutes) these roles to reduce *UA* and *PA* assignments. The main difference between affecting the distance between these two RBAC states is the number of unmatched roles.

Table 5.20
Distance between *HierarchicalMiner* and candidate roles from other algorithms from the literature.

	HM	*HPr*		*HPe*		*GO*	
Source	$\mid R \mid$	$\mid R \mid$	$D(HM \parallel \rangle HPr)$	$\mid R \mid$	$D(HM \parallel \rangle HPe)$	$\mid R \mid$	$D(HM \parallel \rangle GO)$
APJ	497	455	119.871	485	68.065	584	134.111
Domino	33	20	16.682	34	12.210	30	13.382
EMEA	151	34	124.165	190	130.153	173	106.090
Firewall 1	104	65	52.664	91	44.679	114	49.306
Firewall 2	12	10	5.006	15	6.433	16	5.794
Healthcare	18	14	12.610	19	12.613	21	9.018
Average	135.8	99.7	55.2	139.0	45.7	156.3	53.0

Table 5.21
Normalized distance for matched role only.

Source	HPr	HPe	GO
APJ	0.171	0.115	0.095
Domino	0.184	0.339	0.346
EMEA	0.210	0.603	0.557
Firewall1	0.210	0.348	0.378
Firewall2	0.300	0.286	0.150
Healthcare	0.615	0.645	0.334
Average	0.282	0.390	0.310

5.7 Implementation

This section describes the implementations of *HierarchicalMiner* and *AttributeMiner* and the hybrid role mining algorithms described in this chapter.

The *HierarchicalMiner* has been implemented by separating the formal concept analysis and candidate role optimization phases. There is extensive research is fast and efficient methods to compute the set of formal concepts and corresponding lattice [63, 76–79]. The implementation used in the experiments in this dissertation use the C program[7] *Colibri* [76] by Christian Lindig and an updated Java version[8] by Daniel Götzmann for generating concepts. The running time of *Colibri* is linear in the size of the concept lattice [76]. The *HierarchicalMiner* heuristic algorithm to parse, prune, and restructure the formal concept lattice was written in Python. The *AttributeMiner* code was written in Java. The remaining role mining algorithms, role and edge minimization [19], graph optimization [8], RBAM [10], *FastMiner* and *CompleteMiner* [5] are implemented in Python. The ORCA [4] algorithm was implemented in C++. All matrix factorization and noise removal and detection algorithms are implemented in Matlab.

5.8 Practicality

This chapter has illustrated the advantages of formal concept analysis for many role mining problems, and the performance of the *HierarchicalMiner* algorithm. This section evaluates how efficient this approach is. Formal concept analysis can be expensive, in the worst case, producing 2^n concepts where $n = \min(|\,U\,|, |\,P\,|)$. In practice, the number of concepts is strongly dependent on the density of the UP relation, $\frac{|\,UP\,|}{|\,U\,|\times|\,P\,|}$ [80], and the running time is linear in the size of the concept lattice [76]. When the relation is sparse (as it often is in access control systems) concept analysis is often efficient [80]. Table 5.22 provides a comparison of the time required to perform

[7] `http://code.google.com/p/colibri-concepts/`
[8] `http://code.google.com/p/colibri-java/`

formal concept analysis on real-world access control configurations from [19] to the time required to solve the exact role mining problem ($W = \langle 1, 0, 0, 0, \infty \rangle$) and the fast heuristic algorithm *HPr*. The total running time of *HierarchicalMiner* is divided into the time it takes to calculate the formal concept lattice (FCA) and the greedy optimization (*HM*). Formal concept analysis is implemented in C, and *HierarchicalMiner* in Python. All code is running on a 2 GHz Intel Core 2 Duo with 2GB RAM. Note that the formal concept lattice only needs to be calculated once, and can be optimized after with a variety of weight vectors. The running time for the formal concept analysis uses the C version of *Colibri* described in the previous section and may be decreased by using a more efficient formal concept analysis algorithm [63, 77, 78].

It can be concluded from Table 5.22 that this approach is practical in practice, especially when compared to the cost of manual, top-down role engineering on similar sized datasets that takes months. Role mining is an infrequent and non-interactive task, and the final results still require manual verification and approval from a system administrator (a much larger bottleneck). The approach described in this dissertation is practical when compared to the fast heuristic approaches for role minimization and edge concentration [19]. For illustration purposes the running time for their algorithms are also presented, taken from [19]. Their algorithms are implemented in Matlab and running on a 3GHz Xeon processor with 2GB RAM. While a direct comparison is difficult, once the formal concept lattice is generated the *HierarchicalMiner* algorithm is competitive with their approaches.

Table 5.22
Comparing the time required to perform formal concept analysis (FCA) and optimize the lattice with *HierarchicalMiner* (HM) to algorithms by Ene et al. Exact finds the minimum number of roles. HPr performs a heuristic role minimization and HPe performs heuristic edge concentration. All times given in seconds.

Data	$\lvert U \rvert$	$\lvert P \rvert$	Density	Concepts	FCA	HM	Exact	HPr	HPe
Americas L	3485	10127	0.01	36991	591.64	132.28	1564	69.8	177
Americas S	3477	1587	0.02	2764	17.3	0.644	412	10.2	13.1
APJ	2044	1164	< 0.01	798	1.66	0.220	39	11.1	12.0
EMEA	35	3046	0.07	780	0.07	0.983	0.68	0.04	0.74
Healthcare	46	46	0.70	31	0.008	0.620	0.02	0.05	0.05
Domino	79	231	0.04	73	0.008	0.618	0.03	0.01	0.05
Firewall 1	365	709	0.12	317	0.3	1.064	3	0.76	1.02
Firewall 2	325	590	0.19	22	0.13	0.822	2	0.12	0.16
Average					76.39	19.6	252.59	11.51	25.515

6 CONCLUSIONS AND FUTURE WORK

This dissertation presented an approach to role engineering that produces compact and semantically meaningful roles that is resilient to noise in the input data.

First, it defined Weighted Structural Complexity (WSC), a complexity measure that models the human costs of administrating a role-based access control state and is consistent with the motivation behind RBAC. The problem of minimizing the WSC of an RBAC state is formalized and shown to be trivial to solve for some input values and **NP**-complete in general and for inputs of interest. Next, formal concept analysis is presented as a formal framework on which to base many role mining optimization problems. Formal concepts are a natural way to capture hierarchical structure in binary relations and the formal concept lattice produces a valid complete RBAC state with desirable properties. A role mining algorithm, *HierarchicalMiner*, is proposed to optimize the formal concept lattice with respect to the weighted structural complexity. Evaluations indicate it produces compact results compared to other algorithms in the literature at a wide variety of role mining objective.

To further reduce administrative costs, we argue that roles should be semantically meaningful. While semantic meaning is up to the interpretation of the system administrator, we argue roles that can be defined by user attributes are semantically meaningful. An extension to the *HierarchicalMiner* algorithm, called *AttributeMiner*, is proposed. It produces semantically meaningful roles while reducing the weighted structural complexity of the RBAC state.

Finally, real-world data is often noisy and contains errors. The problem of mining RBAC states given noise is presented and two types of noise are considered. The first type of noise is correctness noise, Type I and Type II errors that impact the security and availability of the resources being protected. The second type of noise is applicability noise. These are assignments that are exceptions to the RBAC model

of access control, and depict exceptional or temporary access to resources outside the job duties of the users. Techniques based on rank reduced matrix decomposition are presented that correct for both types of noise with low false positive and false negative rates. When user-attributes are available, false positive and false negatives are further reduced.

The proposed methods for handling noisy input data can be employed for tasks of prediction of unknown values. When combined with candidate role generation, this allows a complete RBAC state to be generated from incomplete and noisy data. This significantly reduces the costs of role engineering.

This dissertation presented a framework and solutions for general role mining problems. These solutions all consider the existing access control data is static, and this dissertation uses a snapshot of the current access control state. Temporal data, indicating which permissions users use and *when* are useful for many new role mining tasks. For example, future research on role mining should consider how permissions are used together such that they can be clustered and assigned to a single role. Permissions that are not temporally correlated should be split into two-or-more roles to satisfy the principle of least privilege. Further, such workflow patterns may be useful to analyze dynamic constraints, preventing a user from activating roles with dynamically mutually exclusive permissions.

The role mining problems considered in this dissertation can be viewed as a concrete instantiation of a general security policy learning problem to approximate the Oracle Policy using a role-based access control policy. Future work should consider other access control models that may be more representative of the Oracle Policy. A goal of such security policy learning problems would be to a security policy for other access control models, such as the Bell-La Padula model [29], or a general meta model for access control [81, 82].

LIST OF REFERENCES

LIST OF REFERENCES

[1] Michael P. Gallaher, Alan C. O'Connor, and Brian Kropp. The economic impact of role-based access control. Technical Report PTI Project Number 07007.012, National Institute of Standards and Technology, March 2002.

[2] Axel Buecker, Jaime Cordoba Palacios, Brian Davis, Todd Hastings, and Ian Yip. Identity management design guide with IBM Tivoli Identity Manager, November 2005.

[3] Martin Kuhlmann, Dalia Shohat, and Gerhard Schimpf. Role mining — Revealing business roles for security administration using data mining technology. In *Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies, SACMAT '03*, pages 179–186, 2003.

[4] Jürgen Schlegelmilch and Ulrike Steffens. Role mining with ORCA. In *Proceedings of the tenth ACM symposium on Access control models and technologies, SACMAT '05*, pages 168–176, 2005.

[5] Jaideep Vaidya, Vijayalakshmi Atluri, and Janice Warner. Roleminer: Mining roles using subset enumeration. *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 144–153, 2006.

[6] Jaideep Vaidya, Vijayalakshmi Atluri, and Qi Guo. The role mining problem: Finding a minimal descriptive set of roles. In Lotz and Thuraisingham [83], pages 175–184.

[7] Haibing Lu, Jaideep Vaidya, and Vijayalakshmi Atluri. Optimal boolean matrix decomposition: Application to role engineering. In *International Conference on Data Engineering Conference*, pages 297–306. IEEE, 2008.

[8] Dana Zhang, Kotagiri Ramamohanarao, and Tim Ebringer. Role engineering using graph optimisation. In Lotz and Thuraisingham [83], pages 139–144.

[9] Dana Zhang, Kotagiri Ramamohanarao, Tim Ebringer, and Trevor Yann. Permission set mining: Discovering practical and useful roles. In *Computer Security Applications Conference, 2008. ACSAC 2008. Annual*, pages 247–256, December 2008.

[10] Alessandro Colantonio, Roberto Di Pietro, and Alberto Ocello. A cost-driven approach to role engineering. In Roger L. Wainwright and Hisham Haddad, editors, *Symposium On Applied Computing, SAC'08*, pages 2129–2136. ACM, 2008.

[11] Alessandro Colantonio, Roberto Di Pietro, and Alberto Ocello. Leveraging lattices to improve role mining. In Sushil Jajodia, Pierangela Samarati, and Stelvio Cimato, editors, *Proceedings of The IFIP TC-11 23rd International Information*

Security Conference, IFIP 20th World Computer Congress, IFIP SEC 2008, September 7-10, 2008, Milano, Italy Tc 11 23rd International Information Security Conference, volume 278 of *IFIP*, pages 333–347. Springer, 2008.

[12] Alessandro Colantonio, Roberto Di Pietro, Alberto Ocello, and Nino Vincenzo Verde. A formal framework to elicit roles with business meaning in RBAC systems. In *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies, SACMAT'09*, pages 85–94, 2009.

[13] Mario Frank, David A. Basin, and Joachim M. Buhmann. A class of probabilistic models for role engineering. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM Conference on Computer and Communications Security*, pages 299–310. ACM, 2008.

[14] Mario Frank, Andreas P. Streich, David Basin, and Joachim M. Buhmann. A probabilistic approach to hybrid role mining. In *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security*, pages 101–111, 2009.

[15] Andreas P. Streich, Mario Frank, David Basin, and Joachim M. Buhmann. Multi-assignment clustering for boolean data. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09*, pages 969–976, 2009.

[16] Ian Molloy, Hong Chen, Tiancheng Li, Qihua Wang, Ninghui Li, Elisa Bertino, Seraphin B. Calo, and Jorge Lobo. Mining roles with semantic meanings. In *Proceedings of the 12th ACM Symposium on Access Control Models and Technologies, SACMAT'08*, pages 21–30, 2008.

[17] Jaideep Vaidya, Vijayalakshmi Atluri, Qi Guo, and Nabil Adam. Migrating to optimal rbac with minimal perturbation. In *Proceedings of the 13th ACM symposium on Access control models and technologies, SACMAT '08*, pages 11–20, 2008.

[18] Qi Guo, Jaideep Vaidya, and Vijayalakshmi Atluri. The role hierarchy mining problem: Discovery of optimal role hierarchies. In *In Proceedings of the 24th Annual Computer Security Applications Conference (ACSAC 2008)*, 2008.

[19] Alina Ene, William Horne, Nikola Milosavljevic, Prasad Rao, Robert Schreiber, and Robert Endre Tarjan. Fast exact and heuristic methods for role minimization problems. In Indrakshi Ray and Ninghui Li, editors, *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies, SACMAT'08*, pages 1–10. ACM, 2008.

[20] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, 1998.

[21] Daniel Lee and H Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, Oct 1999.

[22] Mikkel N. Schmidt, Ole Winther, and Lars Kai Hansen. Bayesian non-negative matrix factorization. In *Independent Component Analysis and Signal Separation, International Conference on*, volume 5441 of *Lecture Notes in Computer Science (LNCS)*, pages 540–547. Springer, 2009.

[23] Andrew I. Schein, Lawrence K. Saul, and Lyle H. Ungar. A generalized linear model for principal component analysis of binary data. In *Proceedings of the 9'th International Workshop on Artificial Intelligence and Statistics*, 2003.

[24] Ajit Paul Singh and Geoffrey J. Gordon. Relational learning via collective matrix factorization. In Ying Li, Bing Liu, and Sunita Sarawagi, editors, *Knowledge Discovery and Data Mining, KDD'08*, pages 650–658. ACM, 2008.

[25] Butler W. Lampson. Protection. In *Proceedings of the 5th Princeton Conference on Information Sciences and Systems*, 1971.

[26] Michael A. Harrison, Walter L. Ruzzo, and Jeffrey D. Ullman. Protection in operating systems. *Communications of the ACM*, 19(8):461–471, 1976.

[27] Matt Bishop, Sophie Engle, Sean Peisert, Sean Whalen, and Carrie Gates. We have met the enemy and he is us. In *Proceedings of the Fifteenth New Security Paradigms Workshop, NSPW'08*, 2008.

[28] James Joshi, Elisa Bertino, Usman Latif, and Arif Ghafoor. A generalized temporal role-based access control model. *IEEE Transactions on Knowledge and Data Engineering*, 17(1):4–23, 2005.

[29] D. E. Bell and L. J. La Padula. Secure computer system: Unified exposition and Multics interpretation. Technical report, The Mitre Corporation, 1976.

[30] D. Brewer and M. Nash. The chinese wall security policy. *In Proceedings of the IEEE Symposium on Security and Privacy*, pages 206–214, 1989.

[31] David D. Clark and David R. Wilson. A comparison of commercial and military computer security policies. *IEEE Symposium on Security and Privacy*, pages 184–194, 1987.

[32] Robert W. Baldwin. Naming and grouping privileges to simplify security management in large databases. *IEEE Symposium on Security and Privacy*, 1990.

[33] David Ferraiolo and Richard Kuhn. Role-based access controls. In *In 15th NIST-NCSC National Computer Security Conference*, pages 554–563, 1992.

[34] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[35] ANSI. Role-based access control. Technical Report ANSI INCITS 359-2004, American National Standard for Information Technology, 2004.

[36] Ravi Sandhu and Qamar Munawer. How to do discretionary access control using roles. In *Proceedings of the third ACM workshop on Role-based access control, RBAC '98*, pages 47–54, 1998.

[37] Sylvia Osborn, Ravi Sandhu, and Qamar Munawer. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Transactions on Information and System Security (TISSEC)*, 3(2):85–106, 2000.

[38] Ninghui Li, Ji-Wom Byun, and Elisa Bertino. A critique of the ANSI standard on role based access control. 2005.

[39] Trent Jaeger and Jonathon E. Tidswell. Rebuttal to the nist rbac model proposal. In *Proceedings of the fifth ACM workshop on Role-based access control, RBAC '00*, pages 65–66, 2000.

[40] Elisa Bertino, Piero A. Bonatti, and Elena Ferrari. TRBAC: A temporal role-based access control model. *ACM Transactions on Information System Security*, 4(3):191–223, 2001.

[41] Elisa Bertino, Barbara Catania, Maria Luisa Damiani, and Paolo Perlasca. GEO-RBAC: a spatially aware RBAC. In *Proceedings of the tenth ACM symposium on Access control models and technologies, SACMAT '05*, pages 29–37, 2005.

[42] Sudip Chakraborty and Indrajit Ray. TrustBAC — integrating trust relationships into the RBAC model for access control in open systems. *Proceedings of the eleventh ACM symposium on Access control models and technologies, SACMAT '06*, 2006.

[43] Maria Luisa Damiani, Elisa Bertino, Barbara Catania, and Paolo Perlasca. GEO-RBAC: A spatially aware RBAC. *ACM Transactions on Information System Security*, 00(00):1–34, 2006.

[44] Edward J. Coyne. Role engineering. In *Proceedings of the first ACM Workshop on Role-based access control, RBAC '95*, page 4, 1995.

[45] E. B. Fernandez and J. C. Hawkins. Determining role rights from use cases. In *ACM Workshop on Role-Based Access Control*, pages 121–125, 1997.

[46] P. Epstein and Ravi Sandhu. Engineering of role/permission assignments. In *Proceedings of the 17th Annual Computer Security Applications Conferences, ACSAC'01*, volume 10, pages 127–136, 2001.

[47] Per Brinch Hansen. *Operating System Principles*. Prentice-Hall, 1973.

[48] David J. Hand, Heikki Mannila, and Padhraic Smyth. *Principles of Data Mining*. The MIT Press, 2001.

[49] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. pages 487–499, 1994.

[50] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8(1):53–87, 2004.

[51] Qun Ni, Jorge Lobo, Seraphin Calo, Pankaj Rohatgi, and Elisa Bertino. Automating role-based provisioning by learning from examples. In *Proceedings of the 14th ACM symposium on Access control models and technologies, SACMAT '09*, pages 75–84, 2009.

[52] Ian Molloy, Hong Chen, Tiancheng Li, Qihua Wang, Ninghui Li, Elisa Bertino, Seraphin Calo, and Jorge Lobo. Mining roles with multiple objectives. In *Transactions on Information and System Security (TISSEC)*, In Submission.

[53] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 2009.

[54] Axel Kern. Advanced features for enterprise-wide role-based access control. In *Annual Computer Security Applications Conference, ACSAC'02*, pages 333–342. IEEE Computer Society, 2002.

[55] Axel Kern, Martin Kuhlmann, Andreas Schaad, and Jonathan D. Moffett. Observations on the role life-cycle in the context of enterprise security management. In *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies, SACMAT'02*, pages 43–51, 2002.

[56] Charles Kemp, Joshua B. Tenenbaum, Thomas L. Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. In *In Proceedings of the 21st National Conference on Artificial Intelligence*, 2006.

[57] Ian Molloy, Ninghui Li, Jorge Lobo, Yuan (Alan) Qi, and Luke Dickens. Mining roles with noisy data. In *SACMAT 2010, 14th ACM Symposium on Access Control Models and Technologies*, 2010.

[58] Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

[59] Xuemin Lin. One the computational complexity of edge concentration. In *Discrete Applied Mathematics*, volume 101, pages 197–205, 2000.

[60] Alina Ene. Biclique covers of bipartite graphs: The minimum biclique cover and edge concentration problems. Technical report, Princeton University, 2007.

[61] Milind Dawande, Pinar Keskinocak, Jayashankar M. Swaminathan, and Sridhar Tayur. On bipartite and multipartite clique problems. *J. Algorithms*, 41(2):388–403, 2001.

[62] James Abello, Alex J. Pogel, and Lance Miller. Breadth first search graph partitions and concept lattices. *Journal of Universal Computer Science*, 10(8):934–954, 2004.

[63] Gerd Stumme, Rafik Taouil, Yves Bastide, Nicolas Pasquier, and Lotfi Lakhal. Computing iceberg concept lattices with Titanic. *Data and Knowledge Engineering*, 42(2):189–222, 2002.

[64] Ajit Paul Singh and Geoffrey J. Gordon. A unified view of matrix factorization models. In Walter Daelemans, Bart Goethals, and Katharina Morik, editors, *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML/PKDD*, volume 5212 of *Lecture Notes in Computer Science*, pages 358–373. Springer, 2008.

[65] Mehmet Koyutürk, Ananth Grama, and Naren Ramakrishnan. Nonorthogonal decomposition of binary matrices for bounded-error data compression and analysis. *ACM Transactions Mathematical Software*, 32(1):33–69, 2006.

[66] Lujo Bauer, Scott Garriss, and Michael K. Reiter. Detecting and resolving policy misconfigurations in access-control systems. In *Proceedings of the 13th ACM symposium on Access control models and technologies, SACMAT '08*, pages 185–194, 2008.

[67] Ian Molloy, Ninghui Li, Tiancheng Li, Ziqing Mao, Qihua Wang, and Jorge Lobo. Evaluating role mining algorithms. In *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies, SACMAT'09*, 2009.

[68] Michael E. Wall, Andreas Rechtsteiner, and Luis M. Rocha. *Singular value decomposition and principal component analysis*, chapter 5, pages 91–109. Kluwer, 2003.

[69] Pau-Chen Cheng, Pankaj Rohatgi, Claudia Keser, Paul A. Karger, Grant M. Wagner, and Angela Schuett Reninger. Fuzzy MLS: An experiment on quantified risk-adaptive access control. *IEEE Symposium on Security and Privacy 2007*, 2007.

[70] Alessandro Colantonio, Roberto Di Pietro, Alberto Ocello, and Nino Vincenzo Verde. Mining stable roles in RBAC. In *IFIP Advances in Information and Communication Technology*, 2009.

[71] Marc Bezem, Karnik Blok, and Maarten Keijzer. Metrics for classifying heterogeneous objects. Technical Report PNA-R9804, Stichting Mathematisch Centrum, May 31 1998.

[72] Harold W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(83–97), 1955.

[73] Justus Schwartz, Angelika Steger, and Andreas Weißl. Fast algorithms for weighted bipartite matching. *Experimental and Efficient Algorithms*, pages 476–487, 2005.

[74] Scott D. Stoller, Ping Yang, C. R. Ramakrishnan, and Mikhail I. Gofman. Efficient policy analysis for administrative role based access control. October 2007.

[75] Axel Kern, Andreas Schaad, and Jonathan Moffett. An administration concept for the enterprise role-based access control model. pages 3–11, June 2003.

[76] Christian Lindig. Fast concept analysis. In Gerhard Stumme, editor, *Working with Conceptual Structures - Contributions to ICCS 2000*, 2000.

[77] Petr Krajca, Jan Outrata, and Vilem Vychodil. Parallel recursive algorithm for FCA. In *Concept Lattices and Their Applications*, 2008.

[78] Simon Andrews. In-Close, a fast algorithm for computing formal concepts. In *17TH International Conference on Conceptual Structures (ICCS'09)*, 2009.

[79] Anne Berry, Jean Paul Bordat, and Alain Sigayret. A local approach to concept generation. *Annals Mathethematics and Artificial Intelligence*, 49(1-4):117–136, 2007.

[80] Christian Lindig. Mining patterns and violations using concept analysis. Technical report, Universität des Saarlandes, Saarbrücken, Germany, June 2007.

[81] David Ferraiolo and Vijay Atluri. A meta model for access control: why is it needed and is it even possible to achieve? In *Proceedings of the 13th ACM symposium on Access control models and technologies, SACMAT '08*, pages 153–154, 2008.

[82] Steve Barker. The next 700 access control models or a unifying meta-model? In *Proceedings of the 14th ACM symposium on Access control models and technologies, SACMAT '09*, pages 187–196, 2009.

[83] Volkmar Lotz and Bhavani M. Thuraisingham, editors. *SACMAT 2007, 12th ACM Symposium on Access Control Models and Technologies, Sophia Antipolis, France, June 20-22, 2007, Proceedings*. ACM, 2007.

APPENDIX

A ADDITIONAL PROOFS

A.1 Trivial Solutions to WSCP

There are several weight combinations that yield trivial solutions to the weighted structural complexity problem. A summary a given in Table 3.2. Here we describe these trivial solutions. For example, assume that both w_r and w_u are 0 and $w_d = \infty$, which indicates that it costs nothing to create a role or assign a role to a user. In this case, given $\langle U, P, UP$, an RBAC system $\gamma\sigma$that minimizes the value of WSC is the one that creates a role r_i for every permission $p_i \in\rangle P$, and $(u_j, r_i) \in\rangle UA$ if and only if $(u_j, p_i) \in\rangle UP$. We have $wsc(\gamma, W) = w_p *\rangle|P|$, which is minimum as every permission in P must be assigned to at least one role so as to make the RBAC system effective. Similarly, when both w_r and w_p are 0 and $w_d = \infty$, an RBAC system that trivially minimizes WSC is the one that creates a role r_i for every user u_i such that $(p_j, r_i) \in\rangle PA$ if and only if $(u_i, p_j) \in\rangle UP$.

When setting $w_u = c_u$, $w_p = c_p$, $w_r = w_h = 0$, and $w_d = \infty$, the minimal RBAC state is a quadripartite graph using two levels of roles—user roles and permission roles—by creating a role for each user and each permission. The UP relation is recreated by assigning permission roles to user roles. We have $wsc(\gamma, W) = c_u *\rangle$ $|\, U \,| + c_p *\rangle| \, P \,|$, which is the minimum as every permission and every user must be assigned to at least one role.

Next, we consider the case when $w_r = w_u = x$, $w_h = c_h$, $w_p = 0$, and $w_d = \infty$. Observe that it costs nothing to add permissions to a role, and a role assignment costs the same as creating a new role. Assume there exists a role r that completely covers a user u and we would like to see if there is a less costly assignment that doesn't include r. At a minimum, we would need to assign u at least two other roles, which has the same or higher cost then creating the role that exactly covers u; the

complexity will be a minimum when each user is assigned a single role that exactly covers it. Next observe that creating a role hierarchy cannot decrease the cost of the RBAC state since it can only decrease the number of permission assignments (at a cost of c_h per assignment) and permission assignments incur zero cost. Thus $wsc(\gamma, W) = w_r * |\, U' \,| + w_u * |\, U \,|$, where U' is the set of unique users such that no two users are assigned the exact same set of permissions[1]. Similarly, when $w_r = w_p = x$, $w_h = c_h$, $w_u = 0$, and $w_d = \infty$, we can create a new role for each unique permission set.

A.2 Lemma 1 Redux: *HierarchicalMiner*

Because *HierarchicalMiner* is a heuristic algorithm, and does not exactly solve the WSCO problem, we cannot rely on Lemma 1 to ensure role creation. Instead, we must evaluate how *HierarchicalMiner* will process each role given the four pruning rules to determine the sufficient condition to ensure the roles are not pruned. This results in the following:

Lemma 2 Given weights $\langle w_r, w_u, w_p, w_h, w_d$ such that $w_d \geq\rangle w_u > 0$ and $w_h < \infty$, a configuration $\rho\sigma = \langle U, P, UP$, and $P_1 \quad P$ where $|P_1| \geq\rangle 2$. Let U_1 be the set of users whose set of permissions is exactly P_1 in ρ. If

$$|U_1| > \frac{(w_r + w_h \,|\, P \,|)}{w_u},$$

then *HierarchicalMiner* will output an RBAC system for $\rho\sigma$that will include a role that is assigned exactly permissions in P_1.

Proof First, a user u is assigned to a single role r in the factored formal concept lattice, the role containing the exact permission set they are assigned, i.e., $\mathcal{P}_u \equiv\rangle \mathcal{P}_r$. Therefore we can eliminate the need to evaluate Case 1 and Case 3, because they assume a role has zero users directly assigned. We begin with Case 2.

[1]This ensures we do not have any redundant roles.

Case 2 states that a role will be pruned when

$$w_r + w_u * \rangle n + w_h * \langle (\mid Sen(r) \mid + \mid Jun(r) \mid) \geq \rangle w_u * \rangle n * \rangle \mid Jun(r) \mid + w_h * \rangle \mid Thr(r) \mid .$$

Thus, solving for n, the number of users assigned to the role, the rule will never prune the role when

$$\frac{w_r + w_h * \langle (\mid Sen(r) \mid + \mid Jun(r) \mid - \rangle \mid Thr(r) \mid)}{w_u * \langle (\mid Jun(r) \mid - \rangle 1)} < n.$$

For Case 2, a role must have two or more junior roles (otherwise it must be directly assigned an additional permission, violating the definition of a formal concept, or falling under Case 4), and the solution is well defined. Next, each junior or senior role must be different by at least one permission, and the total number of junior and senior roles $\mid Sen(r) \mid + \mid Jun(r) \mid < \mid P \mid$. Ignoring the through edges, we have that

$$\frac{w_r + w_h * \rangle \mid P \mid}{w_u} < n_2.$$

Next, we consider Case 4. There are two sub-cases. First, a role is removed from the lattice, affecting its set of authorized permissions when

$$w_h * \mid Jun(r) \mid + w_h * \mid Sen(r) \mid \geq \rangle w_h * \mid Thr(r) \mid + w_u * n * \mid Jun(r) \mid + w_p * m * \mid Sen(r) \mid .$$

Solving for n, we have

$$\frac{w_h * \langle (\mid Sen(r) \mid + \mid Jun(r) \mid - \rangle \mid Thr(r) \mid) - \rangle w_p * \rangle m * \rangle \mid Sen(r) \mid}{w_u * \rangle \mid Jun(r) \mid} < n_{4a}.$$

As above, we can simplify this as

$$\frac{w_h * \rangle \mid P \mid - \rangle w_p * \rangle m * \rangle \mid Sen(r) \mid}{w_u * \rangle \mid Jun(r) \mid} < n_{4a}.$$

And the role is prune completely, directly assigning the permissions to the users, when

$$\begin{gathered} w_r + w_u * \rangle n + w_p * \rangle m + w_h * \rangle \mid Jun(r) \mid + w_h * \rangle \mid Sen(r) \mid \\ \geq \rangle w_h * \rangle \mid Thr(r) \mid + w_u * \rangle n * \rangle \mid Jun(r) \mid + w_p * \rangle m * \rangle \mid Sen(r) \mid + w_d * \rangle m * \rangle n. \end{gathered}$$

Solving for n, we have

$$\frac{w_r - \rangle w_p * \rangle m * \langle (\mid Sen(r) \mid - \rangle 1) + w_h * \langle (\mid Sen(r) \mid + \mid Jun(r) \mid - \rangle \mid Thr(r) \mid)}{w_u * \langle (\mid Jun(r) \mid - \rangle 1) + w_d * \rangle m} < n_{4b},$$

and

$$\frac{w_r + w_h * \rangle | \, P \, | - \rangle w_p * \rangle m * \rangle (| \, Sen(r) \, | - \rangle 1)}{w_u * \rangle (| \, Jun(r) \, | - \rangle 1) + w_d * \rangle m} < n_{4b}.$$

It is clear that $n_2 \geq \rangle n_{4*}$, and n_2 is a sufficient condition to ensure a role is not pruned.

■

Note that when $w_h = \infty$, we cannot guarantee a role will not be pruned. This is a result of *HierarchicalMiner* being a heuristic algorithm, and the four cases not being exhaustive. One could extend *HierarchicalMiner* in such a way that Lemma 1 holds, however this increases the number of sub cases. For example, in Case 2, the role may be removed from the hierarchy completely and directly assigned all permissions, without having the demote users down the lattice.

A.3 Role Stability

In [70], Colantonio et al. consider the problem of mining stable roles, however they do not provide a formal definition for a stable role. Informally they suggest that a role is unstable if the introduction or removal of new users, permissions, or user-permission assignments could change the optimal candidate role set. Next, they define a weight function for a role as either the weighted sum of the users and permissions, or a weighted product of the user-permission relations it captures[2]. For a role r, let U_r be the set of users "associated" with r, and P_r be the set of permissions "associated" to r. For simplicity, assume associated means the set of users that are assigned a superset of P_r in UA. Given a vector $W = \langle w_r, w_u, w_p, w_h, w_d \rangle$, the weight of a role r, denoted $w(r)$, is

$$w(r) = w_u \, | \, U_r \, | \oplus \rangle w_p \, | \, P_r \, |$$

where $\oplus \rangle$is distributive over multiplication.

[2]Their notation is somewhat confusing and non-traditional. I believe they are attempting to individually weight the number of users and permissions, but must make changes to math operations to make the operations associative and work within their solution framework.

They imply a role with limited weight, $w(r) < t$ for threshold t, is unstable, and suggest a method to prune user-permission assignments that can only belong to roles where $w(r) < t$.

Their definition of an instability by role weight does not match their implicit definition of an unstable role. There are many "unstable" roles that their method may still propose. Consider three roles, $r_1 < r_2 < r_3$ as shown in Figure A.1, and assume $w(r_1) > t$ and $w(r_3) > t$; that is, both r_1 and r_3 are stable. By the definition in [70], r_2 may[3] be considered stable, that is $w(r_2) > t$, yet if we prune a single user-permission assignment, r_2 will no longer be generated. We will assume all candidate roles are maximal bicliques (concepts), as [70] does.

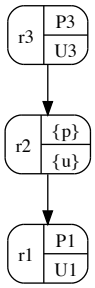


Figure A.1. A small role hierarchy illustrating the incompleteness of defining stable roles by role weight.

Theorem A.3.1 *There exist a role r with sufficient weight $w(r) > t$ that is unstable and is not a formal concept if one user-permission assignment is pruned.*

Proof To be considered stable, $w(r_2) > t$. If r_1, r_2, and r_3 are all concepts, or maximal bicliques, then $|\, U_{r_2} \,| \in \rangle[|\, U_{r_3} \,|+1, |\, U_{r_1} \,|-1]$, and $|\, P_{r_2} \,| \in \rangle[|\, P_{r_1} \,|+1, |\, P_{r_3} \,|-1]$. That is, r_2 must differ from both r_1 and r_3 by at least one user and one permission. We also note that $|\, U_{r_1} \,| \geq \rangle|\, U_{r_3} \,| + 2$, and $|\, P_{r_1} \,| \leq \rangle|\, P_{r_3} \,| - \rangle 2$. Next, let $r = (U_{r_3}, P_{r_1})$

[3]Proving a role must exist is slightly more difficult and I do not believe true.

and assume $w(r) > t$, that is, all roles r' such that $r_1 \leq\rangle r' \leq\rangle r_3$ are stable as defined by [70]. It is trivial to construct such a role.

It is possible that r_2 is not stable by their intuitive definition. Let $(u, p) \in\rangle UP$ where $u \notin\rangle U_{r_3}$ and $p \notin\rangle P_{r_1}$. We can define r_2 where $U_{r_2} = U_{r_3} \cup\rangle\{\, u \,\}$ and $P_{r_2} = P_{r_1} \cup\rangle\{\, p \,\}$. See Figure A.1. Then r_2 is stable, because $w(r_2) > t$, but pruning (u, p) no longer makes r_2 a concept of UP, and the role will not be generated, violating the implicit definition of stability by Colantonio et al.

Let r_2' be the role r_2 without (u, p), that is $U_{r_2'} = U_{r_3}$ and $P_{r_2'} = P_{r_1}$. The intent of r_2' produces r_3, and the extend of r_2' produces role r_1. That is r_2 regresses to produce r_1 and r_3 redundantly in the absence of (u, p). Therefore, if (u, p) is pruned, r_2 will not be generated, and is thus unstable. ■

In the above example we over-assigned the permission p to the user u. We can similarly define an example where we under-assign a permission p to a user u. Imagine two roles, $r_1 < r_2$, where $U_{r_1} = U_{r_2} \cup\rangle\{\, u \,\}$ and $P_{r_2} = P_{r_1} \cup\rangle\{\, p \,\}$. When $w_u = w_p$, then $w(r_1) = w(r_2)$. If user u is removed, then r_1 will no longer be generated since all remaining users U_{r_1} have permissions P_{r_2}.

VITA

VITA

Education

Purdue University, West Lafayette, Indiana USA **August 2010**

Ph.D., Computer Science

- Thesis Topic: Automatic Migration to Role-Based Access Control
- Advisor: Ninghui Li

Syracuse University, Syracuse, New York USA **May 2004**

B.S., Computer Science

- *Magna cum Laude* in Computer Science with Honors
- Minor in Mathematics

Publications

Journal Articles

- I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. Calo, and J. Lobo. Mining Roles with Multiple Objectives. *ACM Transactions on Information and System Security (TISSEC)*, Accepted 2010.
- T. Li, N. Li, J. Zhang, and I. Molloy. Slicing: A New Approach to Privacy Preserving Data Publishing. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, In Review.
- I. Molloy and T. F. Stepinski. Automatic Mapping of Valley Networks on Mars. *Elsevier Computers and Geosciences*, 2007.

Refereed Conference Papers

- I. Molloy, N. Li, J. Lobo, Y. Qi, and L. Dickens. Mining Roles with Noisy Data. *Proceedings of the Fifteenth ACM Symposium on Access Control Models and Technologies (SACMAT'10)*, June 2010.
- N. Ding, R. Xiang, I. Molloy, N. Li, and Y. Qi. Approximate Inference for Nonparametric Bayesian Matrix Factorization. *Journal Machine Learning Research, W&CP 9, Vol. 9. Artificial Intelligence and Statistics (AISTATS'10)*, May 2010.
- I. Molloy, N. Li, and T. Li. On the (In)Security and (Im)Practicality of Outsourcing Precise Association Rule Mining. *IEEE International Conference on Data Mining (ICDM'09)*, December 2009.
- I. Molloy, N. Li, T. Li, Z. Mao, Q. Wang, and J. Lobo. Evaluating Role Mining Algorithms. *Proceedings of the Fourteenth ACM Symposium on Access Control Models and Technologies (SACMAT'09)*, June 2009.
- Z. Mao, N. Li, and I. Molloy. Defeating Cross-Site Request Forgery Attacks with Browser-Enforced Authenticity Protection. *International Conference on Financial Cryptography and Data Security (FC'09)*, February 2009.
- I. Molloy, P.C. Cheng, and P. Rohatgi. Trading in Risk: Using Markets to Improve Access Control. *Proceedings of the Fifteenth New Security Paradigms Workshop (NSPW'08)*, September 2008.
- I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. Calo, and J. Lobo. Mining Roles with Semantic Meanings. *Proceedings of the Thirteenth ACM Symposium on Access Control Models and Technologies (SACMAT'08)*, June 2008.
- I. Molloy, J. Li, and N. Li.Dynamic Virtual Credit Card Numbers. *International Conference on Financial Cryptography Cryptography and Data Security (FC'07)*, February 2007.

Technical Reports

- T. Li, N. Li, J. Zhang, and I. Molloy. Slicing: A New Approach to Privacy Preserving Data Publishing. *arXiv Report:0909.2290*, 2009.
- I. Molloy, N.Li , and T. Li. On the (In)Security and (Im)Practicality of Outsourcing Precise Association Rule Mining. *CERIAS Tech Report*, December 2009.
- W. R. Speirs and I. Molloy. Making Large Hash Functions from Small Compression Functions. *Cryptology ePrint Archive, Report 2007/239*, 2007.

Work Experience

T.J. Watson Research Center, IBM Corporation, Hawthorne, New York USA

Research Co-Op **Fall 2009 to January 2010**

- Continued research on role mining from summer internship

Research Internship - Secure Software and Services **Summer 2009**

- Worked with Jorge Lobo and Pau-Chen Cheng under Seraphin Calo and Josyula Rao
- Applied machine learning and data mining techniques to access control policies
- Automatically infer role-based access control policies and detect policy misconfigurations
- Adapt role-mining tools and techniques for real-world requirements, constraints, and demands

Research Co-Op **Fall 2007 to Fall 2008**

- International Technology Alliance Project on Trust and Risk Management in Dynamic Coalition Environments
- Research topics include risk policies, risk allocation, and policy inference

– Applied fuzzy logic, machine learning, and economic theory to risk-based access control systems for MANETs

Research Internship - Secure Software and Services **Summer 2007**

– Worked under Pau-Chen Cheng and Pankaj Rohatgi

– Simulated and modeled Risk Adaptive Access Control System for IBM System S

– Applied experimental economics to the problem of risk allocation

– Applied economic theory and machine learning techniques to analyze simulation results

Lunar and Planetary Institute, Houston, Texas USA

Visiting Graduate Student **Summer 2005**

– Worked under Tomasz Stepinski

– Designed algorithms to identify valley networks on Martian terrain

– Applied image processing and statistical analysis techniques to laser altimeter data

– Prototyped and implemented algorithms using Mathematica and ArcGIS, Perl and shell scripts, and produced standalone C++ executables

Syracuse University, Syracuse, New York USA

Undergraduate Research Assistant **Spring 2003 to Spring 2004**

Virtual Markets and Wireless Grids (wirelessgrids.net)

* Worked under Lee McKnight

* Designed and implemented demo applications and flexible architecture for grid computing on lightweight devices

∗⟩ Demo applications were written in Java and included distributed Mandelbrot animation, distributed surround sound audio recording, and screen sharing

∗ $A patent application for this work has been filed: 20090068943.

iMint (Intelligent Middleware and Information Networking Technologies) Lab

∗ $Work under Junseok Hwang

∗⟩ Studied, designed, and devised attacks for peer-to-peer reputation systems

NSF Research Experience for Undergrads

∗ $Worked under Junseok Hwang

∗⟩ Studied the problem of collusion and zero-cost identities in peer-to-peer networks and attempted to devise resilient systems and detection mechanisms

Teaching Experience

Purdue University, West Lafayette, Indiana USA

Instructor - Data Structures **Summer 2006**

– Instructor for Data Structures summer session

– Duties included running labs, preparing daily lectures, written and programming assignments, and exams

– Material included object oriented design, introduction to C++, algorithms, searching and sorting, asymptotic analysis, data storage, and other related topics

General Teaching Assistant

- Operating Systems **Fall 2005 to Spring 2007**
  - ∗⟩ Duties included running labs, lecturing, technical demonstrations, and grading
  - ∗⟩ Topics included synchronization mechanisms, scheduling, memory management, systems programming, and other related topics
- Data Structures **Fall 2004 to Spring 2005**
  - ∗ $ Duties included lecturing, technical demonstrations, and grading

Projects

Core-RBAC in MySQL, Purdue University **Fall 2006**

- Implemented Core-RBAC into the open source MySQL DBMS
- Fine grained capabilities with *user@host*-level principals and column-level privileges
- Two person group project written in C++ with relations stored in MySQL tables
- Source code release back to the community

Honors Project, Syracuse University **2002 to 2004**

- Thesis: "Attractors in Real-Valued Circuits" with advisor Howard Blair
- Studied the problem of voltage fluctuation, leakage, tunneling, coupling and other errors in CMOS processors
- Applied chaotic attractors and feedback to eliminate errors and converge on a correct solution

Presentations

- "Evaluating Role Mining Algorithms", *Fourteenth ACM Symposium on Access Control Models and Technologies '09*, Stresa, Italy. June 2009.
- "Defeating Cross-Site Request Forgery Attacks with Browser-Enforced Authenticity Protection", *Thirteenth International Conference on Financial Cryptography and Data Security '09*, Barbados. February 2009.
- "Trading in Risk: Using Markets to Improve Access Control", *Fifteenth New Security Paradigms Workshop '08*, Lake Tahoe, CA, USA. September 2008.
- "Mining Roles with Semantic Meanings", *Thirteenth ACM Symposium on Access Control Models and Technologies '08*, Estes Park, CO, USA. June 2008. "Dynamic Virtual Credit Card Numbers", *Eleventh International Conference on Financial Cryptography and Data Security '07*, Lowlands, Scarborough, Trinidad and Tobago. February 2007.
- Wireless Grid Technical Demonstration. *Virtual Markets in Wireless Grids Project Meeting III Telecom Citys Research & Development Consortium*, Mellon Bank, Everett, MA. January 26, 2004.