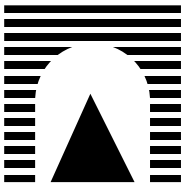


UNIVERSITÀ DEGLI STUDI "ROMA TRE"
FACOLTÀ DI SCIENZE MATEMATICHE FISICHE E NATURALI
XXIV Ciclo del Dottorato di Ricerca in Matematica

THESIS

Role Mining Over Big and Noisy Data Theory and Some Applications

Candidate
Nino Vincenzo VERDE

Advisor
Dr. Roberto DI PIETRO

Coordinator
Prof. Luigi CHIERCHIA

DECEMBER 2011

Abstract

RBAC (*Role-Based Access Control* [2]) is a widely adopted access control model. According to this model, *roles* are created for various job functions within the organization. The permissions required to perform certain operations are assigned to specific roles. System users, in turn, are assigned to appropriate roles based on their responsibilities and qualifications. Through role assignments they acquire the permissions to perform particular system functions. By deploying RBAC systems, organizations obtain several benefits such as simplified access control administration, improved organizational productivity, and security policy enforcement.

Companies that plan to use RBAC model are usually large or medium organizations that are currently using other access control models and/or legacy systems. Despite the benefits related to RBAC, it is sometimes hard for these organizations to adopt such a model. Indeed, there is an important issue that needs to be addressed: the model must be customized to capture the needs and functions of the company. For this purpose, the *role engineering* discipline [21] has been introduced. Various approaches to role engineering have been proposed, which are usually classified as: *top-down* and *bottom-up*. The former requires a deep analysis of business processes to identify which access permissions are necessary to carry out specific tasks. The latter seeks to identify de facto roles embedded in existing access control information. Since bottom-up approaches usually resort to data mining techniques, the term *role mining* is often used as a synonym for bottom-up.

This thesis is devoted to role mining techniques, and their applications to large scale datasets. Several works prove that the role mining problem is reducible to many other well-known $\mathcal{NP}$-hard problems, such as binary matrices factorization [56, 72] and tiling database [38] to cite a few. Therefore, most of the existing theoretical approaches cannot be directly applied to large datasets. Indeed, such algorithms have a complexity that is not linear com-

pared to the number of users or permissions to analyze [6, 29, 78]. In this thesis, the main drawbacks of traditional role mining tasks that are based on minimality measures are highlighted. Indeed, a minimal set of roles is generally not useful to the system administrators. We point out that in order to provide a good candidate role-set, role mining algorithms have to take into account business information as well.

We address the problem of reducing the role mining complexity in RBAC systems by making it practical and usable. The first approach that we propose is to elicit stable candidate roles, by contextually simplifying the role selection task. Furthermore, we introduce two methodologies that can be combined together in order to elicit meaningful roles, while reducing the role mining complexity. The first is a divide et impera strategy that is driven by one or more business attributes. The second methodology, overcomes the main limitation of the divide et impera approach by reducing the problem size without sacrificing on utility and accuracy. The original access control dataset is compressed and then analyzed in order to identify interesting portions, which are then reconstructed. Any existing role mining algorithm can be used to analyze the reconstructed portions—that are orders of magnitude smaller than the original dataset.

We point out that to effectively elicit a deployable role-set, role engineers have to handle the noise that is always present within access control datasets. It is important to figure out if there are assignments that have been not granted, but that, if they would be granted, they could help the management of the role set. Also, it is important to figure out if there are permissions that have been accidentally granted, but that could hinder the role management. We introduce two algorithms that are able to find missing and abnormal user-permission assignments. Furthermore, we introduce a fast update operation that quickly re-evaluate the dataset when a modification occurs during the normal life cycle of the roles.

Further, we introduce a new approach to the role mining, referred to as *visual role mining*. It offers a graphical way to effectively *navigate* the result of *any* existing role mining algorithm, showing at glance what it would take a lot of data to expound. Moreover, we allow to *visually identify meaningful roles* within access control data without resorting to traditional role mining tools.

Finally, some final remarks as well as future research directions are highlighted.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

Access control mechanisms are crucial design elements that aim at mediating requests to data and services. Among all models proposed in the literature, *role-based access control* (RBAC) [2] has become the norm for managing permissions within commercial applications [86]. The high-level formalism and the simplicity of its design made it an attractive and pragmatic choice for implementing access control. Under RBAC, a role is a set of permissions, while users acquire the permissions to perform system functions only when they are assigned to specific roles. Because of the intuitiveness of RBAC, security policies can be easily defined by business users that do not usually have all the needed IT knowledge.

The roles definition phase, also known as *role engineering*, has been largely recognized as the most expensive task in deploying RBAC [63]. Interestingly, role design determines RBAC's cost. When there are hundreds or thousands of users within an organization, with individual functions and responsibilities to be accurately reflected in terms of access permissions, only a well-defined role engineering process allows for significant savings of time and money while protecting data and systems [8]. Usually, role engineering approaches can be categorized as: *top-down* and *bottom-up*. The former generally ignores existing permissions, and carefully decomposes business processes into elementary components, identifying which access permissions are necessary to carry out specific tasks in order to formulate roles. The latter aims at extracting roles from the existing access permissions [52]. Top-down analysis has been recognized as the costliest part for deploying RBAC as it requires a significant amount of analysis of the business processes, and moreover it has to be performed mainly manually [21]. For this reason, and also because the complexity of this task grows with the introduction of new and more complicated information systems, bottom-up approaches are largely preferred to top-down approaches. Indeed, they can be easily automated by resorting to data mining

techniques, thus leading to what is usually referred to as *role mining*.

In recent years, various role-definition approaches have been proposed, where each of them bears in mind a different aspect of the problem [59]. Many of these works proved that the role mining problem is reducible to well-known NP-hard problems, and several heuristic algorithms have been proposed. However, when dealing with large organizations, such algorithms are not practically useful. Indeed, they have a complexity that is not linear compared to the number of users or permissions to analyze [6, 29, 78]. Especially when the role engineering task is periodically and frequently performed as part of the life-cycle of roles, the number of elicited candidate roles as well as the running time become an issue, [8]. To overcome these limitations, role mining products such as Oracle Identity Analytics[1] or Bay31 Role Designer[2] use a *Divide et Impera* approach. For example, Oracle Identity Analytics separates large numbers of users into more manageable groups, called "waves", for the purpose of defining roles. This is accomplished by first dividing users into business units based on their managers, departments, divisions, or other attributes. Then, these business units are grouped into waves (usually four to six business units per wave), which are independently analyzed using role mining, clustering and categorization algorithms. The Bay31 Role Designer allows a "role engineering campaign" manager to delegate parts of the overall role analysis to different people. Each of these people contributes their particular knowledge of the business and Role Designer combines it all to form a coherent set of business role. To identify the subsets of data to analyze, the product uses a partitioning approach inspired by one of our work [14]. In particular, using an index referred to as "entrustability", the access data is divided into smaller subsets that are homogeneous from a business perspective. Instead of performing a single bottom-up analysis on the entire organization, each subset is analyzed independently. This eases the attribution of business meaning to roles elicited by any existing role mining algorithm, and reduces the problem complexity.

The main drawback of the afore mentioned *Divide et Impera* approaches is that it is not possible to elicit roles that spread across several partitions. Yet, such roles can be very useful to system administrators. For instance, it is likely that users spreading across different partitions (e.g., different organization units) have common entitlements that simply permit access to top-level resources (i.e., to resources at their entry points). A so called "structural" role might control access to an application's entry point, whereby a user could only

[1] http://www.oracle.com/us/products/middleware/identity-management/oracle-identity-analytics/index.html

[2] http://www.bay31.com/role_designer

open that application if he or she had been assigned to a structural role that grants that access. Conversely, "functional" roles are defined as controlling access to resources within applications. The role engineering process should include the definition of both structural roles and functional roles [22].

In real scenarios, no data is clean, and the access control information is not an exception. Unfortunately, this is an often neglected aspect. Existing access control datasets contain large portions of permissions that are exceptionally or accidentally granted. Furthermore, it is very common to find portions of permissions that are not explicitly granted to some users, but that may be granted without undermining any security policy. Therefore, when taking into account access control information, by noise we refer to a permission being recorded as a denial or a denial being recorded as a permission.

Noise can cause multiple problems during the role mining task. Indeed, this intrinsic factor hinders the elicitation of meaningful roles, badly biasing the role mining analysis. For example, if we are trying to minimize some mathematical function, we may get a suboptimal solution (i.e., larger number of roles). Furthermore, the discovered role set will reconstitute noisy bits. Since the discovered roles are also contaminated with noise, they perpetuate the existing errors. Thus, due to the noise, the error affects future users who may be mistakenly given over and under-permissions right from the start. This can create great problems and reduce the benefits of using RBAC in the first place. However, the difficulties that arise from noisy access control datasets have not been adequately addressed yet. As a matter of fact, most role mining approaches assume the input data is clean, and only attempt to optimize the RBAC state.

1.1 Candidate Contributions

The research activity of the candidate mainly focused on the Role Based Access Control, and in particular on the problems related to the role mining. However, the scientific interest of the candidate towards other security areas lead to some additional publications. This thesis collects and harmonizes the main contributions that handle the role mining, while the remaining ones are briefly described in Section 1.1.2.

1.1.1 Role Base Access Control

Conference and journal papers that have been published on this topic are summarized in Table 1.1. For each paper, it is indicated: the bibliographical reference that gives full details of the conference/journal, the paper title, and an

Table 1.1 Summary of Candidate's Contribution in RBAC area

Ref.	Title	Contribution
[20]†	*Visual Role Mining: A Picture Is Worth a Thousand Roles*	A graphical view of user-permission assignments that allows for quick analysis and role elicitation
[18]†*	*A novel Approach to Impute Missing Values and Detecting Outliers in Binary Matrices*	Simplifying the role mining task by automatically detecting and discarding exceptions from analyzed data
[19]†*	*Privacy Preserving Role-Engineering*	An approach for a privacy-preserving outsourcing of the role engineering task
[80]	*Role Mining: From Theory to Practice*	How to make role mining *practical* and *usable* for actual deployment
[11]	*A Probabilistic Bound on the Basic Role Mining Problem and its Applications*	A sharp estimation of the minimum number of roles that can be elicited by role mining algorithms
[9]	*A Formal Framework to Elicit Roles with Business Meaning in RBAC Systems*	A metric to evaluate the meaning of roles, to use in conjunction with [6]
[14]	*Mining Business-Relevant RBAC States Through Decomposition*	An entropy-based measure of the expected uncertainty in locating homogeneous users and permissions and its applications
[10]	*Mining Stable Roles in RBAC*	Easing the mining task by discarding user-permission assignments that lead to "unstable" roles
[16]†	*Taming Role Mining Complexity in RBAC*	Efficient algorithms to implement the approach proposed in [10]
[17]†*	*A Business-Driven Decomposition Methodology for Role Mining*	An approach to decompose the role mining problem in business drive subtasks
[12]	*ABBA: Adaptive Bicluster-Based Approach to Impute Missing Values in Binary Matrices*	A novel approach to identify missing user-permission assignments that could simplify the role mining task
[15]†	*A New Role Mining Framework to Elicit Business Roles and to Mitigate Enterprise Risk*	Two metrics to estimate the expected complexity of analyzing mining outcome, as well as a divide-and-conquer approach that use them
[13]	*Evaluating the Risk of Adopting RBAC Roles*	A framework to rank users and permissions by the risk related to markedly deviating from "peers"

*Unpublished and under review process
†Journal paper

outline of the contribution. In the bibliography at the end of the document, the authors of papers cited by Table 1.1 are mainly reported in alphabetical order, but the candidate is either the first or the second contributing author for all the papers.

When dealing with large organizations, the number of elicited candidate roles as well as the running time become an issue. This happens especially when the role engineering task is periodically and frequently performed as part of the roles life-cycle. Indeed, it is often impossible to deal with datasets involving thousand of users and permissions without using a partitioning approach. Several commercial products uses this approach, but often the partitioning is manually executed. In [15], the candidate proposed a divide-and-conquer strategy, leveraging on two indices (minability and similarity) that measure the expected complexity to find roles with clear business meaning. The decomposition process is driven by selecting the business attribute that maximize such indices. In [14] the candidate introduced another index, that provides, for a given partition, the expected uncertainty in locating homogeneous set of users and permissions that are manageable with the same role. By choosing the decomposition with the highest values for any of such indices, we most likely identify roles with a tight business meaning. Both [15] and [14] suggest to restrict the role mining analysis to sets of data that are homogeneous from an enterprise perspective. The key observation is that users sharing the same business attributes will essentially perform the same task within the organization. Consequently, it will be easier for an analyst to assign a business meaning to the roles elicited via bottom-up approaches. When several attributes are at role engineers disposal, the problem that arise is to select the attribute that induce the "best" partitioning. This problem has been taken into account in [17], where ENTRUSTABILITY, MINABILITY GAIN, and SIMILARITY GAIN have been introduced and compared.

Partitioning approaches, however, have a main drawback: it is not possible to find roles that spread across several partitions. Yet, such roles can be very useful to system administrators. For instance, it is likely that users spreading across different partitions (e.g., different organization units) have common entitlements that simply permit access to top-level resources (i.e., to resources at their entry points). A so called "structural" role might control access to an application's entry point, whereby a user could only open that application if he or she had been assigned to a structural role that grants that access. Conversely, "functional" roles are defined as controlling access to resources within applications. The role engineering process should include the definition of both structural roles and functional roles. Therefore, in [80] the candidate introduced an approach that essentially provides practical and usable solutions to the role mining problem, improving the scalability of the role mining

solutions while eliciting structural roles. Hence, overcoming the main limitation of existing *Divide et Impera* approaches. In [80], the role mining problem is made tractable by reducing the problem size without sacrificing on utility and accuracy. The proposed approach does this by effectively compressing the original access control dataset, analyzing the compressed dataset to identify interesting portions, and reconstructing the portions of the original dataset that are worth investigating. Scalability is assured by targeted partitioning that ensures that the created sub-problems focus only on the interesting portions of the original dataset—and are therefore orders of magnitude smaller than the original one. At this point, *any* existing role mining algorithm can be used to analyze user-permission assignments within these subsets. Thus, this approach is agnostic to the specific role mining methodology used, namely it can be used in conjunction with them to enhance their scalability.

When performing the role mining task, it is of utmost importance to handle "clean" datasets. Unfortunately, existing datasets contain large portions of *exceptionally/accidentally granted/denied permissions* that can hinder the elicitation of meaningful roles. In [10], the candidate introduced the concept of "stable" candidate roles: roles that likely remains unchanged during their lifecycle. A theoretical framework that allows to identify and then discard assignments that can only be managed via "unstable" roles is also introduced. Then, in [16] the candidate introduces a fast algorithm that implements such a strategy, avoiding the generation of unstable roles during the application of any role mining algorithm. Denied permissions are represented by *missing* user-permission assignments. Unfortunately, some of the *missing* user-permission assignments, if would be granted, might largely simplify the mining task. Referring to the more general case of missing values in binary matrices, in [12] the candidate introduced a viable and effective approach to identify such missing values.

In [20], the candidate proposed a visual approach to the role mining task. In practice, abstract user-permission patterns (i.e., RBAC roles) are managed as *visual patterns*. The rationale behind this approach is that visual representations of roles can actually amplify cognition, leading to optimal analysis results [31,48]. A graphical way to effectively *navigate* the result of *any* existing role mining algorithm is provided, showing at glance what it would take a lot of data to expound. Visualization of the user-permission assignments is performed in such a way to isolate the *data noise*, allowing role engineers to focus on relevant patterns, leveraging their cognition capabilities. Further, *correlations* among roles are shown as overlapping patterns, hence providing an intuitive way to discover and utilize these relations.

1.1.2 Other Contributions

Additional publications that are not strictly related to RBAC but that are bound to the general field of Information Security (Access Control is included within) are summarized in Table 1.2. In particular, the candidate introduced the use of epidemiological inspired models to assure information survivability in Unattended Wireless Sensor Networks. In the following we will briefly introduce the main features of these networks, highlighting the contributions of the candidate.

In Wireless Sensor Networks (WSNs), data sensed by the nodes are sent in real time (or quasi real time) to the sink. Nodes may rely on a multi-hop protocol to reach the sink, that it is however continuously available. On the contrary, Unattended Wireless Sensor Networks (UWSNs) are characterized by the intermittent presence of the sink. In such a scenario, nodes have to accumulate information sensed on the field, and try to offload it to the sink as soon as it is available. The reasons that have led to the introduction of this type of Wireless Sensor Networks are tightly related to the inaccessibility of the environment where the nodes may be deployed. As an example, consider a monitor system to detect poaching in a national park. The difficulty to hide the sink, and the size of the monitored area are the main reasons to adopt an unattended WSN. This is also the case of an underground, or submarine sensor network: the inaccessibility of the monitored area, and the technical problems that arise to connect the sink with the sensors do not allow the use of a traditional sink. In all these cases, an intermittent sink is the only alternative.

Since sensors are usually deployed in hostile environments, and since the sink cannot continuously check that they are correctly working, UWSNs represent an easy and attractive target for an adversary. Mainly due to cost reasons, a typical sensor is a mass-produced commodity device with no specialized secure hardware. Therefore, while the sink is away, the adversary can compromise the sensors, read, delete or alter an information, and disappear without leaving any evidence of its illegal behavior. In [26], the candidate introduced two epidemic models that are able to assure data survivability in Unattended Wireless. These models do not rely on any cryptographic ability of the sensors. In [27], one of these models has been deeply analyzed. Here, a pure controlled epidemic technique has been used to provide a trade-off between data survivability, optimal usage of sensor resources, and a fast and predictable collecting time. The candidate proved that by estimating the maximal power of an attacker it is possible to set up a probabilistic bound on the survivability of the data. This is the first work in the area that considers the collecting time as an issue; consequently, it might open up a new line of research. In [28], the work

Table 1.2 Summary of Candidate's Contribution in other areas

Ref.	Title	Contribution
[26]	*Introducing Epidemic Models for Data Survivability in Unattended Wireless Sensor Networks*	Preliminary assessment of epidemic-domain inspired approaches to model the information survivability in UWSN
[27]	*Epidemic data survivability in unattended wireless sensor networks*	A theoretically sound result that assures at the same time: Data survivability, an optimal usage of sensors resources, and a fast and predictable collecting time
[28][34]	*Epidemic Theory and Data Survivability in Unattended Wireless Sensor Networks: Models and Gaps*	Highlighting gaps of epidemic models for information survivability

published in [26] has been extended in several directions for an invited journal version. The candidate relaxed the hypothesis regarding the availability of point to point connectivity used in the previous paper, and introduced several geometrical constraints that are typical of a standard deployment setting. The candidate took into account the problems that arise in the geometrical scenario, and proposed a simple but effective solution.

1.2 Outline of the Thesis

The remainder of this thesis is organized as follows.

Chapter 2 (Background and Related Work) provides an overview of the basic access control concepts. Subsequently, the way an organization can migrate from other access control models to RBAC through *role engineering* methodologies is explained, and the typical classification of the various role engineering approaches is presented. Further, a set of concepts that is extensively used in the following chapters is introduced, that is *biclusters*, *maximal biclusters*, *pseudo-biclusters* and *maximal pseudo-biclusters*. Also, a few concepts related to the graph theory and its connection with the Role Based Access Control are described.

Chapter 3 (The Role Mining Problem) introduces the general role mining problem and the required formalism. Since this problem has many interpretations, we report the most significant definitions that have been proposed up to this time. That is: the basic Role Mining Problem, the δ-approximated Role

Mining Problem and the Edge Role Mining Problem. Furthermore, we highlight the drawback of these approaches that generally seek for the minimality.

In Chapter 4 (Practical and Usable Approaches to the Role Mining), we look at how to make role mining *practical* and *usable* for actual deployment. We first introduce a strategy for the reduction of the role mining complexity by pruning unstable assignments. In particular, we introduce a graph describing the existing user-permission assignments, and a pruning operation that corresponds to the identification of unstable roles. Further, we introduce a strategy to analyze an existing dataset, and to decompose the role mining task in several sub-tasks. Each of these sub-tasks are executed on a partition of the dataset that is homogeneous from an enterprise perspective. Finally, we introduce a six step methodology that overcomes the limitations of the Divide et Impera approaches, while still improving the scalability of the role mining solutions. We focus on making the role mining problem tractable by reducing the problem size without sacrificing on utility and accuracy. Results of the application of this methodology on real data support our findings.

Chapter 5 (Data Cleansing in Access Control Datasets) deals with pre-mining activities, and in particular "data cleansing". In access control datasets, it is important to figure out if there are assignments that have been not granted, but that, if granted, could help the management of the role set. Also, it is important to figure out if there are permissions that have been accidentally granted, and that could hinder the role management. In this Chapter, we introduce two algorithms that are able to find missing and abnormal user-permission assignments. Furthermore, we introduce a fast update operation that quickly re-evaluate the dataset when a modification occurs during the normal life cycle of the roles.

In Chapter 6 (Visual Role Mining), we introduce a new approach to the role mining, referred to as *visual role mining*. We offer a graphical way to effectively *navigate* the result of *any* existing role mining algorithm, showing at glance what it would take a lot of data to expound. Moreover, we allow to *visually identify meaningful roles* within access control data without resorting to traditional role mining tools. Visualization of the user-permission assignments is performed in such a way to isolate the *noise*, allowing role engineers to focus on relevant patterns, leveraging their cognition capabilities. Further, *correlations* among roles are shown as overlapping patterns, hence providing an intuitive way to discover and utilize these relations. In fact, this chapter shows that a proper representation of user-permission assignments allows role designers to gain insight, draw conclusions, and design meaningful roles from both IT and business perspectives.

To conclude, Chapter 7 (Conclusion) offers some final remarks.

2

Background and Related Work

This chapter offers a brief overview of the basic access control concepts. Subsequently, we explain the way an organization can migrate from other access control models to RBAC through *role engineering* methodologies. The typical classification of the various role engineering approaches is presented. Furthermore, we introduce a set of tools that is extensively used in the following chapters, that is *biclusters*, *maximal biclusters*, *pseudo-biclusters* and *maximal pseudo-biclusters*. Finally, we introduce a few concepts related to graph theory and Role Based Access Control.

2.1 Role-Based Access Control

In the current competitive environment, data must be secured, and access to that data must be limited to the "minimum necessary". Security models such as Mandatory Access Control and Discretionary Access Control have been the means by which to secure information and regulate access. But due to the inflexibility of these models, the rather new security concept of Role Based Access Control (RBAC), as proposed by the National Institute of Standards and Technology (NIST) [2], became the more prominent security model [86]. The high-level formalism and the simplicity of its design made RBAC an attractive and pragmatic choice for implementing access control. Under RBAC, a role is a set of permissions, while users acquire the permissions to perform system functions only when they are assigned to specific roles. Because of the intuitiveness of RBAC, security policies can be easily defined by business users that do not usually have all the needed IT knowledge.

The need for RBAC

Lower technology costs and a great need for data access and sharing in a competitive market has driven the development of new technologies and standards. Both system vendors and implementers have been looking for the means to properly administer rapidly expanding and costly infrastructures. Indeed, the downtime of users and the delay in account creation can provide large economic losses. Account security in many organizations is loose at best, perpetuated by a high volume of requests and a security administration model grossly overpowered by the infrastructure it is forced to support. Fortunately, Role Based Access Control (RBAC) allows for easiest administration of large and complex corporate environments without sacrificing the need for securing data and access to it.

The earliest forms of access control systems assigned privileges to users. Conventionally, managing entitlements has been considered technical, as they are related to vertically-managed applications without much business involvement. However, with the publication of regulatory requirements such as the US Sarbanes-Oxley Act [68], US Health Insurance Portability and Accountability Act (HIPAA) [49], Gramm-Leach-Biley Act (GLBA) [41], and EU Privacy Protection Directive 2002/58/EC [40], it is increasingly important to revise the entitlement management process from a business perspective, as it becomes a security governance and compliance issue. The addition of *user groups* improved that situation. Many legacy systems and applications managed user permissions by means of groups. Under this model, permissions are assigned to groups, while users get permissions granted by being a member of a group. The ability to assign permissions to a group and determine who can inherit the permission is considered *discretionary* since it is typically decided by system owners. However, authority to assign members to a group is deemed non-discretionary and usually is performed by the security staff.

In order to better understand the benefits of a Role-Based Access Control (RBAC) security administrative model, we must understand some of the current concepts being utilized. While there are many variations and ideas behind security administration, we are going to focus on three basic concepts: Mandatory Access Control, Discretionary Access Control and of course Role Based Access Control.

Mandatory Access Control (MAC) Mandatory Access Control or MAC utilizes security provisions that are typically hard coded into an application or operating system. These provisions or rules apply to all objects, applications and various resources including the end-user that tries to access the data it is designed to protect. More so than operating systems, applications, especially

military, governmental or occasionally specialized in-house developed applications, incorporate the MAC concept. This typically begins by classification of data, for example sensitive, secret and confidential, and next the classification of resources that will be making requests for data. This type of access control is very secure in that it can be granular in design. Some implementations of MAC include a hierarchal structure, meaning that a user assigned secret has access to secret and sensitive data. A user assigned confidential could also have secret and sensitive data access in a hierarchal implementation. Another security benefit of this model is that the security rules are hard coded into the software so the chance for administrative error or social engineering is greatly reduced. Where MAC can fall short is in the development or modification of the rules within the application. Because it is hard coded programmers will need to review the coding of the application and make changes. This could be especially frustrating if the application is a turnkey solution from a vendor requiring vendor assistance for modification. MAC is best suited for specialty applications for a group of users with rather similar needs. As a rule it typically does not function well as a corporate wide authentication and accessibility security model. The inability for MAC to change in the age of consolidation and constant corporate mergers makes it an administrative nightmare for general account administration in a dynamic and evolving environment.

Discretionary Access Control (DAC) Discretionary Access Control (DAC) works both as a centralized security model and a distributed model. A centralized security model is when an administrator or team of administrators distributes access to data, applications and network devices. All requests for access changes need to be completed by this single department. In a large organization this can be very time consuming, especially if the administrators are off site or outsourced. A distributed model allows responsible and knowledgeable personnel to distribute access to data and applications. In large companies this may be a manager, supervisor, or team leader. In small organizations it may simply be a team member that is particularly skilled with computers and security. The benefit of a distributed model is that delays can be avoided since the administration of accounts is dispersed. Because DAC can be implemented in a distributed security model, it can greatly reduce account access change turnaround times by removing the middle man. While DAC would appear a reasonable solution for both large and small network environments, there are also some drawbacks to consider. Since access is distributed at the discretion of the data owner, there is the potential that uniformity of access for end-users with like job functions could be diminished. The lack of understanding by the data owner could allow access greater than the minimum necessary. If explicit

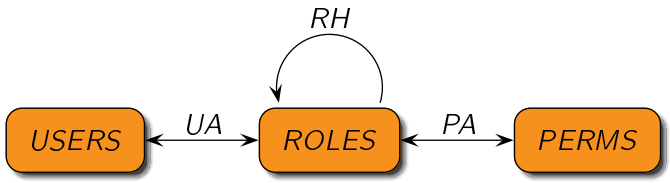


Figure 2.1 RBAC entities

rights to data is not known by the data owners or administrators, then who can be sure that the access is not carried with the user as they move from job function to job function within the company. These issues could open the doors to costly and embarrassing repercussions.

RBAC: Fundamental Concepts

Role-based access control (RBAC) [2] is the next evolutionary step in access control. The fundamental concept of RBAC is that roles aggregate privileges. Users inherit permissions by being members of roles. Based on the least privilege access principle, they are given no more than what is required to perform their job function. In this case, assignment of permissions to a role and determining membership of roles is supposed to be non-discretionary. The *National Institute of Standards and Technology* (NIST) delivered in 2004 a standard for RBAC [2] via the INCITS fast track process. The standard provides users and vendors of information technology products with a coherent and uniform definition of RBAC features. It also offers a formal description of all the entities provided by the RBAC model. In the following, we only summarize the entities of interest for the present thesis:

- *PERMS*, the set of all possible access permissions;
- *USERS*, the set of all system users;
- *ROLES*, the set of all roles;
- $UA \subseteq USERS \times ROLES$, the set of user-role assignments;
- $PA \subseteq PERMS \times ROLES$, the set of permission-role assignments;
- $RH \subseteq ROLES \times ROLES$, the set of hierarchical relationships between pairs of roles.

For the sake of simplicity, we do not consider other entities provided by the standard such as sessions or separation of duty constraints. Indeed, such entities are not relevant for the purpose of this thesis.

The previous entities are also depicted in Figure 2.1. As for role hierarchy, *RH* derives from the partial order [24] based on permission-set inclusion.[1] Hence, $\langle r_1, r_2 \rangle \in RH$ indicates that all the permissions assigned to r_1 are also assigned to r_2, and some more permissions are assigned to r_2. The symbol '$\succeq$' indicates the ordering operator. If $r_1 \succeq r_2$, then r_1 is referred to as the *senior* of r_2, namely r_1 adds certain permissions to those of r_2. Conversely, r_2 is the *junior* of r_1. Additionally, the symbol '$\succcurlyeq$' also indicates the ordering operator, but there is no intermediate elements between operands. In other words,

$$\forall r_1, r_2 \in ROLES \; : \; r_1 \succcurlyeq r_2 \implies \nexists r' \in ROLES \; : \; r' \neq r_1 \; \wedge \; r' \neq r_2 \; \wedge \; r_1 \succeq r' \; \wedge \; r' \succeq r_2.$$

If $r_1 \succcurlyeq r_2$ then r_1 is referred to as an *immediate senior* of r_2, while r_2 is referred

In addition to the functions that are provided by the ANSI standard, we use the following other entities:

- $UP \subseteq USERS \times PERMS$, the existing user-permission assignments to analyze;
- $perms$: $USERS \rightarrow 2^{PERMS}$, the function that identifies permissions assigned to a user. Given $u \in USERS$, it is defined as $perms(u) = \{p \in PERMS \mid \langle u, p \rangle \in UP\}$.
- $users$: $PERMS \rightarrow 2^{USERS}$, the function that identifies users that have been granted a given permission. Given $p \in PERMS$, it is defined as $users(p) = \{u \in USERS \mid \langle u, p \rangle \in UP\}$.

2.2 Role Engineering

Many organizations are in the process of moving to role based access control. The process of developing an RBAC structure for an organization has become known as "role engineering". The roles definition phase has been largely recognized as the most expensive task in deploying RBAC [63]. Interestingly, role design determines RBAC's cost. When there are hundreds or thousands of users within an organization, with individual functions and responsibilities to be accurately reflected in terms of access permissions, only a well-defined role engineering process allows for significant savings of time and money while protecting data and systems [8]. Usually, role engineering approaches can be categorized as: *top-down* and *bottom-up*. The former generally ignores existing permissions, and carefully decomposes business processes into elementary components, identifying which access permissions are necessary to carry out specific tasks in order to formulate roles. The latter aims at extracting roles from the existing access permissions [52]. Top-down analysis has been recognized as the costliest part for deploying RBAC as it requires a significant amount of analysis of the business processes, and moreover it has to be performed mainly manually [21]. For this reason, and also because the complexity of this task grows with the introduction of new and more complicated information systems, bottom-up approaches are largely preferred to top-down approaches. Indeed, they can be easily automated by resorting to data mining techniques, thus leading to what is usually referred to as *role mining*. In the following we will analyze in more details these two approaches.

Top-Down approach is primarily business-driven, and roles are defined based on the responsibilities of a given job function. For roles to be effective, a strong alignment between business and IT objectives is of utmost importance. Roles are defined by reviewing organizational business and job functions and map-

ping the permissions for each job function. This approach provides business oversight and alignment of roles with business functions and re-usability. Top-down role engineering was first illustrated by Coyne [21]. He places system users' activities as high-level information for role identification; this approach is only conceptual, thus it lacks technical details. Fernandez and Hawkins [32] propose a similar approach where use-cases are used to determine the needed permissions. Röckle et al. [65] propose a process-oriented approach that analyzes business processes to deduce roles. The *role-finding* concept is introduced to deduce roles from business needs or functions. Information is organized in three different layers: process layer, role layer, and access rights layer. Crook et al. [23] leverage organizational theory to elicit role-based security policies. Neumann and Strembeck [61] present a more concrete approach to derive roles from business processes. They offer a scenario-based approach where a usage scenario is the basic semantic unit to analyze. Work-patterns involving roles are analyzed and decomposed into smaller units. Such smaller units are consequently mapped with system permissions. Shin et al. [71] use a system-centric approach supported by the UML language to conduct top-down role engineering. Role engineering is discussed from the perspective of systems to be protected, assisting with the general understanding of RBAC roles and permissions in conjunction with business processes. Epstein and Sandhu [30] also use UML to address role engineering. Kern et al. [50] propose an iterative and incremental approach based on the role life-cycle, pertaining to analysis, design, management, and maintenance. The book of Coyne and Davis [22] is a practical reference that helps to assess some of the previously cited role engineering approaches.

Bottom-Up approach is based on performing role-mining/discovery by exploring existing user permissions in current applications and systems. Once roles has been elicited, the next step is to perform role normalization and rationalization. In this approach, roles are defined to meet specific application or system access requirements. One of the challenges of this sampling is that it requires viable tools to perform role mining. An alternate approach is to select a set of representative users and extract the entitlements that best describe the job function. If the user population is significant, it would be ideal to sample a certain percentage of the population to validate the accuracy of the results. One of the outcomes of this approach is that users often accumulate entitlements based on their previous job functions performed over a period of time; it can become too daunting to validate roles without the business involvement. This is a key aspect of role rationalization to be considered as part of a bottom-up approach. Kuhlmann et al. [52] first introduced the term "role mining", trying to apply existing data mining techniques to elicit roles from existing access data. Indeed, role mining can be seen as a particu-

lar application of *Market Basket Analysis* (MBA, also known as *association-rule mining*), a method of discovering customer purchasing patterns by extracting associations or co-occurrences from transactional store databases. This translation can be done by simply considering permissions, roles and users instead of products, transactions and customers, respectively. Among all possible algorithms used in this area, Apriori [1] is the most common. After the first proposal, the community started to identify specific algorithms to solve this particular problem instead of using existing approaches. The first algorithm explicitly designed for role engineering was ORCA [69] which applies hierarchical clustering techniques on permissions. In practice, ORCA discovers roles by merging permissions appropriately. However, the order in which permissions are merged determines the outcome of roles. Moreover, it does not allow overlapping roles, which is a significant drawback. Vaidya at al. [78] applied subset enumeration techniques to generate a set of candidate roles, computing all possible intersections among permissions possessed by users. Subset enumeration techniques had been advocated earlier by Rymon [67]. More recently, the same authors of [78] also studied the problem of finding the minimum number of roles that cover all permissions possessed by users [75, 76]. By leveraging binary integer programming, Lu et al. [56] presented a unified framework for modeling the role number minimization problem. Ene et al. [29] offered yet another alternative model to minimize the number of roles, reducing it to the well-known problem of the minimum biclique covering. Zhang et al. [85] provide an attempt to contextually minimize the number of user-role, permission-role, and role-role relationships. Frank et al. [34] model the probability of user-permission relationships, thus allowing to infer the role-user and role-permission assignments so that the direct assignments become more likely. The authors offer a sampling algorithm that can be used to infer their model parameters. Several works prove that the role mining problem is reducible to many other well-known $\mathcal{NP}$-hard problems, such as clique partition, binary matrix factorization, bi-clustering, graph vertex coloring (see Chapter 4) to cite a few. Recently, Frank et al. [35] provided a detailed analysis of the requirements for role mining as well as the methods used to assess results. They also proposed a novel definition of the role mining problem that fulfills the requirements that real-world enterprises typically have.

Limitations of Standard Role Engineering Approaches The standard top-down and bottom-up approaches do not always lead to the optimal set of roles from a business perspective. In [6], we firstly described an approach that allows for the discovery of roles with business meanings through a role mining

algorithm. The most similar approach to ours has been provided by Molloy at al. [58]. It tackles the problem in two settings. When only user-permission relations are available, the authors propose to discover roles by resorting to *formal concept analysis* (FCA). FCA is a theory of data analysis which identifies conceptual structures among data sets. The mathematical lattices that are used in FCA can be interpreted as classification systems. If user attributes are additionally available, they utilize user attributes to provide a measurement of the RBAC state complexity, called "weighted structural complexity". The authors observe that adopting the RBAC model reduces the number of relationships to be managed and give a cost (weight) for each parameter of the global structural complexity.

Above mentioned role mining approaches assume clean input data which is not what happens in a real scenario. Indeed, the noisy input data is pervasive. In [60], Molloy et al. suggested a cleaning approach by dividing the role mining problem into two steps: noise removal and candidate role generation. They used non-binary rank reduced matrix factorization to identify noise. This approach is also applicable outside role engineering and may be used to identify errors or predict missing values in any access control matrix. In [77], Vaidya et al. proposed a formal model of noise and experimentally evaluated the δ-approximated algorithm previously proposed against its robustness to noise. Unfortunately, their algorithm is only able to handle additive noise (that is when a permission can only be incorrectly given, not incorrectly revoked).

2.3 Biclusters

In this section, we will introduce a set of concepts that are extensively used in the following chapters: biclusters, maximal biclusters, pseudo-biclusters and maximal pseudo-biclusters.

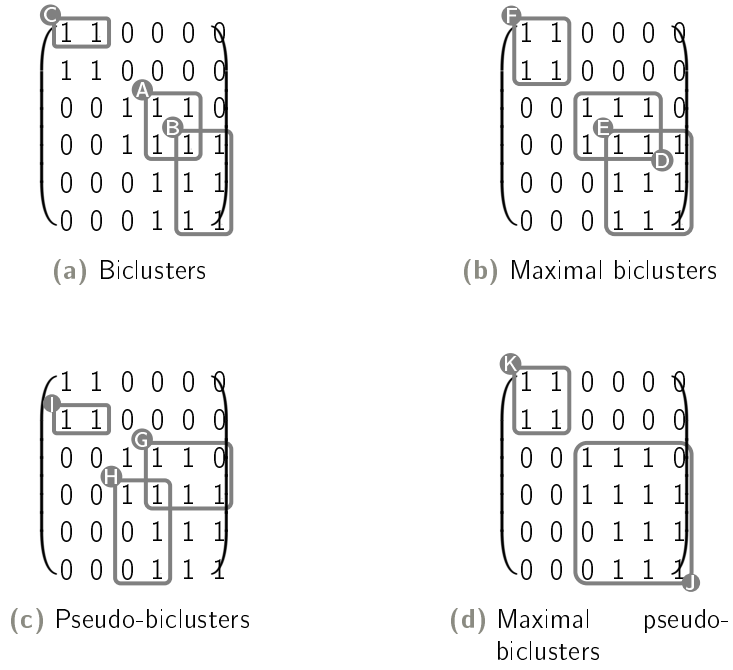
Definition 2.1 (Bicluster) Given a binary matrix M, a *bicluster* B is a pair $\langle R, C\rangle : R \subseteq [n],\ C \subseteq [m]$ such that the submatrix of M identified by selecting only the rows R and the columns C is completely filled by 1's, namely:

$$\forall i \in R,\ \forall j \in C\ :\ m_{ij} = 1.$$

Definition 2.2 (Maximal Bicluster) Let $B = \langle R, C\rangle$ be a bicluster in the binary matrix M. It is also a *maximal bicluster* if:

$$\nexists \text{ a bicluster } B' = \langle R', C'\rangle\ :\ R \times C\ \subset R' \times C'.$$

Informally, a maximal bicluster is "representative" for all its possible subset of rows (or columns) that have 1's for a given subset of columns (or rows).



(a) Biclusters (b) Maximal biclusters

(c) Pseudo-biclusters (d) Maximal pseudo-biclusters

Figure 2.2 Some examples of data patterns

The key observation behind a maximal bicluster is that two columns (or rows) which always contain 1's in a certain set of rows (columns) should simultaneously belong to the same bicluster. Many real-world examples justify the interest in maximal biclusters. In data mining applications, what we refer to as maximal bicluster is also known as *closed itemset* [82]. Applications include the discovery of association rules, strong rules, correlations, sequential rules, episodes, multidimensional patterns, and many other important discovery tasks [43]. Thus, a maximal bicluster definitely represents an interesting pattern to identify among the available data.

Figure 2.2(a) shows an example of binary matrix and also highlights some of the biclusters that can be identified within it. For example, the last three rows and last two columns—the cell grouping denoted by 'B' in the figure—contains cells filled by 1's. Hence, they represent a bicluster. However, they do not represent a maximal cluster, because the third from last column also contains 1's in the last three rows. Indeed, the cells denoted by 'E' in Figure 2.2(b) represent a maximal bicluster: 'E' cannot be "expanded" by adding other rows and/or column. Further, the maximal bicluster 'E' contains the bicluster 'B'.

In the following we will extend the bicluster concept given in definitions 2.1 and 2.2:

Definition 2.3 (Pseudo-Bicluster) Given a binary matrix M, a *pseudo-bicluster* B is a pair $\langle R, C\rangle : R \subseteq [n],\ C \subseteq [m]$ that has at least one row and one column filled by 1's, formally:

$$\exists i \in R,\ \exists j \in C,\ \forall \ell \in R,\ \forall k \in C\ :\ m_{ik} = 1,\ m_{\ell j} = 1.$$

For ease of exposition, for a given pseudo-bicluster $B = \langle R, C\rangle$ we also denote with $\hat{R} \subseteq R$ the set of rows filled by 1's, and with $\hat{C} \subseteq C$ the set of columns filled by 1's, that is:

$$\forall i \in \hat{R}, \forall j \in C : m_{ij} = 1,$$
$$\forall j \in \hat{C}, \forall i \in R : m_{ij} = 1.$$

Definition 2.4 (Maximal Pseudo-Bicluster) Let $B = \langle R, C\rangle$ be a pseudo-bicluster in the binary matrix M. It is also a *maximal pseudo-bicluster* if:

$$\nexists \text{ a pseudo-bicluster } B' = \langle R', C'\rangle\ :\ \hat{R} \times \hat{C}\ \subset\ \hat{R}' \times \hat{C}'.$$

A maximal pseudo-bicluster is a pseudo-bicluster such that its rows and columns filled by 1's cannot be "expanded" by adding columns and rows. Figure 2.2(c) shows some examples of pseudo-biclusters. In particular, the matrix portion denoted by 'H' has all cells equal to 1 for the fourth row and the fourth column of the matrix. However, it is not a maximal pseudo-bicluster, since the fourth row and the fourth column contain other cells equal to 1 that are not contained within 'H'. Hence, the pseudo-bicluster 'H' can be "expanded" to the area denoted by 'J' in Figure 2.2(d), which represents a maximal pseudo-bicluster. Note that, in this case, there is more than one column filled by 1's.

The following lemma relates biclusters to pseudo-biclusters:

Lemma 2.1 *If $B = \langle R, C\rangle$ is a bicluster, it is also a pseudo-bicluster.*

PROOF Since $B = \langle R, C\rangle$ is a bicluster, then $\forall i \in R,\ \forall j \in C\ :\ m_{ij} = 1$. This also means that $\forall \ell \in R : m_{\ell j} = 1$ and $\forall k \in C, m_{ik} = 1$, namely $C = \hat{C}$ and $R = \hat{R}$. Thus, B is a pseudo-bicluster according to Definition 2.3. ■

Biclusters in the Role Based Access Control Biclusters, Pseudo-Bicluster, maximal biclusters and maximal pseudo-biclusters have been previously introduced considering the general case of a binary matrix M. In the Role Based Access Control we use these concepts related to the set *UP*, that defines a binary relation between the sets *USERS* and *PERMS*.

For the sake of clarity, below we contextualize the definitions of pseudo-bicluster, and maximal pseudo-bicluster within the role based access control domain.

Definition 2.5 (Pseudo-Bicluster in RBAC) Given the user-permission assignments *UP* to analyze, a *pseudo-bicluster* B is a pair $\langle U, P\rangle : U \subseteq USERS,\ P \subseteq PERMS$ such that at least one user has all the permissions P granted, and at least one permission is granted to all the users U, formally:

$$\exists u \in U,\ \exists p \in P,\ P \subseteq perms(u), U \subseteq users(p).$$

For a given pseudo-bicluster $B = \langle U, P\rangle$ we also denote with $\hat{U} \subseteq U$ the set of users that have all of and only those permissions $p \in P$ granted, and with $\hat{P} \subseteq P$ the set of permissions that are granted to all of and only the users $u \in U$, formally:

$$\hat{U} := \{u \in USERS, perms(u) = P\}$$
$$\hat{P} := \{p \in PERMS, users(p) = U\}$$

Definition 2.6 (Generator of a Pseudo-Bicluster) An assignment $\langle u, p\rangle$ is a generator of the Pseudo-Bicluster $B = \langle U, P\rangle$ if and only if $u \in \hat{U}$ and $p \in \hat{P}$.

Definition 2.7 (Maximal Pseudo-Bicluster in RBAC) Let $B = \langle U, P\rangle$ be a pseudo-bicluster of the user-permission assignments set *UP*. It is also a *maximal pseudo-bicluster* if:

$$\nexists \text{ a pseudo-bicluster } B' = \langle U', P'\rangle \ :\ \hat{U} \times \hat{P} \subset \hat{U}' \times \hat{P}'.$$

In other words, a maximal pseudo-bicluster $B = \langle U, P\rangle$ is a pseudo-bicluster to which we cannot add any other user such that it has all of and only the permissions P granted, nor can we add any other permission that is granted to all of and only the users U. It is important to notice that the more a maximal pseudo-bicluster $B = \langle U, P\rangle$ is *dense* – i.e., almost all the users $u \in U$ have almost all the permissions $p \in P$ granted – the more likely the pattern represented by B is interesting from the role mining perspective. Thus, dense Maximal Pseudo-Biclusters are preferable because the users and the permissions involved are likely to be more similar. Also, Maximal Pseudo-Biclusters that involve many users and many permissions are preferable because they can be potentially managed with "large" roles.

2.4 Graph Theory

We now summarize the relevant graph-related concepts that are required in the rest of this thesis. A *graph* G is an ordered pair $G = \langle V, E \rangle$, where V is the set of *vertices*, and E is a set of unordered *pairs of vertices* (or *edges*). The *endpoints* of an edge $\langle v, w \rangle \in E$ are the two vertices $v, w \in V$. Two vertices in V are *neighbors* if they are endpoints of an edge in E. We refer to the set of all neighbors of a given vertex $v \in V$ as $N(v)$, namely $N(v) = \{v' \in V \mid \langle v, v' \rangle \in E\}$. The *degree* of a vertex $v \in V$ is indicated with $d(v)$ and represents the number of neighbors of v, that is $d(v) = |N(v)|$. The degree of a graph $G = \langle V, E \rangle$ is the maximum degree of its vertices, namely $\Delta(G) = max_{v \in V}\{d(v)\}$.

Given a set $S \subseteq V$, the subgraph *induced* by S is the graph whose vertex set is S, and whose edges are the members of E such that the corresponding endpoints are both in S. We denote with $G[S]$ the subgraph induced by S. A *bipartite graph* $G = \langle V_1 \cup V_2, E \rangle$ is a graph where the vertex set can be partitioned into two subsets V_1 and V_2, such that for every edge $\langle v_1, v_2 \rangle \in E$, $v_1 \in V_1$ and $v_2 \in V_2$. A *clique* is a subset S of V such that the graph $G[S]$ is a complete graph, namely for every two vertices in S an edge connecting the two exists. A *biclique* in a bipartite graph, also called *bipartite clique*, is a pair of vertex sets $B_1 \subseteq V_1$ and $B_2 \subseteq V_2$ such that $\langle b_1, b_2 \rangle \in E$ for all $b_1 \in B_1$ and $b_2 \in B_2$. In the rest of the thesis we will say that a set of vertices S induces a biclique in a graph G if $G[S]$ is a complete bipartite graph. In the same way, we will say that a set of edges induces a biclique if their endpoints induce a biclique. A *maximal* (bi)clique is a set of vertices that induces a complete (bipartite) subgraph and is not a subset of the vertices of any larger complete (bipartite) subgraph. Among all maximal (bi)cliques, the largest one is the *maximum* (bi)clique. The problem of enumerating all maximal cliques in a graph is usually referred to as the *(maximal) clique enumeration problem*. As for maximal biclique, Zaki and Ogihara [84] showed that there exists a one-to-one correspondence among maximal bicliques and several other well-known concepts in computer science, such as *closed item sets* (maximal sets of items shared by a given set of transactions) and *formal concepts* (maximal sets of attributes shared by a given set of objects). Indeed, many existing approaches to role mining have reference to these concepts [7, 29, 56, 58, 78].

A *clique partition* of $G = (V, E)$ is a collection of cliques $C_1, \ldots, C_k$ such that each vertex $v \in C$ is a member of exactly one clique. It is a partition of the vertices into cliques. A *minimum clique partition* (MCP) of a graph is the smallest collection of cliques such that each vertex is a member of exactly one clique. A *biclique cover* of G is a collection of biclique $B_1, \ldots, B_k$ such that for each edge $\langle u, v \rangle \in E$ there is some B_i that contains both u and v. We say that

B_i covers $\langle u, v\rangle \in E$ if B_i contains both u and v. Thus, in a biclique cover, each edge of G is covered at least by one biclique. A *minimum biclique cover* (MBC) is the smallest collection of bicliques that covers the edges of a given bipartite graph. The minimum biclique cover problem can be reduced to many other $\mathcal{NP}$-complete problems, like binary matrices factorization [56, 72] and tiling database [38] to cite a few. Several role mining approaches leverage these concepts [10, 11, 29, 56, 75].

A mathematical tool related to graph theory used is this thesis is the *clustering coefficient*. It was first introduced by Watts and Strogatz [81] in the social network field, to measure the cliquishness of a typical neighborhood. Given $G = \langle V, E\rangle$, we indicate with $\delta(v)$ the number of *triangles* of v, formally:

$$\delta(v) = \left|\left\{\langle u, w\rangle \in E \mid \langle v, u\rangle \in E \ \wedge \ \langle v, w\rangle \in E\right\}\right|. \tag{2.2}$$

A path of length two for which v is the center node is called a *triple* of the vertex v. We indicate with $\tau(v)$ the number of triples of v, namely:

$$\tau(v) = \left|\left\{\langle u, w\rangle \in V \times V \mid \langle v, u\rangle \in E \ \wedge \ \langle v, w\rangle \in E\right\}\right|. \tag{2.3}$$

Definition 2.8 (Clustering Coefficient) The *clustering coefficient* of a graph G is defined as:

$$C(G) = \frac{1}{|V|}\sum_{v \in V} c(v)$$

where

$$c(v) = \begin{cases} \dfrac{\delta(v)}{\tau(v)}, & \tau(v) \neq 0; \\ 1, & \text{otherwise} \end{cases} \tag{2.4}$$

quantifies how close the vertex v and its neighbors are to being a clique. The quantity $c(v)$ is also referred to as the *local* clustering coefficient of v, while $C(G)$ is average of all local clustering coefficients, and it is also referred to as the *global* clustering coefficient of G. Thus, $C(G)$ can be used to quantify "how well" a whole graph G is partitionable in cliques. Another possible definition for the clustering coefficient is to set to 0 when there are no triples. Anyway, our definition is more suitable for our purposes.

3

The Role Mining Problem

Several approaches exist for role mining, and majority of them employ existing data mining techniques or their variants to discover roles [9, 29, 52, 59, 70]. In this section, we will introduce the needed formalism to describe in a general framework the role mining problem, then we will report on the main variants that have been introduced in the literature: the basic Role Mining Problem (RMP), the δ-approximated RMP and the edge RMP. An inherent problem with these approaches is that they try to minimize some mathematical function (i.e. the number of roles), without considering that such a role-set would be not usable by system administrators. In Section 3.3, we will highlight the drawbacks of these variants, while introducing the relevance of the "business meaning". Indeed, in this thesis we will introduce several goodness/interestingness metrics for insightful bottom-up analysis, all of them are based on some business information.

3.1 Formal Definition of Candidate Role-Sets

We now provide some definitions required to formally describe the role engineering problem:

Definition 3.1 (Configuration) Given an access control system, we refer to its *configuration* as the tuple $\varphi = \langle USERS, PERMS, UP \rangle$, that is the set of all existing users, permissions, and the corresponding relationships between them within the system.

A system configuration represents the user authorization state before migrating to RBAC, or the authorizations derivable from the current RBAC implementation. In the latter case, the user-permission relationships can be derived

as:

$$UP = \{\langle u, p\rangle \mid \exists r \in ROLES : u \in ass\_users(r) \ \wedge \ p \in ass\_perms(r)\}$$

Definition 3.2 (State) An RBAC *state* is a tuple $\psi = \langle ROLES, UA, PA\rangle$, namely an instance of all the sets characterizing the RBAC model.

An RBAC state is used to implement a system configuration. Indeed, the role engineering goal is to find the "best" state that correctly describes a given configuration. In particular we are interested in finding the following kind of states:

Definition 3.3 (Candidate Role-Set) Given an access control system configuration φ, a *candidate role-set* is the RBAC state ψ that "covers" all possible combinations of permissions possessed by users according to φ, namely a set of roles whose union of permissions matches exactly with the permissions possessed by the user. Formally

$$\forall u \in USERS, \exists R \subseteq ROLES : \bigcup_{r \in R} ass\_perms(r) = \{p \in PERMS \mid \langle u, p\rangle \in UP\}.$$

Definition 3.4 (Cost Function) Let Φ, Ψ be respectively the set of all possible system configurations and RBAC states. We refer to the *cost function cost* as

$$cost\colon \Phi \times \Psi \to R^+$$

where R^+ indicates positive real numbers including 0; it represents an administration cost estimate for the state ψ used to implement the configuration φ.

The administration cost concept was first introduced in [6]. Leveraging the cost metric makes it possible to find candidate role-sets with the lowest possible effort to administer the resulting RBAC state.

Definition 3.5 (Optimal Candidate Role-Set) Given a configuration φ, an *optimal candidate role-set* is the the RBAC state ψ that simultaneously represents a candidate role-set for φ, and minimizes the cost function $cost(\varphi, \psi)$.

The main goal related to mining roles is to find optimal candidate role-sets. In the next section we focus on optimizing a particular cost function. Let *cost* indicate the number of needed roles, the role mining objective becomes to find a candidate role-set having the minimum number of roles for a given system configuration.

3.2 Role Mining Problems

3.2.1 Basic and δ-approximated Role Mining Problem

In this section, we present the basic Role Mining Problem (RMP) and one of its variants, δ-approx RMP [75].

Definition 3.6 (Basic Role Mining Problem (RMP)) Given a configuration φ, find the candidate role set $\psi = \langle ROLES, UA, PA \rangle$ that minimizes the number of roles $|ROLES|$.

Given the user-permission matrix, the basic RMP asks us to find a user-to-role assignment *UA* and a role-to-permission assignment *PA* such that *UA* and *PA* exactly describe *UP* while minimizing the number of roles. The basic RMP corresponds therefore to the optimal candidate role-set (Definition 3.5) with cost function defined as:

$$cost(\varphi, \psi = \langle ROLES, UA, PA \rangle) := |ROLES|$$

While exact match is a good thing to have, at times we may be satisfied with an approximate match. For example, consider a case where we have 1000 users and 100 permissions. The size of *UP* is 5000 (i.e., 5000 user-permission assignments are allowed out of the possible 100,000). Now, suppose 100 roles are required to exactly match the given user-permission data. However, if we allow approximate matching—i.e., if it is good enough to match 99% of the matrix (4950 of the user-permission assignments), assume that the minimum number of roles required is only 60. As long as we do not add any spurious permissions, the second case is clearly better than the first, since we significantly reduce the number of roles. This significantly reduces the burden of maintenance on the security administrator while leaving only a few user-permission assignments uncovered. Also, any given user-permission assignment is only a snapshot of the current state of the organizations. Permissions and (to a lesser extent, Roles) are dynamic. Thus while exact match may be the best descriptor in the static case, it is probably not good for the dynamic case. Approximate match might be a prudent choice for dynamic data.

We now introduce the notion of δ-consistency between *UA*, *PA* and *UP*, and then the δ-approximated RMP.

Definition 3.7 (δ-Consistency) A given user-to-role assignment *UA*, role-to-permission assignment *PA* and user-to-permission assignment *UP* are δ- consistent if and only if

$$\|M(UA) \times M(PA) - M(UP)\|_1 \leq \delta$$

where $M(UA)$, $M(PA)$ and $M(UP)$ denote the matrix representation of UA, PA and UP respectively.

Note that the L1 norm is expanded to matrices as follows: suppose A and B are matrices in $X^{n\times m}$, then:

$$\|A-B\|_1 = \sum_{i=1}^{n} \|a_i - b_i\| = \sum_{i=1}^{n}\sum_{j=1}^{m} \left|a_{ij} - b_{ij}\right|$$

Essentially, the notion of δ-consistency allows us to bound the degree of difference between the user-to-role assignment UA, role-to-permission assignment PA and user-to-permission assignment UP. For UA, PA, and UP to be σ-consistent, the user-permission matrix generated from UA and PA should be within δ of UP. We now define the approximate Role Mining Problem using δ- consistency.

Definition 3.8 (δ-approximated RMP) Given a configuration φ, and a threshold δ, find a candidate role set $\psi = \langle ROLES, UA, PA\rangle$ that is $\delta - consistent$ with UP and that minimizes the number of roles, $|ROLES|$.

Therefore, the basic Role Mining Problem defined earlier is a special case of the δ-approx RMP (with δ set to 0).

3.2.2 Edge Role Mining Problem

Another possibility is to discover a candidate role set in such a way that the total number of user-to-role assignments and role-to-permission assignments ($|UA|+|PA|$) is minimal. This could potentially be of more practical value from the perspective of the security administrator as less number of assignments need to be managed. We refer to this as the minimum edge role mining problem.

Definition 3.9 (Edge Role Mining Problem (RMP)) Given a configuration φ, find the optimal candidate role set $\psi = \langle ROLES, UA, PA\rangle$ with respect to the cost function $cost(\varphi, \psi = \langle ROLES, UA, PA\rangle) := |UA|+|PA|$.

Although, intuitively, it may appear that the basic-RMP and edge-RMP are related (with minor modifications in the solution of one working for the other), in fact, the two are independent. In other words, by solving the basic-RMP one does not necessarily solve the edge-RMP, or vice versa. For example, given $ROLES$, by simply increasing the number of users assigned to each role, one may end up with larger $|UA|+|PA|$, which could be minimized with a different (larger) set of roles.

3.3 Minimality against Business Meaning

One may note that the most useful set of roles may be different from the minimal set of roles or the minimum set of edges. It is analogous to employing the best normalization in designing a database schema: From the practical point of view, one may denormalize the database to improve the query response. Similarly, the minimal set of roles gives us a good set of roles to begin with. At least, it shows the bare minimum required to accurately describe the current access control state of the organization.

However, there is an important drawback that regards minimality: the lack of business meaning. Roles discovered by analyzing existing access permissions through role mining algorithms are often no more than a set of permissions with no connection to the business practice. Indeed, the main objective of most of the role mining approaches is only to reduce the number of roles or to simplify the access control administration from a system perspective. But organizations are unwilling to deploy roles they cannot understand, even though such roles are limited in number.

By just trying to minimize the number of roles, elicited roles could be meaningless from the administrator perspective. That is, each role will only correspond to a group of users and permissions, without a well defined business meaning. Obviously, such roles will be difficult to manage, they will be error prone, and they could have some difficulties in being inserted within the risk management framework in use within the organization. Indeed, once a role is created, its life-cycle will follow the life-cycle of the company: new users or new permissions can be added, old ones can be removed or replaced, users can change their job position and subsequently the needed permissions, etc. This continuous adjustment of access control information typically introduces “noise” within the data—namely, permissions exceptionally or accidentally granted or denied—, thus increasing the risk of making mistakes when managing the access control system. As a consequence, it is important to create roles that administrators can easily understand and manage [9].

4

Practical and Usable Approaches to the Role Mining

Role Based Access Control (RBAC) is the de facto standard in access control models, and is widely used in many applications and organizations of all sizes. However, the task of finding an appropriate set of roles, called role engineering, remains the most challenging roadblock to effective deployment. In recent years, this problem has attracted a lot of attention, with several bottom-up approaches being proposed, under the field of role mining. However, most of these theoretical approaches cannot be directly applied to large scale datasets, which is where they are most necessary. Indeed, such algorithms have a complexity that is not linear compared to the number of users or permissions to analyze [6, 29, 78].

The number of elicited candidate roles as well as the running time become an issue, especially when the role engineering task is periodically and frequently performed as part of the life-cycle of roles [8]. To overcome these limitations, role mining products use a *Divide et Impera* approach. For example, administrators that use Oracle Identity Analytics[1] manually separates large numbers of users into more manageable groups, called "waves", for the purpose of defining roles. This is accomplished by first dividing users into business units based on their managers, departments, divisions, or other attributes. Then, these business units are grouped into waves (usually four to six business units per wave), which are independently analyzed using role mining, clustering and categorization algorithms.

In this chapter, we look at how to make role mining *practical* and *usable* for actual deployment. We first introduce a strategy for the reduction of the

[1] `http://www.oracle.com/us/products/middleware/identity-management/oracle-identity-analytics/index.html`

role mining complexity by pruning unstable assignments. In particular, we introduce a graph describing the existing user-permission assignments, and a pruning operation that corresponds to the identification of unstable roles. Then, we propose a strategy to analyze an existing dataset, and to decompose the role mining task into several sub-tasks. Each of these sub-tasks is executed on a partition of the dataset that is homogeneous from an enterprise perspective. This is quite different from the approach of Oracle Identity Analytics that is completely manual, and not business driven. Finally, we introduce a six steps methodology that makes role mining scalable without sacrificing on utility and is agnostic to the actual role mining technique used. This last proposed approach overcomes the drawbacks of the divide et impera approaches, and can be combined with any existing role mining algorithm.

4.1 Pruning Techniques to Reduce the Role Mining Complexity

This section formally describes a strategy for the reduction of the role mining complexity by pruning *unstable* assignments. We first explain the mapping between the role engineering problem, the biclique cover and the clique partition problems, as in [29]. Then we introduce a pruning methodology, and prove the relation between the degree of graph nodes and their instability. Finally, we explain how to identify unstable assignments, and we report on the experimental results.

Definition 4.1 (Role Weight) Given a role $r \in ROLES$, let P_r and U_r be the sets of permissions and users associated to r, that is $P_r = \{p \in PERMS \mid \langle p, r\rangle \in PA\}$ and $U_r = \{u \in USERS \mid \langle u, r\rangle \in UA\}$. We indicate with $w : ROLES \to \mathbb{R}$ the *weight* function of roles, defined as

$$w(r) = c_u \left|U_r\right| \oplus c_p \left|P_r\right|, \tag{4.1}$$

where the operator "$\oplus$" represents a homogeneous[2] binary function of degree 1, while c_u and c_p are real numbers greater than 0.

[2]A function is *homogeneous* when it has a multiplicative-scaling behavior, that is if the argument is multiplied by a factor, then the result is multiplied by some power of this factor. Formally, if $f : V \to W$ is a function between two vector spaces over a field F, then f is said to be homogeneous of degree k if $f(\alpha\mathbf{v}) = \alpha^k f(\mathbf{v})$ for all nonzero $\alpha \in F$ and $\mathbf{v} \in V$. When the vector spaces involved are over the real numbers, a slightly more general form of homogeneity is often used, requiring only that the previous equation holds for all $\alpha > 0$. Note that any linear function is homogeneous of degree 1, by the definition of linearity. Since we require functions with two parameters, we can alternatively state that the multiplication must be *distributive* over "$\oplus$". Thus, an example of valid "$\oplus$" operator is the sum.

In the following, we use the role weight as an indicator of the "stability" of a role:

Definition 4.2 (Role Stability) Let $r \in ROLES$ be a given role, w be the role weight function, and $t \in \mathbb{R}$ be a real number that we refer to as a "threshold". We say that r is *stable* with respect to t if $w(r) > t$. Otherwise, r is *unstable*.

Definition 4.3 (Assignment Stability) Let the pair $\langle u, p\rangle \in UP$ be a given assignment, and $t \in \mathbb{R}$ be a real number that we refer to as a "threshold". Let $R_{\langle u,p\rangle}$ be the set of roles that can be used to manage the assignment $\langle u, p\rangle$, namely $R_{\langle u,p\rangle} = \{r \in ROLES \mid \langle u, r\rangle \in UA,\ \langle p, r\rangle \in PA\}$, and let w be the role weight function. We say that $\langle u, p\rangle$ is *stable* with respect to t if it belongs to at least one stable role, namely $\exists r \in R_{\langle u,p\rangle} : w(r) > t$. Otherwise, the assignment is *unstable*, that is $\forall r \in R_{\langle u,p\rangle} : w(r) \leq t$.

4.1.1 Role Engineering and Biclique Cover

We first observe that a given configuration $\varphi = \langle USERS, PERMS, UP\rangle$ can be represented by a bipartite graph

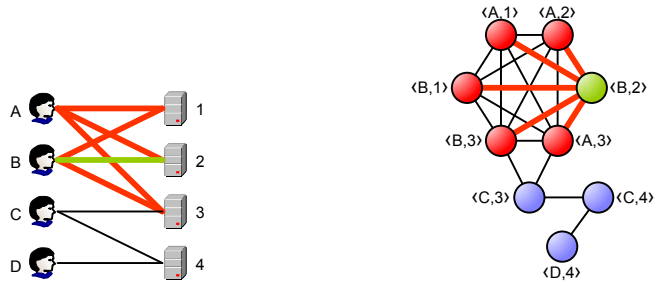
$$G = \big\langle V_1 \cup V_2,\ E\big\rangle = \big\langle USERS \cup PERMS,\ UP\big\rangle, \tag{4.2}$$

where two vertices $u \in USERS$ and $p \in PERMS$ are connected by an edge if the user u is granted permission p, namely $\langle u, p\rangle \in UP$. A biclique cover of the graph G univocally identifies a candidate role-set $\psi = \langle ROLES, UA, PA\rangle$ for the configuration φ. Indeed, every biclique identifies a role, and the vertices of the biclique identify the users and the permissions assigned to this role [10, 11, 29]. Thus, finding the optimal role-set is equivalent to identifying the biclique cover such that the corresponding roles are optimal.

By starting from the bipartite graph G, it is possible to construct an undirected unipartite graph G' in the following way: each edge in G (i.e., an assignment of UP) becomes a vertex in G', and two vertices in G' are connected by an edge if and only if the endpoints of the corresponding edges of G induce a biclique. To ease the exposition, we define the function B: $UP \to 2^{UP}$ that indicates all edges in UP which induces a biclique together with the given edge, namely:

$$B(\langle u, p\rangle) = \big\{\langle u', p'\rangle \in UP \mid \langle u, p'\rangle, \langle u', p\rangle \in UP \ \wedge\ \langle u, p\rangle \neq \langle u', p'\rangle\big\}. \tag{4.3}$$

Note that two edges $\omega_1 = \langle u_1, p_1\rangle$ and $\omega_2 = \langle u_2, p_2\rangle$ of UP that share the same user (that is, $u_1 = u_2$) or the same permission (that is, $p_1 = p_2$) induce a biclique. Also, $\langle u_1, p_1\rangle$ and $\langle u_2, p_2\rangle$ induce a biclique if the pair $\langle u_1, p_2\rangle, \langle u_2, p_1\rangle \in$



(a) Bipartite view

(b) Unipartite view

Figure 4.1 An example of a given assignment (green) and the set of assignments that induce a biclique with it (red), in both graph G and G'.

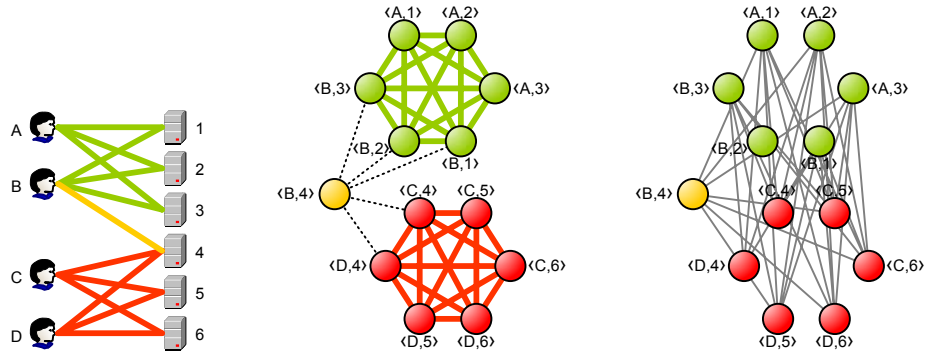
UP exist. Moreover, given $\omega_1, \omega_2 \in UP$, it can be easily verified that $\omega_1 \in B(\omega_2) \iff \omega_2 \in B(\omega_1)$ and $\omega_1 \in B(\omega_2) \implies \omega_1 \neq \omega_2$. Therefore, the undirected unipartite graph G' induced from G can be formally defined as:

$$G' = \langle V', E' \rangle = \left\langle UP,\ \left\{ \langle \omega_1, \omega_2 \rangle \in UP \times UP \mid \omega_1 \in B(\omega_2) \right\} \right\rangle \tag{4.4}$$

In this way, the edges covered by a biclique of G induce a clique in G'. Thus, every biclique cover of G corresponds to a collection of cliques of G' such that their union contains all of the vertices of G'. From such a collection, a clique partition of G' can be obtained by removing any redundantly covered vertex from all but one of the cliques it belongs to. Similarly, any clique partition of G' corresponds to a biclique cover of G.

To clarify this concept, Figure 4.1 show a simple example, where $USERS = \{A, B, C, D\}$, $PERMS = \{1, 2, 3, 4\}$, and $UP = \{\langle A, 1\rangle, \langle A, 2\rangle, \langle A, 3\rangle, \langle B, 1\rangle, \langle B, 2\rangle, \langle B, 3\rangle, \langle C, 3\rangle, \langle C, 4\rangle, \langle D, 4\rangle\}$. In the figure, the assignment $\langle B, 2\rangle$ represents an edge in the bipartite graph (Figure 4.1(a)) and a vertex in the unipartite graph (Figure 4.1(b)). The figures show in red and thicker lines all the assignments that induce a biclique with $\langle B, 2\rangle$, according to Equation 4.3; for example, $\langle B, 3\rangle$ share the same user of $\langle B, 2\rangle$, while $\langle A, 1\rangle$ induce a biclique with $\langle B, 2\rangle$ since the assignments $\langle B, 1\rangle$ and $\langle A, 2\rangle$ exist.

It is known that finding a clique partition of a graph is equivalent to finding a coloring of its complement [10, 11, 29]. To this aim, let the graph $\overline{G}'$ made up of the same vertices of G', but edges of $\overline{G}'$ are the complement of edges of G'. Given an assignment $\omega \in UP$, we indicate with $\overline{B}(\omega)$ the assignments that do *not* induce a biclique together with ω, namely



(a) Biclique cover in G

(b) Clique partition in G'

(c) Vertex coloring in $\overline{G}'$

Figure 4.2 Relationship among biclique cover, clique partition, and vertex coloring.

$$\overline{B}(\omega) = (UP \setminus \{\omega\}) \setminus B(\omega). \tag{4.5}$$

Hence, the graph $\overline{G}'$ can be formally defined as:

$$\overline{G}' = \langle \overline{V}', \overline{E}' \rangle = \left\langle UP,\ \{\langle \omega_1, \omega_2 \rangle \in UP \times UP \mid \omega_1 \in \overline{B}(\omega_2)\} \right\rangle \tag{4.6}$$

Any coloring of the graph $\overline{G}'$ identifies a candidate role-set of the given system configuration $\varphi = \langle USERS, PERMS, UP \rangle$, from which we have generated G. Thus, finding a proper coloring for $\overline{G}'$ means finding a candidate role-set that covers all possible combinations of permissions possessed by users according to φ; namely, a set of roles such that the union of related permissions matches exactly with the permissions possessed by the users.

The above-mentioned properties are graphically depicted in Figure 4.2. In particular, Figure 4.2(a) shows a possible biclique cover. This cover is composed by 3 different bicliques: $\{\langle A,1\rangle, \langle A,2\rangle, \langle A,3\rangle, \langle B,1\rangle, \langle B,2\rangle, \langle B,3\rangle\}$ (green), $\{\langle B,4\rangle\}$ (yellow), and $\{\langle C,4\rangle, \langle C,5\rangle, \langle C,6\rangle, \langle D,4\rangle, \langle D,5\rangle, \langle D,6\rangle\}$ (red). Figure 4.2(b) represents the same information in the unipartite view in terms of clique partition. Figure 4.2(c) demonstrates that the same information represents a vertex coloring in the complement of the unipartite graph. Edges in G belonging to the same biclique have the same color, and vertices in G' and $\overline{G}'$ have the same color of their corresponding edges in G. Moreover, vertices in G' that belong to the same clique are connected with an edge

with the same color of their vertices, while dashed lines indicate that their endpoints do not belong to any clique of the chosen partition.

4.1.2 Methodology

To generate a candidate role-set that is stable and easily analyzable, we split the problem in three steps:

Step 1 Define a weight-based threshold.

Step 2 Catch the unstable user-permission assignments.

Step 3 Restrict the problem of finding a set of roles that minimizes the administration cost function by only using stable user-permission assignments.

In particular, we introduce a pruning operation on the vertices of $\overline{G}'$ that corresponds to identifying unstable user-permission assignments. We suggest to not manage these assignments with roles, but to directly assign permission to users or, equivalently, to create "special" roles composed by only one permission. In this way, we are able to limit the presence of unstable roles.

Moreover, we will show that the portion of the graph that survives after the pruning operation can be represented as a graph $\overline{G}'$ with a limited degree. Since the third step corresponds to coloring $\overline{G}'$, the information about the degree can be leveraged to select an efficient coloring algorithms among those available in the literature that make assumptions on the degree. The choice of which algorithm to use depends on the definition of the administration cost function.

It is also important to note that when the graph G is not connected, it is possible to consider any connected component as a separate problem. Hence, the union of the solutions of each component will be the solution of the original graph, as proven in the following lemma:

Lemma 4.1 *A biclique cannot exist across two or more disconnected components of a bipartite graph G.*

PROOF Let $G_1, \ldots, G_m$ be the disconnected components of G. We will show that a biclique across two components G_i and G_k, with $i \neq k$, cannot exist. Let $\mathcal{B}$ the biclique across G_i and G_k, with $i \neq k$, and let $\mathcal{B}_i$ and $\mathcal{B}_k$ be the sets of vertices of $\mathcal{B}$ belonging respectively to G_i and G_k. From the biclique definition, it follows that edges between the two vertex sets of $\mathcal{B}_i$ and $\mathcal{B}_k$ must exist. But it is a contradiction, since G_i and G_k are two disconnected components, hence edges between their vertices cannot exists. ■

Since a biclique corresponds to a role, the previous lemma states that a role r, made up of users U_r and permissions P_r, cannot exist if all the users in U_r do not have all the permissions in P_r. If this were the case, we would have introduced some user-permission relationships that were not in the configuration $\varphi = \langle USERS, PERMS, UP \rangle$. This lemma has an important implication:

Theorem 4.1 *If G is disconnected, the union of the biclique covers of each component of G is a biclique cover of G.*

PROOF From Lemma 4.1, we know that a biclique across two or more disconnected components of G cannot exist. Thus, each disconnected component has a biclique cover that cannot intersect with the biclique cover of any other component. Therefore, the union of these biclique covers will be a cover of G. ■

As a main consequence of the theorem, if the graph G is disconnected, we can study each component independently. In particular, we can use the union of the biclique cover of the different components to build a biclique cover of G. According to what we will see in the next section, we can use this result to limit the degree of $\overline{G}'$ when the bipartite graph G is disconnected.

Unstable Assignment Identification

In our model, the role mining problem corresponds to finding a proper coloring for the graph $\overline{G}'$. Depending on the cost function used, the optimal coloring can change. For instance, if the cost function is defined as the total number of roles (as in the basic RMP), the optimal coloring is the one which uses the minimum number of colors. In this section we will analyze the degree of the graph $\overline{G}'$ by highlighting how this information can affect the assignment stability and, as a consequence, the administration effort.

According to Equation 4.6 the degree of the graph $\overline{G}'$ can be expressed as:

$$\Delta(\overline{G}') = \max_{\omega \in UP} |\overline{B}(\omega)| \,. \tag{4.7}$$

To understand the relation between the graph degree and the stable assignment identification problem, it is useful to recall the graph meaning in terms of RBAC semantic. A vertex of $\overline{G}'$ is a user-permission relationship in the set UP. An edge in $\overline{G}'$ between two vertices ω_1 and ω_2 exists if the corresponding user-permission relationships cannot be in the same role, due to the fact that the user in ω_1 does not have the permission in ω_2, or the user in ω_2 does not have the permission in ω_1. Consequently, a vertex of $\overline{G}'$ that has a high degree means that this vertex cannot be colored using the same colors

of a high number of other vertices. In other words, this user-permission relationship cannot be in the same role together with a high number of other user-permission relationships.

The previous considerations have an important aftermath: if a user-permission assignment cannot be in the same role together with a high number of other user-permission assignments, it will belong to a role with few user-permission assignments, and we can estimate the maximal weight of such a role. Hence, we can prune those user-permission assignments which can only belong to roles with a weight that is lower than a fixed threshold.

In particular, suppose that for each edge $\omega \in UP$ of the bipartite graph G there are at least d other edges such that the corresponding endpoints induce a biclique together with the endpoints of ω. In this case, every edge of G will not be in biclique with less than $|UP| - d$ other edges, according to the following lemma:

Lemma 4.2 *Let UP be the set of edges of the bipartite graph G. Then:*

$$\forall \omega \in UP,\ |B(\omega)| > d \implies \Delta(\overline{G}') \leq |UP| - d$$

PROOF Since $\forall \omega \in UP,\ |B(\omega)| > d$, the following holds: $\forall \omega \in UP,\ |\overline{B}(\omega)| < |UP| - d - 1$. The proof follows from $\Delta(\overline{G}') = max_{\omega \in UP} |\overline{B}(\omega)|$. ■

Thus, given a suitable value for d, the idea is to prune the graph $\overline{G}'$ by deleting the vertices that have a degree higher than $|E(G)| - d$. This corresponds to pruning edges in G that induce a biclique with at most d other edges. Moreover:

Theorem 4.2 *The pruning operation based on removing from $\overline{G}'$ vertices ω such that $|B(\omega)| \leq d$ will prune only user-permission assignments that cannot belong to any role $r \in ROLES$ such that $w(r) > d \times (c_U \oplus c_P)$.*

PROOF Let ω be the assignment we would like to prune since $|B(\omega)| \leq d$. The corresponding vertex in $\overline{G}'$ has a degree strictly greater than $|UP| - d$. Such a vertex cannot be colored with the colors of his neighbors, thus it can be colored with at most the same colors of the $(|UP| - 1) - (|UP| - d - 1) = d$ remaining vertices. Hence, there exist at most d assignments that can belong to the same role ω belongs to. Let r be such a role. According to Definition 4.1, the maximal weight of r will be $(c_U \times d) \oplus (c_P \times d) = d \times (c_U \oplus c_P)$, since each assignment belonging to r could add at most one user and one permission to the role. ■

Note that many coloring algorithms known in the literature make assumptions on the degree of the graph. Since our pruning approach limits the degree

of $\overline{G}'$, it allows for an efficient application of this class of algorithms. Without our pruning operation, the degree of the graph $\overline{G}'$ could be high, up to $|UP|-1$. This is the case when a user-permission assignment that must be managed alone in a role exists. Note also that when the graph G is disconnected in two or more components, any edge of one component does not induce a biclique together with any edge of the other components. Thus, in these cases $\Delta(\overline{G}')$ is very high. But, because Theorem 4.1, we can split the problem by considering the different components distinctly, and then join the results of each component.

4.1.3 Measuring Role Engineering Complexity

In this section we will discuss about the application of the *clustering coefficient* in RBAC (Definition 2.8). In particular, we will show that the clustering coefficient can measure the complexity of the identification and selection of the roles required to manage existing user-permission assignments. We will show that the pruning operation proposed in Section 4.1.2 not only does identify the user-permission assignments that are unstable, but it is able to simplify the identification and selection of stable roles among all the candidate roles as well. The main result is that stable assignments may have a low value for clustering coefficient due to the presence of unstable assignments. A low value for clustering coefficient is a synonym for high role engineering complexity. This can be summarized with the following statement:

assignments with unstable neighbors
$\Longrightarrow$ *low clustering coefficient*
$\Longrightarrow$ *complex role engineering task.*

Clustering Coefficient in G'

Let G' be the unipartite graph derived from user-permission assignments UP according to Equation 4.4. Consequently, Equation 2.3 becomes:

$$\tau(\omega) = \left|\left\{\langle \omega_1, \omega_2\rangle \in UP \times UP \mid \omega_1, \omega_2 \in B(\omega),\ \omega_1 \neq \omega_2\right\}\right|, \tag{4.8}$$

namely $\tau(\omega)$ is the set of all possible pairs of elements in UP that both induce a biclique with ω. Further, Equation 2.2 becomes:

$$\delta(\omega) = \left|\left\{\langle \omega_1, \omega_2\rangle \in UP \times UP \mid \omega_1, \omega_2 \in B(\omega),\ \omega_1 \in B(\omega_2)\right\}\right|, \tag{4.9}$$

namely $\delta(\omega)$ is the set of all possible pairs of elements in UP that both induce a biclique with ω, and that also induce a biclique with each other.

The clustering coefficient index (Definition 2.8) of the graph G' derived from an access control system configuration is thus defined as its minability index:

Definition 4.4 (Minability)

$$\mathcal{M}(UP) := C(G') = \frac{1}{|UP|} \sum_{\omega \in UP} c(\omega), \tag{4.10}$$

where $c(\omega)$ is the local clustering coefficient of ω (see Equation 2.4) defined as:

$$c(\omega) = \begin{cases} \dfrac{\delta(\omega)}{\tau(\omega)}, & \tau(\omega) \neq 0; \\ 1, & \text{otherwise.} \end{cases} \tag{4.11}$$

The value of $c(\omega)$ quantifies how close ω and its neighbors are to being a biclique. In our model, this corresponds to measure how close ω and its neighbors are to being a role. Hence, $C(G')$ quantifies how well the bipartite graph, induced by the user-permission relationships *UP*, is coverable with distinct bicliques. That is, the easiness of identifying a candidate role set for the analyzed data. Notice that, according to Equation 4.11, when a user-permission assignment does not induce a biclique with any other assignment, or it induces biclique with just one another assignment, its local clustering coefficient is conventionally set to 1. This case is identified by $\tau(\omega) = 0$. Moreover, Definition 4.4 and Equation 4.11 only require *UP* and $B(\cdot)$ to be provided, by neglecting whether we are considering the bipartite or the unipartite graph. Thus, in the remainder of this chapter we indicate with both $C(G')$ and $C(G)$ the global clustering coefficient of the given system configuration represented by *UP*, while $c(\omega)$ is the local clustering coefficient without specifying G or G'.

In the remaining of this section we explain the relationship between the clustering coefficient and the complexity of the role mining problem. In particular, let G the bipartite graph set up from *UP* according to Equation 4.2. Given a role $\overline{r} \in ROLES$, let $P_{\overline{r}} = \{p \in PERMS \mid \langle p, \overline{r} \rangle \in PA\}$ be the set of its assigned permissions, and $U_{\overline{r}} = \{u \in USERS \mid \langle u, \overline{r} \rangle \in UA\}$ be the set of its assigned users. If the following equation holds

$$\nexists U \subseteq USERS, \nexists P \subseteq PERMS \,:\, U \times P \subseteq UP,\ U_{\overline{r}} \times P_{\overline{r}} \subset U \times P, \tag{4.12}$$

then the role $\overline{r}$ represents a maximal biclique in G. Indeed, according to its definition, a maximal biclique in G is a pair of vertex sets $U \subseteq USERS$ and $P \subseteq$

PERMS that induces a complete subgraph, namely $\forall u \in U,\ \forall p \in P\ :\ \langle u, p\rangle \in UP$, and is not a subset of the vertices of any larger complete subgraph, that is $\nexists U' \subset USERS$ and $\nexists P' \subset PERMS$ such that $\forall u \in U',\ \forall p \in P'\ :\ \langle u, p\rangle \in UP$ and contextually $U' \subset U$ and $P' \subset P$.

Informally, a role delineated by a maximal biclique is "representative" for all the possible subset of permissions shared by a given set of users. The key observation behind a maximal bicliques in RBAC is that two permissions which always occur together among users should simultaneously belong to the same candidate roles. Moreover, defining roles made up of as many permissions as possible likely minimizes the administration effort of the RBAC system by reducing the number of required role-user assignments.

The following theorem relates the clustering coefficient index to the complexity of the role mining problem in terms of number of maximal bicliques:

Theorem 4.3 *Let $\mathcal{M}$ be the set of all possible maximal bicliques that can be identified in G. Given a user-permission assignment $\omega \in UP$, let $\mathcal{M}(\omega) \subseteq \mathcal{M}$ be the set of all possible maximal bicliques the given user-permission assignment belongs to. Then, the following holds:*

- $c(\omega) = 1 \iff |\mathcal{M}(\omega)| = 1$;
- $c(\omega) = 0 \iff |\mathcal{M}(\omega)| = |B(\omega)|$;
- $c(\omega) \in (0,1) \iff |\mathcal{M}(\omega)| \in (1, |B(\omega)|)$.

PROOF To simplify the notation, given a role $r \in ROLES$ we indicate the set of users assigned to the role with $U_r = \{u \in USERS \mid \langle u, r\rangle \in UA\}$, and the set of permissions assigned to that role with $P_r = \{p \in PERMS \mid \langle p, r\rangle \in PA\}$.

First, we analyze the case $c(\omega) = 1$. Let $\overline{r}$ be a role made up of the users and permissions involved by the assignments ω and $B(\omega)$, formally $U_{\overline{r}} = \{u \in USERS \mid \exists p \in PERMS,\ \langle u, p\rangle \in B(\omega) \cup \{\omega\}\}$ and $P_{\overline{r}} = \{p \in PERMS \mid \exists u \in USERS,\ \langle u, p\rangle \in B(\omega) \cup \{\omega\}\}$. We now demonstrate that at least one maximal biclique exists and it is represented by $\overline{r}$. According to Equation 4.3, $\forall \langle u_1, p_1\rangle, \langle u_2, p_2\rangle \in B(\omega) \cup \{\omega\} \implies \exists \langle u_1, p_2\rangle, \langle u_2, p_1\rangle \in UP$, namely both users u_1, u_2 have permissions p_1, p_2 granted. According to Equation 4.11, $c(\omega) = 1 \implies \tau(\omega) = \delta(\omega)$, thus the previous consideration holds for every possible pair of user-permission relationships in $B(\omega) \cup \{\omega\}$. This means that $B(\omega) \cup \{\omega\} = U_{\overline{r}} \times P_{\overline{r}}$. We now prove by contradiction that $\overline{r}$ is a maximal biclique. If $\overline{r}$ were not a maximal biclique, two sets $U \subseteq USERS$ and $P \subseteq PERMS$ would exist such that $U_{\overline{r}} \times P_{\overline{r}} \subset U \times P \subseteq UP$. Let $\omega = \langle u, p\rangle$. Yet, for each $\langle u', p'\rangle \in (U \times P) \setminus (U_{\overline{r}} \times P_{\overline{r}})$ it can be easily shown that both the assignments $\langle u, p'\rangle, \langle u', p\rangle$ always exists in $U \times P$. Hence, according to Equation 4.3, $\langle u', p'\rangle \in B(\omega)$, meaning that $(U \times P) \setminus (U_{\overline{r}} \times P_{\overline{r}}) = \emptyset$. Therefore, $\overline{r}$ is a maximal biclique. We now demonstrate that another maximal biclique that

contains ω cannot exist. Indeed, if $\overline{r}'$ is a maximal biclique containing ω (i.e., $\omega \in U_{\overline{r}'} \times P_{\overline{r}'}$), for all $\omega' \in U_{\overline{r}'} \times P_{\overline{r}'} \setminus \{\omega\}$ it can be shown that $\omega' \in B(\omega)$. Hence, $\overline{r} = \overline{r}'$. Finally, having only one maximal biclique that contains ω implies that $c(\omega) = 1$. Let $\overline{r}$ be such a maximal biclique. Since it is the only maximal biclique, for each pair $\omega_1, \omega_2 \in U_{\overline{r}} \times P_{\overline{r}}$ such that $\omega_1 \neq \omega_2$ we have $\omega_1 \in B(\omega_2)$. Thus, $\delta(\omega) = \tau(\omega)$, which corresponds to state that $c(\omega) = 1$.

When $c(\omega) = 0$, we now demonstrate that it is possible to identify $|B(\omega)|$ distinct maximal bicliques made up of ω combined with each element of $B(\omega)$. Let $\omega = \langle u, p \rangle$. First, observe that such maximal bicliques are distinct since $c(\omega) = 0 \implies \delta(\omega) = 0$. We want to show that for each $\langle u_i, p_i \rangle \in B(\langle u, p \rangle)$, the role $\overline{r}_i$ is such that $U_{\overline{r}_i} = \{u, u_i\}$ and $P_{\overline{r}_i} = \{p, p_i\}$ is a maximal biclique. We now prove by contradiction that $\overline{r}_i$ is a maximal biclique. If $\overline{r}_i$ were not a maximal biclique, two sets $U \subseteq USERS$ and $P \subseteq PERMS$ would exist such that $\{u, u_i\} \times \{p, p_i\} \subset U \times P \subseteq UP$. Let $\langle u', p' \rangle \in (U \times P) \setminus (\{u, u_i\} \times \{p, p_i\})$. It can be easily shown that $\langle u', p' \rangle \in B(\langle u_i, p_i \rangle)$, thus $\delta(\omega) \neq 0$. But, according to Equation 4.11, this means that $c(\omega) > 0$, which is a contradiction. Moreover, more than $|B(\omega)|$ distinct maximal bicliques that contain ω cannot exist. Indeed, let $n \in \mathbb{N} : n > |B(\omega)|$ be the number of the distinct maximal bicliques that contain ω. Let $\overline{r}_i$ indicate the i^{th} maximal biclique, and let $\omega_i \in (U_{\overline{r}_i} \times P_{\overline{r}_i}) \setminus \{\omega\}$. Thus, $\forall i \in 1 \dots n : \omega_i \in B(\omega)$, contradicting the inequality $B(\omega) < n$. We now prove that having $|B(\omega)|$ maximal bicliques implies that $c(\omega) = 0$. Let $\overline{r}_i$ indicate the i^{th} maximal biclique, and let $\omega_i \in (U_{\overline{r}_i} \times P_{\overline{r}_i}) \setminus \{\omega\}$. Since the roles are distinct, $\forall i, j \in 1 \dots |B(\omega)| : i \neq j$ we have that $\omega_i \notin B(\omega_j)$. Thus, $\delta(\omega) = 0$ and, according to Equation 4.11, $c(\omega) = 0$.

Finally, by excluding the previous two cases we merely have that $c(\omega) \in (0, 1) \iff 1 < |\mathcal{M}(\omega)| < |B(\omega)|$. ■

The previous theorem allows us to make some considerations on the complexity of the role mining problem. Given a user-permission assignment ω, the higher its local clustering coefficient is, the less the number of possible maximal bicliques to analyze is. Thus, given two assignments $\omega_1, \omega_2 \in UP$ such that $c(\omega_1) = 1$ and $c(\omega_2) = 0$, it will be more difficult to choose the best maximal biclique to "cover" ω_2 than selecting the best maximal biclique to cover ω_1. Indeed, in the first case we have only one choice, while in the second case we have $|B(\omega_2)|$ choices.

Clustering Coefficient and Vertex Degree

In the previous section we demonstrated that the local clustering coefficient of a given assignment expresses the ambiguity in selecting the best maximal biclique to cover it when finding the best biclique cover. Hereafter, we show

that the local clustering coefficient value and the number of assignments that induce a biclique are bound. In particular, we prove that the presence of unstable assignments decreases the maximum local clustering value allowed for stable assignments. Therefore, keeping unstable assignments within the data to analyze hinders the role engineering process by increasing the ambiguity in selecting the best roles to cover stable assignments.

Theorem 4.4 *Let $\omega \in UP$ be a user-permission assignment such that $|B(\omega)| > 1$. Then, the following holds:*

$$c(\omega) \leq \frac{\underset{\omega' \in B(\omega)}{\text{avg}} \ |B(\omega')| - 1}{|B(\omega)| - 1}. \tag{4.13}$$

PROOF According to its definition, the local clustering coefficient of a vertex in G' is the ratio between its triangles (Equation 4.9) and its triples (Equation 4.8). All the neighbors of a vertex ω are represented by $B(\omega)$. Thus, we have

$$\tau(\omega) = \binom{|B(\omega)|}{2} = \frac{|B(\omega)| \ \big(|B(\omega)| - 1\big)}{2}.$$

Each neighbor pair requires that they are also neighbors between them in order to be a triangle. Thus, each neighbor ω' of ω can belong to at most $|B(\omega')| - 1$ triangles of ω, where '-1' allows for discarding ω among the set of the neighbors of ω'. Therefore, the number of triangles of ω is at most the sum of all the maximal "contributions" of its neighbors, namely

$$\delta(\omega) \leq \frac{1}{2} \sum_{\omega' \in B(w)} |B(\omega')| - 1,$$

where '$1/2$' is required to take into account that each triangle is considered twice. By combining the previous equations, we obtain:

$$c(\omega) = \frac{\delta(\omega)}{\tau(\omega)} \leq \frac{\dfrac{1}{2} \displaystyle\sum_{\omega' \in B(w)} \big|B(\omega')\big| - 1}{\dfrac{|B(\omega)| \, (|B(\omega)| - 1)}{2}} = \frac{\underset{\omega' \in B(\omega)}{\text{avg}} \ |B(\omega')| - 1}{|B(\omega)| - 1},$$

completing the proof. ∎

Notice that $c(\omega) = 1$ means that all the neighbors of ω in G' have, among their neighbors, all the neighbors of ω. Thus, the right side of the inequality in Equation 4.13 is equal to or greater than 1. Similarly, $c(\omega) = 0$ means that

each pair of neighbors of ω are not neighbors among them. Thus, the right side of the inequality in Equation 4.13 is equal to or greater than 0.

Finally, let us assume that all the neighbors of ω have a degree that is lower than the degree of ω, namely $\forall \omega' \in B(\omega) : |B(\omega')| < |B(\omega)|$. Then, $c(\omega) < 1$. This likely happens to assignments that have a high degree and many unstable assignments as neighbors. Hence, unstable assignments make the task of selecting the best maximal clique to cover stable assignments more difficult. From this point of view, unstable assignments are a sort of "noise" within the data, that badly bias any role mining analysis. Indeed, the number of elicited roles may be large when compared to the number of users and permissions, mainly due to noise within the data—namely, permissions exceptionally or accidentally granted or denied. In such a case, classical role mining algorithms discover multiple small fragments of the true role, but miss the role itself [54]. The problem is even worse for roles which cover many user-permission assignments, since they are more vulnerable to noise [57].

In Section 4.1.5 we will show through experiments on real data that the clustering coefficient increases when pruning unstable assignments.

4.1.4 Pruning Algorithms

In the following we describe two different methods to compute, for each assignment in *UP*, the number of assignments that induce biclique with it. Hence, enabling the pruning strategy thoroughly described by Theorem 4.2. We propose two algorithms: the first one is deterministic and has a computational complexity of $O(|UP|^2)$; the second one uses a randomized approach, leading to a complexity of $O(k|UP|)$, where k represents the number of the chosen random samples. Furthermore, we prove a bound for the approximation introduced by the randomized algorithm.

Deterministic Approach

The idea behind the deterministic approach is the following: we scan each assignment $\omega \in UP$ to identify all the neighbors, namely the set $B(\omega)$. In turn, we increase by 1 the neighbor-counter of each assignment in $B(\omega)$ in order to say that "assignments in $B(\omega)$ have one more neighbor, that is ω". This schema is perfectly equivalent to directly associating the value $|B(\omega)|$ to ω, without increasing the complexity. Yet, it can be easily randomized, as we will see in the next section.

We now show that computing the set of all neighbors of an assignment $\omega = \langle u, p \rangle$ just requires a search on *UP* for all the users possessing the permission p and all the permissions possessed by u. In particular, the following lemma

4.1 The deterministic algorithm to prune unstable assignments

```
 1: procedure CountNeighbors(UP)
 2:     for all ⟨u, p⟩ ∈ UP do
 3:         for all ⟨u′, p′⟩ ∈ Neighbors(⟨u, p⟩, UP) do
 4:             count[⟨u′, p′⟩] ← count[⟨u′, p′⟩] + 1
 5:         end for
 6:     end for
 7:     return count[·]/|UP|
 8: end procedure

 9: procedure Neighbors(⟨u, p⟩, UP)
10:     U ← {u′ ∈ USERS | ⟨u′, p⟩ ∈ UP}
11:     P ← {p′ ∈ PERMS | ⟨u, p′⟩ ∈ UP}
12:     return (U × P \ {⟨u, p⟩}) ∩ UP
13: end procedure
```

holds:

Lemma 4.3 *Given an assignment $\omega = \langle u, p\rangle \in UP$, let $U_\omega = \{u' \in USERS \mid \langle u', p\rangle \in UP\}$ be the set of all users possessing the corresponding permission, and $P_\omega = \{p' \in PERMS \mid \langle u, p'\rangle \in UP\}$ be the set of all permissions possessed by the corresponding user. Then $B(\omega) = (U_\omega \times P_\omega) \cap UP$.*

PROOF First, we prove that $B(\omega) \subseteq U_\omega \times P_\omega$. By contradiction, suppose that an assignment $\langle u', p'\rangle \in UP$ exists such that $\langle u', p'\rangle \in B(\omega)$ but $u' \notin U_\omega$ and/or $p' \notin P_\omega$. According to Equation 4.3, $\langle u', p'\rangle \in B(\omega)$ implies one of the following cases: 1) $u' = u$; 2) $p' = p$; 3) $\exists\langle u, p'\rangle, \langle u', p\rangle \in UP$. In all these three cases there is a contradiction, since by construction of P_ω, U_ω, there must be $p' \in P_\omega$ and $u' \in U_\omega$. Finally, by intersecting $U_\omega \times P_\omega$ with UP we discard all the assignments that do not exist. ■

Lemma 4.3 is used to define the procedure NEIGHBORS in Algorithm 4.1. Line 10 computes all possible users possessing the given permission, Line 11 computes all possible permissions assigned to given user, while Line 12 eliminates from the Cartesian product of these sets all the assignments that not exist within UP. Note that NEIGHBORS has a complexity of $O(|UP|)$. Indeed, both Line 10 and Line 11 can be executed in $O(|UP|)$ by simply scanning over all the assignments. In the same way, the intersection of Line 12 can be executed in $O(|UP|)$.

CountNeighbors implements the described counting strategy. The loop from Line 2 to Line 6 scans all possible assignments in order to check their neighborhood. Lines from 3 to 5 scan all the neighborhood of the current assignment to increment the corresponding neighbor-counter $count[\cdot]$. Notice that Line 4 can be performed in $O(1)$, while the inner loop in $O(|UP|)$ and the outer loop in $O(|UP|)$. Hence, the computational complexity of CountNeighbors is $O(|UP|^2)$. Line 7 gives the resulting neighbor-counts. All the values are *normalized* by dividing them by $|UP|$. In this way, we assign a value to each assignment that ranges from 0 to 1, thus the threshold d must range in this interval as well.

Finally, to implement our pruning strategy, we only need a procedure that searches for assignments such that $count[\omega] \leq d$. It is reasonable to give an efficient implementation for it with a computational complexity of $O(log|UP|)$, for instance through a binary tree. However, this requires $count[\cdot]$ to be sorted at the end of the procedure CountNeighbors. This takes $O(|UP|\, log|UP|)$, hence without changing the complexity of the procedure CountNeighbors.

It is very important to note that the neighbor-counters are inferred with only one CountNeighbors run, that has a complexity of $O(|UP|^2)$. In turn, by changing a threshold d that does not require the complete re-imputation of neighbor-counters, it is possible to generate m versions of the dataset in $O(mlog|UP|)$. Each run can be subsequently analyzed by trying to find the one that better reaches a certain target function. The tuning of the threshold d depends on the final objective of the data analysis problem. First, we can define a metric that measures how well the objective has been reached. Then, this metric can be used to evaluate the imputed dataset. This can be an iterative process, executed several times with different thresholds, thus choosing the threshold value that provides the best result. Section 4.1.5 shows a practical application of this methodology in a real case.

Randomized Approach

In the previous section we offered an algorithm that computes the number of neighbors for each assignment in a time $O(|UP|^2)$. Then, in $O(log|UP|)$ it is possible to identify those assignments that have a number of neighbors below the threshold, namely unstable assignments. In the following we present an alternative algorithm to be used in place of procedure CountNeighbors of Algorithm 4.1, which compute in $O(k|UP|)$ an *approximated* neighbor-counter value for the assignments, where k is a parameter that can be arbitrarily chosen. Moreover, we will show how to select the best value for k, and, when $k \ll |UP|$, it achieves good results in a significantly shorter time. Notice that the pruning procedure can still be performed in $O(log|UP|)$ only if the

4.2 The randomized algorithm to prune unstable assignments

```
1: procedure RandomizedCountNeighbors(UP, k)
2:     for all i = 1...k do
3:         ⟨u, p⟩ ← choose an assignment in UP uniformly at random
4:         for all ⟨u', p'⟩ ∈ Neighbors(⟨u, p⟩, UP) do
5:             count[⟨u', p'⟩] ← count[⟨u', p'⟩] + 1
6:         end for
7:     end for
8:     return count[·]/k
9: end procedure
```

neighbor-counters are sorted at the end of the procedure RandomizedCountNeighbors. Since this operation requires $O(|UP| \, log \, |UP|)$, the complexity of RandomizedCountNeighbors does not change if $log|UP| = O(k)$.

Algorithm 4.2 describes RandomizedCountNeighbors as an alternative approach for the procedure CountNeighbors of Algorithm 4.1. These two procedures have the same structure, apart from one aspect: instead of checking the neighborhood of all assignments in *UP*, we select only k assignments uniformly at random (see Line 3). The rest of the algorithm is exactly the same of the deterministic one, apart from Line 8 that normalizes all the counters by dividing them by k. Therefore, RandomizedCountNeighbors has a computational complexity of $O(k\,|UP|)$.

The following theorem demonstrates the bound on the approximation introduced by RandomizedCountNeighbors:

Theorem 4.5 *Let $\omega = \langle u, p \rangle$ be an assignment, and let $\tilde{d}_k(\omega)$ be the output of the procedure* RandomizedCountNeighbors *described in Algorithm 4.2 for such an assignment. Then*

$$\Pr\left(\left|\tilde{d}_k(\omega) - \frac{|B(\omega)|}{|UP|}\right| \geq \varepsilon\right) \leq 2\exp\left(-2k\varepsilon^2\right).$$

Proof We will use the Hoeffding inequality [44] to prove this theorem. It says that if $X_1 \ldots X_k$ are independent random variables such that $0 \leq X_i \leq 1$, then

$$\Pr\left(\left|\sum_{i=1}^{k} X_i - \mathbb{E}\left[\sum_{i=1}^{k} X_i\right]\right| \geq t\right) \leq 2\exp\left(-\frac{2t^2}{k}\right), \tag{4.14}$$

where $\mathbb{E}[\cdot]$ is the expected value of a random variable. In our case, X_i indicates whether ω induce a biclique with a randomly chosen assignment $\omega_i \in UP$, namely

$$X_i = \begin{cases} 1, & \omega \in B(\omega_i); \\ 0, & \text{otherwise.} \end{cases}$$

Hence, Equation 4.14 can be rewritten as

$$\Pr\left(\left|\frac{1}{k}\sum_{i=1}^{k} X_i - \mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} X_i\right]\right| \geq \varepsilon\right) \leq 2\exp\left(-2k\varepsilon^2\right), \tag{4.15}$$

where $\varepsilon = t/k$. Notice that the value $\frac{1}{k}\sum_{i=1}^{k} X_i$ is exactly the output of Algorithm 4.2. Hence, in order to prove that the algorithm gives an approximation of $|B(\omega)|/|UP|$, we have to prove that $\mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} X_i\right]$ is equal to $|B(\omega)|/|UP|$. Because of the linearity of the expectation, the following equation holds:

$$\mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} X_i\right] = \frac{1}{k}\sum_{i=1}^{k}\mathbb{E}[X_i]. \tag{4.16}$$

Since the assignment ω_i is picked uniformly at random, the probability to choose it is $1/|UP|$. Thus,

$$\forall i \in 1 \ldots k, \quad \mathbb{E}[X_i] = \sum_{j=1}^{|UP|} \frac{X_j}{|UP|},$$

completing the proof. ■

For practical applications of Algorithm 4.2, it is possible to calculate the number of samples needed to obtain an expected error less than ε with a probability greater than p. The following equation directly derives from Theorem 4.5:

$$k > -\frac{1}{2\varepsilon^2}\ln\left(\frac{1-p}{2}\right). \tag{4.17}$$

For example, if we want an error $\varepsilon < 0.05$ with probability greater than 98.6%, it is enough to choose $k \geq 993$.

4.1.5 Experimental Results

To prove the viability of our approach, we applied it to several real-world datasets at our disposal. In the following, we first report the application of our

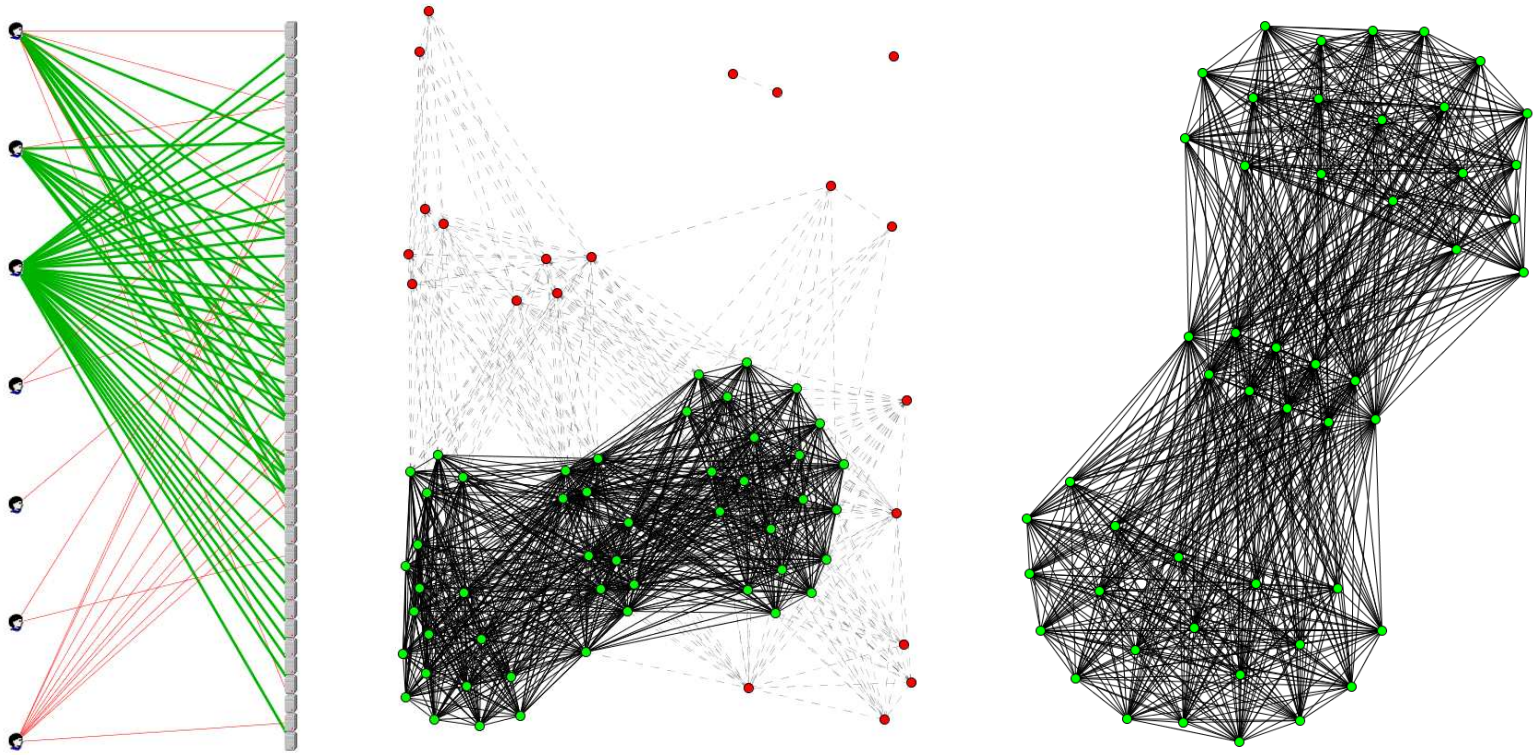
model to the access control configuration related to users of an organization unit of a large company. Then, by using the previous dataset, we highlight the effect of the pruning operation on the role mining complexity. Finally, we show how it is possible to compute the optimal threshold to use with our pruning strategy. In all the tests we used the approximated version of our pruning algorithm (with $k = 1000$), and normalized values for the pruning threshold, as detailed in Section 4.1.4.

A Real Case

Figure 4.3 shows an example of our strategy when applied to a real dataset. Figure 4.3(a) represents the bipartite graph G built from the access control configuration relative to users of an Organization Unit (OU) of a large company. The OU analyzed counts 7 users (nodes on the left) and 39 permissions (nodes on the right), with a total of 71 user-permission assignments. We have chosen an OU with few users and permissions to ease graph representation. According to a pruning threshold equal to 0.39, stable assignments are depicted with thicker edges, while unstable assignments with thinner edges. Figure 4.3(b) depicts the unipartite graph G', built according to Equation 4.4. The user-permission assignments of G correspond to the vertices of G', and two vertices are connected by an edge if they induce a biclique. Dashed edges indicate that one of the two endpoints will be pruned. Figure 4.3(c) shows only the stable assignments, namely the ones that will survive to the pruning operation. By comparing these last two figures it is possible to see that the main component of the whole graph survives after the pruning, while pruned assignments correspond to "noise". Indeed, the pruned vertices induce a biclique with only a small fraction of nodes of the main component.

Effects of the Pruning on the Mining Complexity

Theorem 4.4 states that the local clustering coefficient of a vertex is upper bounded by the ratio of the average neighborhoods degrees and its own degree. As a consequence, stable assignments have a limited clustering coefficient because of the low degree of their neighbors. This means that these assignments are difficult to manage in a role mining process. Yet, they also are the most "interesting" one since they are stable assignments. Our pruning operation is able to increase the average degrees of neighbors, and, at the same time, to decrease the degree of stable assignments. Thus, it is able to increase the above limitation of the local clustering coefficient. In the following, we will experimentally show that when the pruning is executed, not only does the above local clustering coefficient limit increase, but even the clustering



(a) Bipartite representation of the analyzed OU

(b) Corresponding unipartite graph G'

(c) Pruned unipartite graph

Figure 4.3 Our model applied to a real Organizational Unit

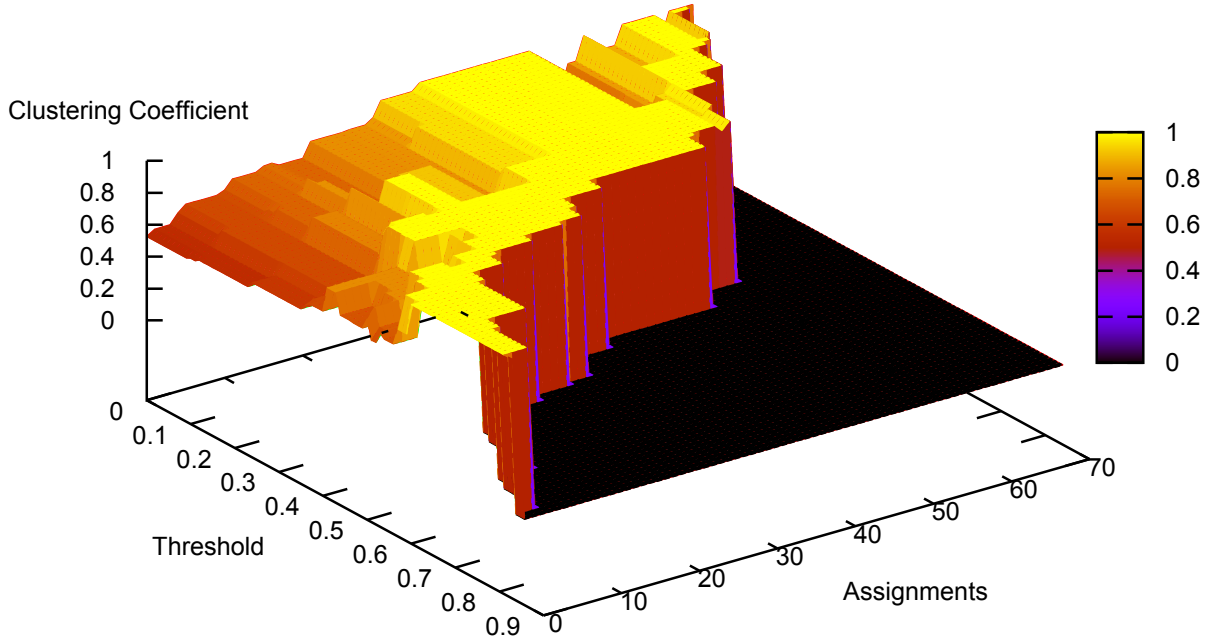


Figure 4.4 Pruning effect on local clustering coefficient

coefficient grows.

Figure 4.4 graphically shows this behavior. The dataset analyzed is the same that has been used in Section 4.1.5. The clustering coefficient has been reported for all the assignments, which are ordered by *descending* degree (i.e., descending stability), and for different pruning thresholds. For representation purposes, we have assigned 0 to the clustering coefficient of pruned assignments. By analyzing Figure 4.4, it turns out that originally stable assignments have a limited clustering coefficient. Indeed, all the assignments numbered between 0 and 20 have a clustering coefficient lower than 0.73 when no pruning operation is executed (threshold $= 0$). Further, it turns out that the clustering coefficient increases when a higher pruning threshold is used. For example, when the threshold is equal to 0.39, all the assignments numbered between 10 and 50 have a clustering coefficient equal to 1. Note that, according to Theorem 4.3 in these cases only one maximal biclique which they can belong to exists. In terms of RBAC, there exists only one role (represented by a maximal biclique) that they can belong to. Furthermore, the pruned assignments are only 20 out of 71, the assignments with a clustering coefficient equal to 1 are 40, while only 10 assignments have a clustering coefficient between 0 and 1. Anyway, the clustering coefficient of 5 out of these 10 assignments increased

from 0.52 to 0.65, while it was almost steady for the other 5 assignments. This means that the mining complexity has been actually reduced.

Threshold Tuning

The tuning of the threshold to use in our pruning algorithm depends on the final objective of the data analysis problem. In particular, we first need to define a metric that measures how well the objective has been reached. Then, it is possible to use this metric to choose the best threshold. The metric that we used in our tests is a *multi-objective* function that considers different aspects of the role engineering problem. Multi-objective analysis often means to trade-off conflicting goals. In a role engineering context, for example, we execute the pruning while requiring to minimize the complexity of the mining, minimize the number of pruned assignments, and maximize the stability of the candidate role-set.

A viable approach to solve a multi-objective optimization problem is to build a *single aggregated objective function* from the given objective functions [25]. One possible way to do this is combining different functions in a *weighted sum*, with the following general formulation:

$$\sum_{f_i \in F} \alpha_i f_i. \tag{4.18}$$

F is the set of the functions to optimize, and α_i is a scale parameter that can be different for each function $f_i \in F$. Put another way, one specifies scalar weights for each objective to be optimized, and then combines them into a single function that can be solved by any single-objective optimizer. Once we defined the aggregated objective function, the problem of finding the best trade-off corresponds to the minimization of this function. The weight parameters can be negative or positive, according to the need of minimizing or to maximizing the corresponding function. Clearly, the solution obtained will depend on the values (more precisely, the relative values) of the specified weights. Thus, it may be noticed that the weighted sum method is essentially subjective, in that an analyst needs to supply the weights.

As for the practical computation of the best threshold, we identified the following objective functions:

- Clustering Coefficient, that indicates the global clustering coefficient of the unipartite graph G' built from *UP*. It is a measure of the mining complexity.
- Pruned Assignments, that is the number of assignments that are pruned by our algorithm.

- Maximal Bicliques, namely the number of maximal bicliques identifiable in G. They represent the number of maximal roles of the underlying access control configuration.
- Average Weight, that is the average weight of the roles relative to the set of maximal bicliques. The weight of a role r is defined as $\left|U_r\right| \times \left|P_r\right|$.

These objectives have been combined in the following multi-objective function:

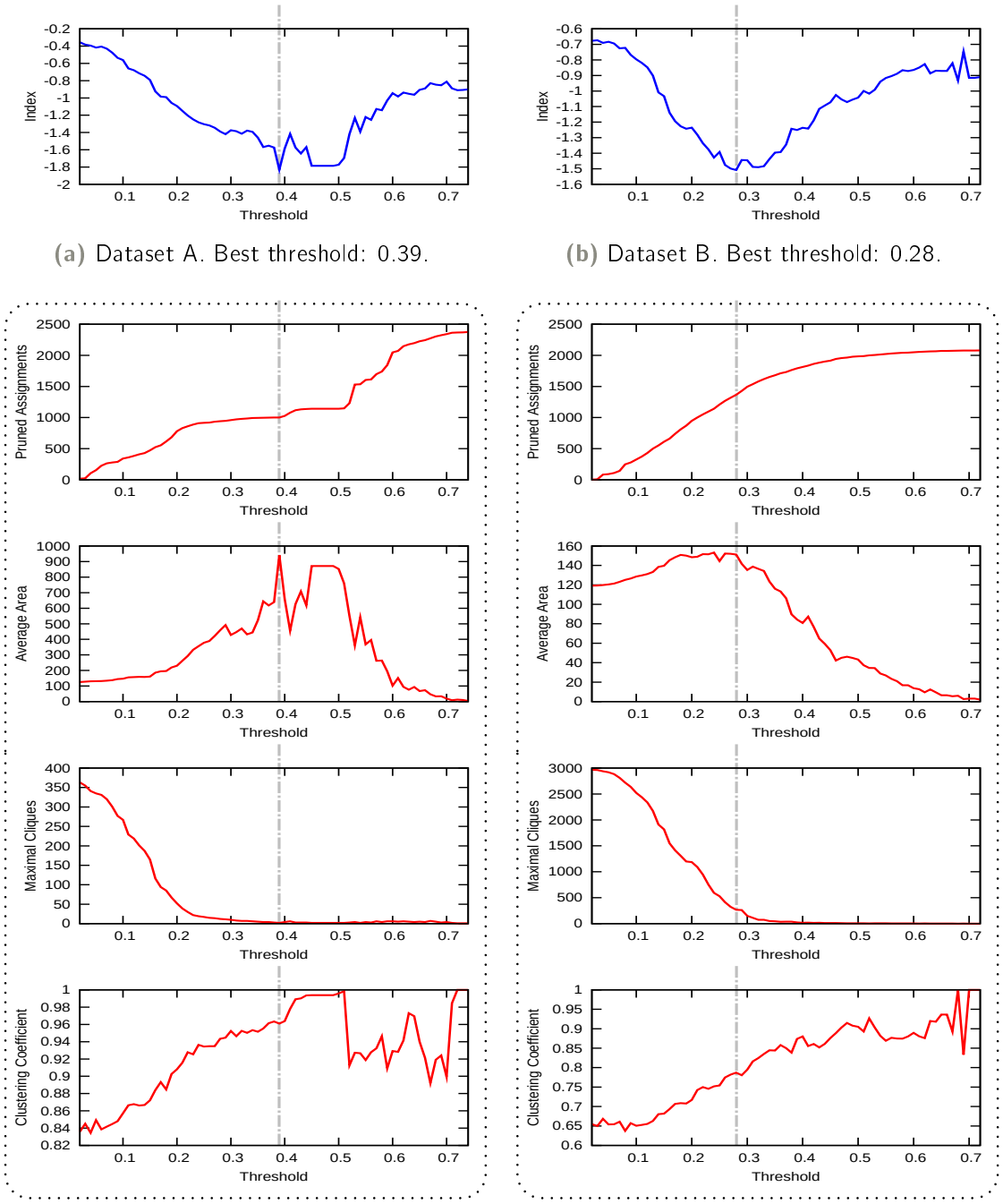
$$\text{Index} = -\text{Clustering Coefficient} + \frac{0.3 \times \text{Pruned Assignments}}{max(\text{Pruned Assignments})} + \frac{0.8 \times \text{Maximal Bicliques}}{max(\text{Maximal Bicliques})} - \frac{0.8 \times \text{Average Weight}}{max(\text{Average Weight})}$$

Finding the "best" threshold means to minimize the previous equation. Weights have been chosen by giving a higher relevance to the clustering coefficient; an intermediate relevance to the maximal bicliques number and to the average weight; and finally, a low relevance to the number of pruned assignments. Thus, we are willing to reduce the number of pruned assignments, by contextually reducing the complexity of the role mining task, the number of maximal bicliques, and maximizing the average weight.

In Figure 4.5, we report two examples of the threshold tuning applied to two real datasets at our disposal. The two analyzed cases concern two organization units of a large company. They are comparable with respect to their size: the first one counts 54 users and 285 permissions, with a total of 2,379 assignments; the second one is composed of 48 users, 299 permissions, and a total of 2,081 assignments. The difference between them mainly lies on the mining complexity: the first one has a global clustering coefficient higher than the second one (0.84 vs. 0.66). In particular:

- Dataset A: high clustering coefficient (0.84), 54 users, 285 permissions, and 2,379 assignments.
- Dataset B: low clustering coefficient (0.66), 48 users, 299 permissions, and 2,081 assignments.

By using the given cost function, a high relevance is given to Clustering Coefficient, a medium one is given to Average Weight and Maximal Bicliques, while less relevance is given to Pruned Assignments. In this way, we are willing to prune a high number of assignments to reduce the complexity of the role mining task, by contextually minimizing the number of maximal bicliques and maximizing the average weight. Figures 4.5(a) and 4.5(b) represent the aggregated functions for these two organization unit. Figures 4.5(c) and 4.5(d)



(a) Dataset A. Best threshold: 0.39.

(b) Dataset B. Best threshold: 0.28.

(c) Objective functions of Figure (a)

(d) Objective functions of Figure (b)

Figure 4.5 Finding the best threshold

show the four functions that compose the aggregated one. In both cases, the minimum of the aggregated function is highlighted with a vertical dashed line.

As for the first organization unit, the minimum is reached when the threshold is equal to 0.39. Indeed, in Figure 4.5(c) it can be seen that this is a good trade-off among all the four single functions: the pruned assignments are 1,001 out of 2,379; the average weight (that indicates the average stability) has grown almost 9 times from the original average weight; the number of maximal bicliques has been decreased from 350 to 2; finally the clustering coefficient has been increased from 0.84 to 0.96. Note that, since we have only 2 maximal bicliques, we are able to manage all the assignments survived to the pruning with only 2 roles. Put another way, we found two stable roles that together are able to manage 1,378 out of 2,379 assignments.

As for the second organization unit, the minimum of the multi-objective function is reached when the threshold is equal to 0.28 (see Figure 4.5(b)). In this case, the pruned assignments are 1,367, the average weight increased from 120 to 151, the number of maximal bicliques has been decreased from 3,000 to 266, while the clustering coefficient has been increased from 0.66 to 0.78. At first sight, it seems that we pruned too much assignments, but these results depend both on the dataset we are analyzing and on the targets that role engineers want to reach. Indeed, this dataset has a higher complexity with respect to the first one, and we provided high weights for Clustering Coefficient and Maximal Bicliques. If we gave less relevance to these two parameters, a lower threshold would have been a good trade off. In that case, the pruned assignments would have been less than 1,367, and the average weight would have been higher than the original one. In general, the role engineers mission is to establish the weights of the multi-objective aggregated function in such a way to get as close as possible to the target that they want to reach.

4.2 Business-Relevant RBAC States through Decomposition

It is generally accepted that role mining must count on business requirements to increase its effectiveness. Indeed, roles elicited without leveraging on business information are unlikely to be intelligible by system administrators. A business-oriented categorization of users and permissions (e.g., organizational units, job titles, cost centers, business processes, etc.) could help administrators to identify the job profiles of users and, as a consequence, which roles should be assigned to them. Nonetheless, most of the existing role mining

techniques yield roles that have no clear relationship with the business structure of the organization where the role mining is being applied. To face this problem, we propose a methodology that allows role engineers to leverage business information during the role finding process. The key idea is decomposing the dataset to analyze into several partitions, in a way that each partition is homogeneous from a business perspective. Each partition groups users or permissions with the same business categorization (e.g., all the users belonging to the same department, or all the permissions that support the execution of the same business process). Such partitions can then be role-mined independently, hence achieving four main results: (1) elicited roles have a clearer relationship with business information; (2) role mining complexity is reduced; (3) mining algorithms do not seek to find commonalities among users with fundamentally different job profiles or among uncorrelated permissions; (4) any role mining algorithm can be used within this framework. When several business attributes are available, analysts need to figure out which one produces the decomposition that leads to the most intelligible roles.

4.2.1 The Business-Driven Decomposition Approach

The key observation is that users sharing the same business attributes will essentially perform the same task within the organization. For instance, if invoice clerks are supposed to perform similar activities (e.g., they all gather data from vendor invoices to ensure that billing information is accurate), it would be better off analyzing these users and their permissions separately from users and permissions of the rest of the organization. This will avoid the algorithm churning away to find commonalities among users with fundamentally different job profiles (e.g., we do not expect common permissions between data entry clerks and invoice clerks). Rather, the role-mining algorithm will naturally discover roles that are inherently relevant to invoice clerks' job function, while being more intuitive for humans. In general, restricting role mining techniques to users that have common business attributes (e.g., same department, job title, country, etc.) will ensure that elicited roles are only related to such business characteristics. Consequently, it will be easier for a role engineer to assign a business meaning to roles suggested by bottom-up approaches. Moreover, partitioning data also introduces benefits in terms of execution time of role mining algorithms. Indeed, most role mining algorithms have a complexity that is not linear compared to the number of users or permissions to analyze [6, 29, 78].

To address the above-mentioned issues, we will introduce a methodology that helps role engineers to leverage business information during the role min-

ing process. Instead of performing a single bottom-up analysis on the entire organization as a whole, we propose to divide the access data into smaller subsets that are homogeneous according to a business perspective. Each data partition is made up of user and/or permission that share the same value for an attribute that is relevant for the business. We advocate that using role mining without applying some human insight usually leads to roles that are not intuitive for humans. This is because role-mining algorithms usually lead to roles that are optimal in a mathematical way (e.g., minimizing their number), but appear to humans like arbitrary collections of permissions, without any business meaning. One can avoid this pitfall by orienting role-mining to the high-level classification of the organization—usually translated into user and permission attributes.

To apply the proposed divide-and-conquer strategy, several enterprise information can be used. Business processes, workflow tasks, and organization units are just some examples of business attributes that can be leveraged to identify data partitions. When dealing with information from several sources, the main problem is thus ascertaining which information induces the partitioning that hits the following objectives most: (1) simplify the overall role engineering task; and, (2) simplify the role management task, by helping determine which permissions are required by the user and which roles should be assigned. To select the most suitable business attributes, we introduce and evaluate three different indices referred to as: ENTRUSTABILITY, MINABILITY GAIN, and SIMILARITY GAIN. For all of them, we will show that the decomposition with the highest index value is the one that best fits access control data, and thus it is the one that most likely leads to roles with a clear purpose for business people. We compare the behavior of the three different indices, pointing out appealing features and drawbacks of each of them. Successively, we show the application of the proposed methodology over real enterprise data. Results support the quality and viability of the proposal.

Pseudo-Roles and Entrustability

In the following, we introduce the *pseudo-role* concept as a means to identify sets of users and permissions that can be managed by the same role. In turn, we formally introduce the ENTRUSTABILITY index to measure how much a partition reduces the uncertainty in locating such sets of users and permissions in each subset of the partition.

Pseudo-Roles The Pseudo-Role concept is related to the pseudo-bicluster introduced in Section 2 (Definition 2.5). Indeed the pseudo-role generated

by $\langle u, p\rangle \in UP$ is a pseudo-bicluster such that $u \in \hat{U}$ and $p \in \hat{P}$, that is a pseudo-bicluster that has $\langle u, p\rangle$ among its generators. Formally:

Definition 4.5 (Pseudo-Role) Given a user-permission assignment $\langle u, p\rangle \in UP$, the *pseudo-role generated by* $\langle u, p\rangle$ is a role made up of users $users(p)$ and permissions $perms(u)$.

The pseudo-role tool will be employed to identify those user-permission assignments that can be managed together with a given assignment through a single role. Since a pseudo-role $\hat{r}$ is *not* an actual role, with abuse of notation we refer to its users as $ass\_users(\hat{r})$ and to its permissions as $ass\_perms(\hat{r})$. Several user-permission assignments can generate the same pseudo-role. In particular:

Definition 4.6 (Frequency of a Pseudo-Role) The percentage of user-permission assignments of UP that generates a pseudo-roles $\hat{r}$ is referred to as its *frequency*, defined as:

$$f(\hat{r}) := \frac{1}{|UP|}\Big|\{\langle u, p\rangle \in UP \mid ass\_users(\hat{r}) = users(p) \wedge ass\_perms(\hat{r}) = perms(u)\}\Big|$$

The introduction of the pseudo-roles concept is supported by the following theorem:

Theorem 4.6 *Given a user-permission assignment $\langle u, p\rangle \in UP$, let $\hat{r}$ be the pseudo-role generated by $\langle u, p\rangle$. Then*

$$UP_{\hat{r}} := \big(ass\_users(\hat{r}) \times ass\_perms(\hat{r})\big) \cap UP$$

is the set of all possible user-assignment relationships that can be covered by any role to which $\langle u, p\rangle$ belongs to. Hence, for each possible RBAC state $\langle ROLES, UA, PA\rangle$ that covers the assignments in UP the following holds:

$$\forall r \in ROLES \,:\, u \in ass\_users(r),\ p \in ass\_perms(r) \implies ass\_users(r) \times ass\_perms(r) \subseteq UP_{\hat{r}}\ .$$

PROOF First, we prove that any assignment that can be managed together with $\langle u, p\rangle$ must be within $UP_{\hat{r}}$. Let $\langle u', p'\rangle \in UP$ be an assignment outside the pseudo-role $\hat{r}$, namely $\langle u', p'\rangle \notin UP_{\hat{r}}$. If, by contradiction, $\langle u, p\rangle$ and $\langle u', p'\rangle$ can be managed through the same role r', then by definition all the users $ass\_users(r')$ must have permissions $ass\_perms(r')$ granted. Hence, both the

assignments $\langle u', p\rangle$ and $\langle u, p'\rangle$ must exist in UP. But, according to Definition 4.5, $u' \in ass\_users(\hat{r}) = users(p)$ and $p' \in ass\_perms(\hat{r}) = perms(u)$, that is a contradiction.

Now we prove that any assignment within $UP_{\hat{r}}$ can be managed together with $\langle u, p\rangle$ via a single role. Given $\langle u'', p''\rangle \in UP_{\hat{r}}$, Definition 4.5 yields $u'' \in ass\_users(\hat{r}) = users(p)$ and $p'' \in ass\_perms(\hat{r}) = perms(u)$. Thus, both the assignments $\langle u'', p\rangle$ and $\langle u, p''\rangle$ exist in UP, completing the proof.

According to the previous theorem, a pseudo-role groups all user-permission assignments that are manageable through any of the roles that also covers the pseudo-role generators. The pseudo-role frequency indicates the minimum number of assignments covered by the pseudo-role (i.e., the generators) that are manageable through the same role. Consequently, the higher the frequency of a pseudo-role is, the more pseudo-role assignments can be managed by one role. Similarly, the lower the frequency is, the more likely it is that the assignments covered by a pseudo-role cannot be managed by a single role. Therefore, the ideal situation is when pseudo-role frequencies are either close to 1 or close to 0: frequent pseudo-roles circumscribe a portion of assignments that are worth investigating since they likely contain a role for managing most of the assignments; conversely, unfrequent pseudo-roles identify assignment sets that are not worth analyzing.

Entrustability Based on the previous observations, we are interested in finding the decomposition that produces pseudo-roles with frequencies either close to 1 or to 0. In the following we show that the *entropy* concept is a natural way to capture these circumstances. Let $\mathcal{A}$ be the set of all values assumed by a given business information—for instance, $\mathcal{A}$ can represent the "job title" information, and one of the actual values $a \in \mathcal{A}$ can be "accountant". Let $\mathcal{P} := \{UP_{a_1}, \ldots, UP_{a_n}\}$ be a n-partition of UP induced by the business information $\mathcal{A}$ such that the number of subsets are $n = |\mathcal{A}|$, each subset is such that $UP_{a_i} \subseteq UP$, the subset indices are $\forall i \in 1, \ldots, n : a_i \in \mathcal{A}$, and the subset are such that $UP = \bigcup_{a\in\mathcal{A}} UP_a$. UP_a indicates all assignments that "satisfy" the attribute value a (e.g., if $\mathcal{A}$ represents the "job title" information, all the assignments where users are "accountant" are one subset). Notice that, according to the previous partition definition, subsets can overlap, namely $\left|UP_a \cap UP_{a'}\right| \geq 0$ when users or permissions can be associated to more than one attribute value. Let $\mathcal{R}_a$ be the set of all pseudo-roles that can be generated within the subset UP_a, and $\mathcal{R} := \bigcup_{a\in\mathcal{A}} \mathcal{R}_a \cup \mathcal{R}_*$ where $\mathcal{R}_*$ represents the pseudo-roles belonging to UP before decomposing it. Notice that the same pseudo-role might belong to both $\mathcal{R}_*$ and another set $\mathcal{R}_a$, namely $\left|\mathcal{R}_* \cap \mathcal{R}_a\right| \geq 0$, but not necessarily with the same frequencies.

Let $A \in \mathcal{A}$ be the random variable that corresponds to a value of the given business attribute, while the random variable $R \in \mathcal{R}$ denotes a pseudo-role generated by a generic user-permission assignment. Let $\Pr(\hat{r})$ be the empirical probability of a pseudo-role $\hat{r} \in \mathcal{R}$ being generated by an unspecified user-permission assignment. More specifically,

$$\Pr(\hat{r}) := \frac{1}{|UP|} \sum_{\omega \in UP} g(\omega, \hat{r})$$

where

$$g(\omega, \hat{r}) := \begin{cases} 1, & \omega \text{ generates } \hat{r} \text{ in } UP; \\ 0, & \text{otherwise.} \end{cases}$$

Similarly, the empirical probability of a pseudo-role being generated by an unspecified user-permission assignment that "satisfies" the business attribute a is

$$\Pr(\hat{r} \mid A = a) := \frac{1}{\left|UP_a\right|} \sum_{\omega \in UP_a} g_a(\omega, \hat{r})$$

where

$$g_a(\omega, \hat{r}) := \begin{cases} 1, & \omega \text{ generates } \hat{r} \text{ in } UP_a; \\ 0, & \text{otherwise.} \end{cases}$$

Notice that, for each attribute value a, when $\hat{r} \in \mathcal{R}_a$, then $\Pr(\hat{r})$ corresponds to the frequency definition. Conversely, if $\hat{r} \in \mathcal{R} \setminus \mathcal{R}_a$, then $\Pr(\hat{r}) = 0$.

As stated before, the natural measure for the information of the random variable R is its entropy $H(R)$. The binary entropy, defined as

$$H(R) := -\sum_{\hat{r} \in \mathcal{R}} \Pr(\hat{r}) \log_2 \Pr(\hat{r})$$

quantifies the missing information on whether the pseudo-role $\hat{r}$ is generated from some unspecified user-permission assignment when considering the set UP as a whole. By convention, $0 \times \log_2 0 = 0$. The conditional entropy is defined as

$$H(R \mid A) := -\sum_{a \in \mathcal{A}} \Pr(a) \sum_{x\hat{r} \in \mathcal{R}} \Pr(\hat{r} \mid A = a) \log_2 \Pr(\hat{r} \mid A = a) \ ,$$

where $\Pr(a) := \left|UP_a\right| / \sum_{a \in \mathcal{A}} \left|UP_a\right|$ measures the empirical probability of choosing an assignment that satisfies a. $H(R \mid A)$ quantifies the missing information on whether the pseudo-role $\hat{r}$ is generated from some unspecified user-permission assignment when A is known. The mutual information

$$I(R;A) := H(R) - H(R \mid A)$$

measures how much the knowledge of A changes the information on R. Hence, $I(R;A)$ measures how much the knowledge of the business information A helps us to predict the set of users and permissions that are manageable by the same role within each subset. Since $I(R;A)$ is an absolute measure of the entropy variation, we introduce the following measure for the fraction of missing information removed by the knowledge of A with respect to the entropy $H(R)$ before partition:

$$\text{ENTRUSTABILITY}(A) := \frac{I(R;A)}{H(R)} = 1 - \frac{H(R \mid A)}{H(R)} \ .$$

By selecting the decomposition with the highest ENTRUSTABILITY value, we choose the decomposition that simplifies the subsequent role mining analysis most. Notice that the previous equations consider one business attribute at a time. Given ℓ business information $\mathcal{A}_1, \ldots, \mathcal{A}_\ell$, it is simple to extend the definition of the ENTRUSTABILITY index by partitioning *UP* in subsets of assignments that contextually satisfies all business information which has been provided.

Similarity Gain

In the following we will introduce the definitions of similarity between two users and conditioned similarity, and then we will introduce a new concept: the *similarity gain*. It evaluates the gain achieved in similarity between users when a given business information is used to induce the decomposition of the dataset.

Definition 4.7 (Similarity Between Two Users) Given a pair of users $u_1, u_2 \in$ *USERS*, the *similarity* between them is defined as:

$$s(u_1, u_2) := \frac{\left|perms(u_1) \cap perms(u_2)\right|}{\left|perms(u_1) \cup perms(u_2)\right|} \tag{4.19}$$

Definition 4.8 (Similarity Among a Set of Users) Given a set of users *USERS*, the *similarity* index of that set is the average similarity between all possible (unordered) user pairs, that is:

$$\mathcal{S}(USERS) := \begin{cases} \dfrac{1}{\binom{|USERS|}{2}} \displaystyle\sum_{\substack{u_1,u_2 \in USERS: \\ u_1 \neq u_2}} s(u_1, u_2), & |USERS| > 1; \\ 1, & \text{otherwise.} \end{cases} \tag{4.20}$$

In other words, the similarity of a set of users corresponds to the *Jaccard Coefficient* between all possible (unordered) user pairs. The *Jacccard Coefficient* is

a tool largely used in the data mining domain to group together items that are similar. Therefore, the similarity index seems to be a good candidate metric to find subsets of users that are similar from a business perspective.

Let $\mathcal{P}$ be the partition of UP induced by a given business information $\mathcal{A} = \{a_1 \dots a_n\}$. In other words, let $\mathcal{P} := \{UP_{a_1}, \dots, UP_{a_n}\}$ be an n-partition of UP such that $UP_{a_i} \subseteq UP$ and $UP = \bigcup_{i=1}^{n} UP_{a_i}$. Each subset UP_{a_i} identifies a set of users Υ_i, such that $\Upsilon_i := \{u \in USERS \mid \exists p \in PERMS, \langle u, p\rangle \in UP_{a_i}\}$. According to (4.20), the similarity of Υ_i can be defined as:

$$\mathcal{S}(\Upsilon_i) := \begin{cases} \dfrac{1}{\binom{|\Upsilon_i|}{2}} \displaystyle\sum_{\substack{u_1,u_2 \in \Upsilon_i: \\ u_1 \neq u_2}} s_{\Omega_i}(u_1, u_2), & |U| > 1; \\ 1, & \text{otherwise.} \end{cases} \tag{4.21}$$

where $s_{\Omega_i}(u_1, u_2)$ is the similarity of the users u_1 and u_2 obtained by only considering the permissions that are involved in UP_{a_i}. Equation 4.21 can also be rewritten in the following way:

$$\mathcal{S}(\Upsilon_i) := \frac{1}{\sigma_i + \binom{|\Upsilon_i|}{2}} \left(\sigma_i + \sum_{\substack{u_1,u_2 \in \Upsilon_i: \\ u_1 \neq u_2}} s_{\Omega_i}(u_1, u_2) \right),$$

where

$$\sigma_i := \begin{cases} 1, & |\Upsilon_i| = 1; \\ 0, & \text{otherwise.} \end{cases}$$

Using the previous definitions, the conditioned similarity is defined as:

Definition 4.9 (Conditioned Similarity) Given a n-partition $\mathcal{P} := \{UP_{a_1}, \dots, UP_{a_n}\}$ for UP such that $UP = \bigcup_{i=1}^{n} UP_{a_i}$ and the induced sets of users $\Upsilon_i := \{u \in USERS \mid \exists p \in PERMS, \langle u, p\rangle \in UP_{a_i}\}$, we define the similarity index *conditioned by* $\mathcal{P}$ as

$$\mathcal{S}_\Omega(USERS) = \frac{\sum_{i=1}^{n} \mathcal{S}(\Upsilon_i)\left(\sigma_i + \binom{|\Upsilon_i|}{2}\right)}{\sum_{i=1}^{n}\left(\sigma_i + \binom{|\Upsilon_i|}{2}\right)} = \frac{\sum_{i=1}^{n} \sigma_i + \sum_{i=1}^{n} \mathcal{S}(\Upsilon_i)\binom{|\Upsilon_i|}{2}}{\sum_{i=1}^{n} \sigma_i + \sum_{i=1}^{n} \binom{|\Upsilon_i|}{2}}. \tag{4.22}$$

Definition 4.10 (Similarity Gain) Given a partition $\mathcal{P}$ for UP defined as above, the *similarity gain* of $\mathcal{P}$ is formally defined as:

$$\text{SIMILARITY\_GAIN}_{\mathcal{P}}(USERS) := 1 - \frac{\mathcal{S}(USERS)}{\mathcal{S}_\Omega(USERS)} \tag{4.23}$$

where $USERS$ is the set of all the users involved in UP.

Therefore, the similarity gain measures the percentage increase (or decrease) of the similarity after partitioning UP in smaller subsets. This index reaches the maximum on 1, but it can assume negative values as well. A negative value means that the partition $\mathcal{P}$ is not advantageous from the similarity perspective, and that is probably better to analyze the whole set UP instead of decomposing it. When we have at our disposal several user attributes, we can evaluate the similarity gain for each induced partition, and then select the one with the highest value. Note that we have defined the similarity index, and the corresponding similarity gain, between users, but it is also possible to define the similarity between permissions in a very analogous fashion. As a matter of fact, in the experimental section we will consider both these indices. To ease exposition, we omit the corresponding definitions.

Minability Gain

The *minability* index has been previously introduced in Section 4.1. We will now introduce the conditioned minability. Given an attribute $\mathcal{A} = \{a_1 \dots a_n\}$, let $\mathcal{P} := \{UP_{a_1}, \dots, UP_{a_n}\}$ be the partition of UP induced by $\mathcal{A}$. According to Definition 4.4 the minability index of each subset UP_{a_i} is

$$\mathcal{M}(UP_{a_i}) := \frac{1}{\left|UP_{a_i}\right|} \sum_{\omega \in UP_{a_i}} m_{\Omega_i}(\omega),$$

where $m_{\Omega_i}(\omega)$ indicates the local minability of ω obtained considering only the user-permission assignments belonging to UP_{a_i}. This leads to the following definition:

Definition 4.11 (Conditioned Minability) Given a n-partition $\mathcal{P} := \{UP_{a_1}, \dots, UP_{a_n}\}$ of UP, the minability index *conditioned by* $\mathcal{P}$ is

$$\mathcal{M}_\Omega(UP) = \frac{\sum_{i=1}^{n} \mathcal{M}(UP_{a_i}) \left|UP_{a_i}\right|}{\sum_{i=1}^{n} \left|UP_{a_i}\right|} = \frac{1}{|UP|} \sum_{i=1}^{n} \sum_{\omega \in UP_{a_i}} m_{\Omega_i}(\omega) \tag{4.24}$$

$$= \frac{1}{|UP|} \sum_{\omega \in UP} m_{\Omega_i}(\omega). \tag{4.25}$$

We will now introduce the MINABILITY GAIN, that is the percentage minability increase or decrease due to a given partition.

Definition 4.12 Given a n-partition $\mathcal{P} := \{UP_{a_1}, \dots, UP_{a_n}\}$ of UP, the *minability gain* of $\mathcal{P}$ is formally defined as:

$$\text{MINABILITY\_GAIN}_{\mathcal{P}}(UP) := 1 - \frac{\mathcal{M}(UP)}{\mathcal{M}_\Omega(UP)} \tag{4.26}$$

In the following, we will use the MINABILITY GAIN to evaluate the quality of a partition induced by a business information. Having at our disposal several business information, the one with the highest minability gain will be the most suitable for decomposing the dataset in smaller subsets.

4.2.2 Indices Comparison and Discussion

In this section we discuss the distinguishing features of the proposed indices through some practical examples. In particular, we first introduce a few trivial datasets to better analyze the behavior of each index. Later, we discuss complex real data to support the quality of the result of our methodology. In details, our decomposition approach can leverage on any of ENTRUSTABILITY, SIMILARITY GAIN, and MINABILITY GAIN. They grasp different aspects, and in this section we are going to discuss and to compare them against both synthetic and real data. Since such indices respectively rely on the entropy, similarity, and minability concepts, we will first analyze them. To this aim, we introduce a normalized version of the entropy; indeed, minability and similarity range from 0 to 1, but the entropy index does not lie in the same range. The worst case for the entropy index is reached when each assignment belonging to *UP* generates a different pseudo-role, and in this case the entropy is equal to $\log_2 |UP|$. Therefore, we define the normalized entropy as:

Definition 4.13 (Normalized Entropy) Given a random variable $R \in \mathcal{R}$ that denotes a pseudo-role generated by a generic user-permission assignment belonging to *UP*, the normalized entropy of R is defined as:

$$\overline{H(R)} := 1 - \frac{H(R)}{\log_2 |UP|} \tag{4.27}$$

A normalized entropy equal to 1 corresponds to cases where the uncertainty on selecting the roles to use is minimal, while a normalized entropy equal to 0 corresponds to cases where the uncertainty is maximal.

Comparison Among Normalized Entropy, Minability and Similarity Normalized entropy, minability, and similarity catch different aspects of the analyzed data. The normalized entropy evaluates the uncertainty in deciding the role to be used to manage the assignments. The similarity index corresponds to the Jaccard coefficient between the users, and it measures how much similar are users in terms of their assigned permissions. Instead, the minability index measures the complexity of selecting candidate roles given a set of user-permission assignments. Entropy and minability look very similar, but there are a few substantial differences that we are going to point out.

Table 4.1 Comparison of normalized entropy, minability, and similarity on special cases

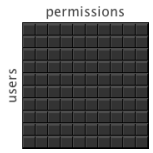
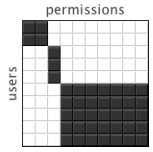
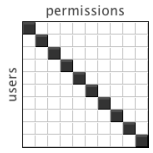
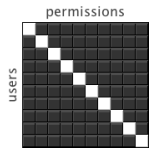
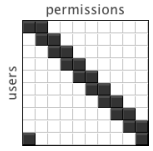
Configurations			$\overline{H(R)}$	$\mathcal{S}(USERS)$	$\mathcal{M}(UP)$
Best Role Mining Cases	CASE 1: One role		**1**	**1**	**1**
	CASE 2: Non-overlapping roles		between 0 and 1 (it depends on the number and size of the roles—in this example, 0.85)	between 0 and 1 (it depends on the number of users in each role—in this example, 0.31)	**1**
	CASE 3: Distinct users		**0**	**0**	**1**
Worst Role Mining Cases	CASE 4		**0**	close to 1 (in this example, 0.80)	close to 1 (in this example, 0.81)
	CASE 5		**0**	close to 0 (in this example, 0.07)	**0**

Table 4.1 provides a classification of the peculiarities of the three indices. In particular, the matrix representation of five access control configuration is reported together with the values of $\overline{H(R)}$, $\mathcal{S}(USERS)$ and $\mathcal{M}(UP)$. In the given representation, users are the rows of the matrix, permissions are the columns, and a black cell indicates a permission granted to the related user—see Chapter 6 for further details about the visual representation of user-permission matrices. The five different access control matrices presented in Table 4.1 are classified in two groups: the best role mining cases, and the worst ones. The first one contains configurations that are easy to analyze, while the second one contain configurations that can be considered the most difficult to analyze when trying to elicit roles.

It can be seen that when all the user-permission assignments belonging to

UP can be managed with just one role, all the indices equal 1 (CASE 1). It means that entropy, minability, and similarity are all good indices to recognize these kind of configurations. The second configuration (CASE 2) takes into account a set of user-permission assignments that can be managed with several roles that do not overlap. In this case, the minability index equals 1. As a matter of fact, each black rectangle identifies a role, and therefore it is easy to elicit a candidate role sets. Instead, the normalized entropy is less than 1: It depends on the fact that the assignments have to be managed with more than one role—so there is a certain uncertainty in selecting them. As for the similarity between users, its value is less than 1. Indeed, it is the average of all the similarities between pairs of users. Similarly to entropy, the similarity index does not identify this special case as one of the most convenient configuration for role mining. CASE 3 is a special configuration of CASE 2, where each user has his own permissions granted. In this case, the similarity is the minimum one (i.e., zero) and the uncertainty in selecting a role to assign to a random user is maximum (i.e., the normalized entropy is zero). The fourth case is one of the worst configurations from a role mining perspective (CASE 4). Here, it is not clear which is the best role-set that has to be used, since a large number of different roles can be elicited. The normalized entropy is 0: Each assignment will generate a different pseudo-role, and the uncertainty on deciding the roles to use is maximal. However, the minability is greater than 0: even if it is difficult to decide the roles, it is still possible to elicit "big" clusters of user-permission assignments, that is roles that cover several user-permission assignments. The similarity of users is close to 1; it depends on the fact that each pair of users shares $n-2$ permissions, where n is the number of permissions. The matrix represented in the fifth case is the worst case from a role mining perspective (CASE 5). Indeed, no role can manage more than two assignments, and each assignment can be covered by exactly two roles. Both the entropy and the minability correctly recognize this case as the worst one. Instead, similarity is greater than 0.

To conclude, among the three indices we proposed, the similarity index is the most susceptible to particular cases, and probably the less recommended to evaluate configuration that are "good" from the role mining perspective. It is only useful when finding a role that covers *all* the users. Minability and normalized entropy are instead good candidate for this purpose. However, they capture different aspects of the dataset, and therefore they have different behaviors. Normalized entropy tends to 1 when the majority of the assignments can be managed by only one "big" role, whereas not all users and permissions are necessarily covered with such a role. Minability tends to 1 when the set of roles required to cover all the assignments (and thus all users and permissions) can be easily identified. A good practice is to take into consideration all

these indices when trying to evaluate the quality of a given dataset for the role mining, and this is what we will do in Section 4.2.3 analyzing real enterprise data.

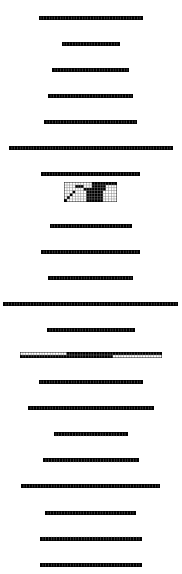
4.2.3 Entrustability, Similarity gain, and Minability gain on Real Data

To complete our comparison among proposed indices, in Figure 4.6 we report on the representation of the user-permission relationships involved in a real case. Data relates to an organization branch of an organization that counts 29 users, 240 permissions and a total of 867 user-permission assignments. We adopted several business information at our disposal to partition the whole dataset in smaller subsets. For each one we show its matrix representation, and the values of normalized entropy, similarity, and minability. The "Job Title" attribute induces a partition of 22 tiny subsets: 20 of them have normalized entropy, minability and similarity value equal to 1. Therefore, it is easy to elicit roles with a clear business meaning to manage all the assignments covered by these 20 subsets. Indeed, it is possible to create the roles "Military Occupation", "Fish Cleaner" and "Film processing technician" that are represented in Figure 4.6. Note that naming roles with the name of the attributes they are referring to, makes it easy for the security administrator to recognize their purpose. Further, note that the subset with the lowest indices values is "Without a Job Title". This subsets group-in all the users that do not have a job title. Therefore, it is expected that their respective tasks will be quite different.

The attributes "Cost Center" and "Branch" induce a partition made up of less subsets when compared to "Job Title". In particular, the "Cost Center" attribute induces 8 subsets out of 14 with normalized entropy, similarity, and minability equal to 1. On the contrary, the "Branch" attribute induces only 3 subsets out of 8 with maximal values for all the three indices. However, only looking to the matrix representations, it is difficult to evaluate which one of these two attributes is the most suitable to be used when decomposing the original dataset. To this end, we have to evaluate the quality of the partition induced by these attributes, and we can do it by looking at their ENTRUSTABILITY, MINABILITY GAIN and SIMILARITY GAIN indices. Note that in many real cases, the dataset is so large that it would be impossible to visualize neither the corresponding matrix representation, nor the representations of the partition induced by the attributes. Therefore the three indices that we introduced in this section are very helpful to make decisions when dealing with very large datasets.

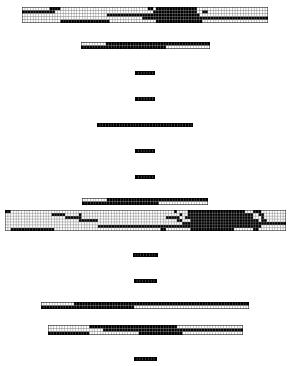
Figure 4.7 reports on the values of ENTRUSTABILITY, MINABILITY GAIN, and

Military occupation	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
Physical geographer	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
Life skill counselor	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
Apartment rental agent	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
Layout artist	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
Fish cleaner	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
Decommissioning and decontamination worker	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
Without a Job Title	$\overline{H(R)}$ = 0.26	$\mathcal{S}(USERS)$ = 0.54	$\mathcal{M}(UP)$ = 0.87
Technical sales support worker	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
University instructor	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
Foreign language interpreter	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
Film processing technician	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
Manufactured Job Title and mobile home installer	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
Jailer	$\overline{H(R)}$ = 0.67	$\mathcal{S}(USERS)$ = 0.33	$\mathcal{M}(UP)$ = 0.88
Industrial engineering technician	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
Network systems and data communications analyst	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
Billing and posting machine operator	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
Process technician	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
Marriage and family therapist	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
Clinical nurse specialist	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
Industrial millwright	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
Textile bleaching and dyeing machine operator	$H(R)$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00



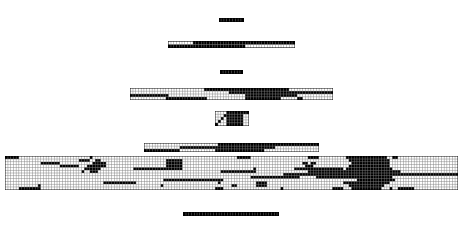
(a) Job Title

1000800	$\overline{H(R)}$ = 0.45	$\mathcal{S}(USERS)$ = 0.35	$\mathcal{M}(UP)$ = 0.84
1000805	$\overline{H(R)}$ = 0.69	$\mathcal{S}(USERS)$ = 0.47	$\mathcal{M}(UP)$ = 0.89
1002534	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
1002519	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
1002030	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
1002267	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
1002031	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
1004042	$\overline{H(R)}$ = 0.68	$\mathcal{S}(USERS)$ = 0.41	$\mathcal{M}(UP)$ = 0.88
1000776	$\overline{H(R)}$ = 0.35	$\mathcal{S}(USERS)$ = 0.46	$\mathcal{M}(UP)$ = 0.83
1002928	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
1003186	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
1000777	$\overline{H(R)}$ = 0.72	$\mathcal{S}(USERS)$ = 0.29	$\mathcal{M}(UP)$ = 0.91
1000799	$\overline{H(R)}$ = 0.55	$\mathcal{S}(USERS)$ = 0.34	$\mathcal{M}(UP)$ = 0.85
1001384	$H(R)$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00



(b) Cost Center

Solomon	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
Netcong	$\overline{H(R)}$ = 0.68	$\mathcal{S}(USERS)$ = 0.41	$\mathcal{M}(UP)$ = 0.88
Marysville	$\overline{H(R)}$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00
Winchester	$\overline{H(R)}$ = 0.44	$\mathcal{S}(USERS)$ = 0.34	$\mathcal{M}(UP)$ = 0.82
Mccool Junction	$\overline{H(R)}$ = 0.43	$\mathcal{S}(USERS)$ = 0.72	$\mathcal{M}(UP)$ = 0.95
Laurel	$H(R)$ = 0.56	$\mathcal{S}(USERS)$ = 0.37	$\mathcal{M}(UP)$ = 0.85
Long Island City	$\overline{H(R)}$ = 0.21	$\mathcal{S}(USERS)$ = 0.30	$\mathcal{M}(UP)$ = 0.75
San Rafael	$H(R)$ = 1.00	$\mathcal{S}(USERS)$ = 1.00	$\mathcal{M}(UP)$ = 1.00



(c) Branch

Figure 4.6 Graphical representation of user-permission relationships involved in a real dataset that counts 29 users, 240 permissions, and a total of 867 user-permission assignments. Three attributes are used to decompose the dataset: "Job Title", "Cost Center", and "Branch". For each attribute, the name of the subset, normalized entropy, similarity, and minability values are shown, together with its graphical representation.

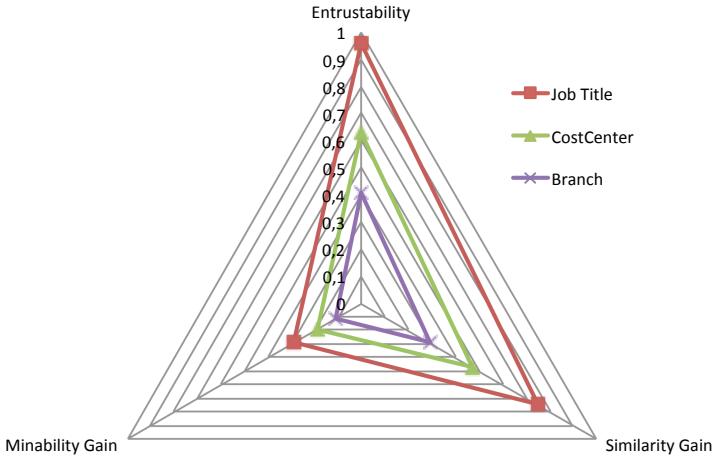


Figure 4.7 Entrustability, Minability Gain and Similarity Gain when partitioning the dataset using "Job Title", "Branch", and "Cost Center" attributes

SIMILARITY GAIN for the real case described above. The figure points out that the best partition is the one induced by the "Job Title" attribute, then there is the one induced by "Cost Center" and finally "Branch". This information turns out to be clear and described in a very compact way: we do not need to evaluate the single subsets induced by the three attributes.

In this section, we described a methodology that helps role engineers leverage business information during the role mining process. To drive this process, the ENTRUSTABILITY, MINABILITY GAIN, and SIMILARITY GAIN indices have been introduced. All of them measure the quality of the partition induced by a business attribute, each one capturing different aspects of the dataset. Leveraging on these indices, role engineers are able to identify the decomposition that increases business meaning in elicited roles in subsequent role mining steps, thus simplifying the analysis. An example on real data illustrates both the efficiency and the effectiveness of the proposed methodology, as well as the practical implications of the proposed indices.

4.3 Overcoming the Limitations of Divide et Impera Approaches

The main drawback of all the afore mentioned *Divide et Impera* approaches is that it is not possible to elicit roles that spread across several partitions.

Yet, such roles can be very useful to system administrators. For instance, it is likely that users spreading across different partitions (e.g., different organization units) have common entitlements that simply permit access to top-level resources (i.e., to resources at their entry points). A so called "structural" role might control access to an application's entry point, whereby a user could only open that application if he or she had been assigned to a structural role that grants that access. Conversely, "functional" roles are defined as controlling access to resources within applications. The role engineering process should include the definition of both structural roles and functional roles [22].

The main objective of this section is to offer an approach that essentially provides practical and usable solutions to the role mining problem, improving the scalability of the role mining solutions while eliciting structural roles. Hence, overcoming the main limitation of existing *Divide et Impera* approaches. We focus on making the role mining problem tractable by reducing the problem size without sacrificing on utility and accuracy. The proposed approach does this by effectively compressing the original access control dataset, analyzing the compressed dataset to identify interesting portions, and reconstructing the portions of the original dataset that are worth investigating. Scalability is assured by targeted partitioning that ensures that the created sub-problems focus only on the interesting portions of the original dataset—and are therefore orders of magnitude smaller than the original one. At this point, *any* existing role mining algorithm can be used to analyze user-permission assignments within these subsets. Thus, our approach is agnostic to the specific role mining methodology used, namely it can be used in conjunction with them to enhance their scalability.

4.3.1 The Six Step Methodology

Our approach relies on the fact that it is possible to significantly reduce the computational burden of role mining while still maintaining utility by judiciously partitioning, locally analyzing, and recombining the given data. We can thus decompose our approach into six successive steps, as shown in Figure 4.8. In the first step the original dataset is decomposed into several partitions. In step 2, similar clusters of users are identified within each partition. In step 3, we develop a significantly smaller set of representative users that represent each cluster. Thus, steps 1-3 can be viewed as the process of building a compressed dataset. In the fourth step, the compressed dataset is evaluated to find portions that are interesting to analyze. In step 5, the corresponding portions of the original dataset are rebuilt. Once these interesting portions are identified, in step 6, role engineering can be carried out over these portions. Note that any existing (or new) role mining algorithm can be used at

[tb]

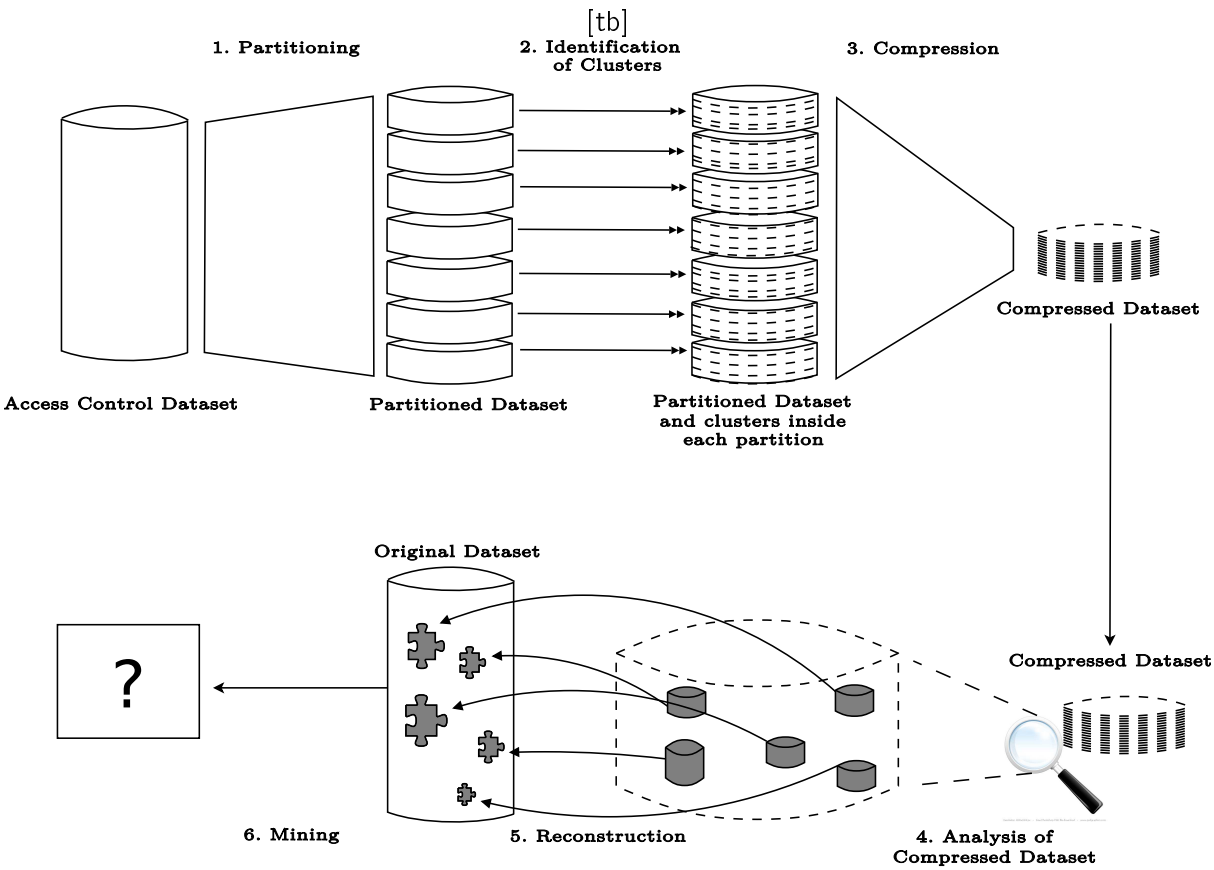
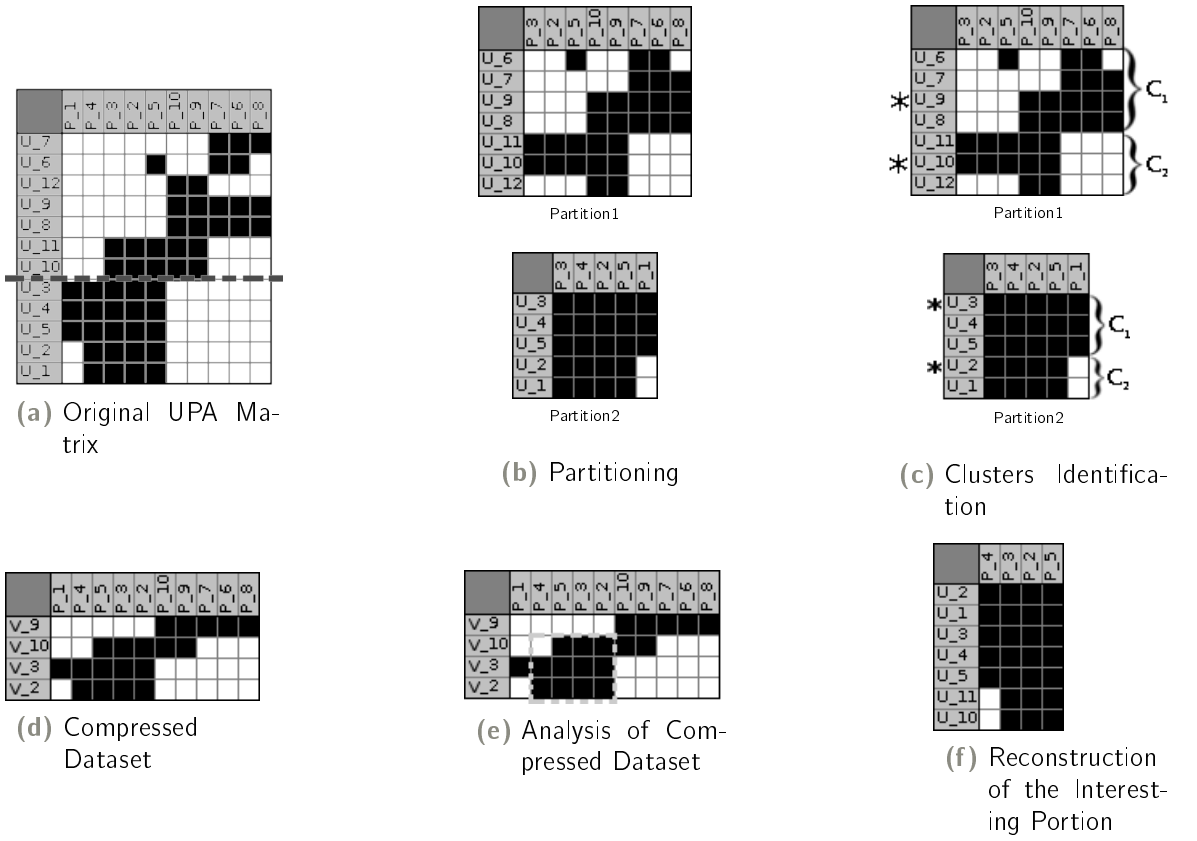


Figure 4.8 The Proposed Methodology at a Glance

this point. Indeed, our methodology does not rely on any specific role mining algorithm, and is agnostic to the specific mining approach taken. We now describe in detail each individual step. Figure 4.9 depicts a small access control dataset of 12 users, 10 permissions, and 51 user-permission assignments that we use in the following to explain each step. Note that following the various steps, the users and permissions represented in the matrices have been automatically permuted to aid in the exposition (this is not necessary for any of the steps, it is simply to give a more intuitive picture of the approach).

Step 1. Partitioning

In this step, the original dataset is decomposed into several disjoint subsets. This is similar to past work [14, 15] that also attempts to make analysis of large scale data feasible. However, in all prior approaches, each partition is



(a) Original UPA Matrix

(b) Partitioning

(c) Clusters Identification

(d) Compressed Dataset

(e) Analysis of Compressed Dataset

(f) Reconstruction of the Interesting Portion

Figure 4.9 An Expository Example

analyzed independently, without searching for the roles that extend across different partitions. On the contrary, in our approach, the subsequent steps specifically focus on the identification of all roles.

Partitioning can be done in many ways – for example, by utilizing business information, following some other rule of thumb, or even at random. However, it has been shown that using business information is typically preferable to the others since it generates more meaningful roles [14]. Therefore, the decomposition approach introduced in Section 4.2 can be used. Business information includes both user and permission attributes – any of these can be used. For example, it is possible to partition users by the country they are working in, or by organizational branches. Formally, given $U \subseteq USERS$, the partition of UP induced by U is defined as the set:

$$\{\langle u, p\rangle \in UP : u \in U\} \tag{4.28}$$

Similarly, the access control dataset can be partitioned using permission attributes as well. In this case, a subset $P \in PERMS$ induces the partition

$$\{\langle u, p\rangle \in UP\colon p \in P\}$$

For the sake of clarity, in the following we will only consider partitions induced by users attributes, though either way above is fine. The bold dotted line between users U_10 and U_3 in Figure 4.9(a) represents a partitioning based on some business information that splits the users into the two partitions $\{U\_7, U\_6, U\_12, U\_9, U\_8, U\_11, U\_10\}$ and $\{U\_3, U\_4, U\_5, U\_2, U\_1\}$, that are respectively indicated with Partition 1 and Partition 2, and depicted in Figure 4.9(b).

Step 2. Identification of Clusters

In step 2, we analyze each partition independently, trying to identify clusters of users, from which representative users will be picked. Since each partition is analyzed independently, the memory load is correspondingly reduced and it is also easily possible to parallelize this step.

To identify clusters, we use the well known clustering algorithm: *Partitioning Around Medoids* (PAM) [47, 79]. This is similar to the k-means clustering algorithm except that medoids are used instead of means and dissimilarities are used instead of distances. The "medoid" of a cluster is that object whose average dissimilarity to all the other objects in the cluster is minimal. PAM is used within each partition to identify the clusters of users. This algorithm has two key properties:

1. It is less subject to outliers than other clustering algorithms;
2. It performs clustering with respect to any specific distance metric.

These properties make it especially suitable for access control data since outliers are quite prevalent in these datasets and can cause major security violations, and the notion of user distance is not well defined. The distance metric that is used to evaluate the similarity of two users is the one introduced in Definition 4.7. For sake of clarity we report it in the following:

$$s_u\left(u_1, u_2\right) = \frac{\left|perms(u_1) \cap perms(u_2)\right|}{\left|perms(u_1) \cup perms(u_2)\right|}. \tag{4.29}$$

where $u_1, u_2 \in USERS$. Now, the distance or dissimilarity between two users u_1 and u_2 is defined as:

$$D\left(u_1, u_2\right) = 1 - s_u\left(u_1, u_2\right). \tag{4.30}$$

We now describe the basic PAM algorithm [47], and then discuss our modified heuristic for it. The PAM algorithm proceeds in five steps:

1. Randomly select k initial medoids.
2. Associate each user to the closest medoid.
3. The cost of a configuration is defined as the sum of the distances of each non-medoid to its closest medoid. Now, for each medoid, and for each non-medoid, the cost of the new configuration achieved after swapping the two, is computed.
4. The configuration minimizing the cost is selected, and the previous two steps repeated until no further improvement is possible.
5. The set of medoids, and their associated objects, are finally returned.

The computational cost of this algorithm is $O\big(k\,(n-k)^2\big)$, where k is the number of medoids, and $n-k$ is the number of non-medoids. Usually $k \ll n$, therefore the factor of k can be ignored, and $n-k \approx n$. Therefore, the computational cost is really $O\big(n^2\big)$, i.e., quadratic with respect to the number of users. The quadratic cost is due to the fact that all possible swaps of medoid and non-medoid objects are considered before selecting the one minimizing the overall cost.

Since the number of users is typically quite high, to reduce the cost of the basic algorithm described above, we propose a modified heuristic presented in Algorithm 4.3, where instead of checking all possible swaps, a small number of random swaps are chosen, and a swap carried out immediately if the new cost is lower than the current cost. Thus, an additional parameter to the algorithm is s, the number of tentative swaps executed.

The computational cost of Algorithm 4.3 is $O\big(sk(n-k)\big)$, where s is the number of tentative swaps, k is the number of medoids, and n is the number of users. Asymptotically the algorithm leads to a local minimum when s grows. Our experimental results (in Section 4.3.2), show that very good results can be achieved even with very small values of s.

A key parameter for clustering is the number of medoids (i.e., number of clusters) that are being searched for in each partition. One way of selecting this is through computing the silhouette coefficients [47], that is a measure of how tightly grouped all the data in the clusters are. Algorithm 4.3 is executed using different values of k, the silhouette coefficients are computed in each case, and the number of medoids giving the maximum silhouette coefficient chosen. However, computing the silhouette coefficient for different k is a very time consuming process, and in the experimental section we will use a

4.3 Heuristic Partitioning Around Medoids

```
1: procedure HeuristicPAM(Set of Users USERS, number of swaps s, number of
   medoids k)
2:     MEDOIDS = Randomly selected u_1,...,u_k ∈ USERS;
3:     for i = 0...s do
4:         Randomly select u ∈ USERS : u ∉ MEDOIDS ;
5:         Calculate the cost c of the new configuration, where u and its medoid
   are swapped;
6:         if c < cost of the old Configuration then
7:             Swap u and its medoid;
8:         end if
9:     end for
10:    return MEDOIDS
11: end procedure
```

given percentage of users in each partition as medoids. Our results show that high quality results can be achieved by simply selecting only 5% of users as medoids.

Figure 4.9(c) shows the result obtained by applying Algorithm 4.3, with $k = 2$, in both of the partitions depicted in Figure 4.9(b). The clusters identified are highlighted with curly braces and the corresponding medoids are denoted with an asterisk: thus, C_1 containing the users U_6, U_7, U_8, U_9 (with medoid U_9) and C_1 containing the users U_10, U_11, U_12 (with medoid U_10) are discovered in Partition 1. Similarly, U_1, U_2 (with medoid U_2) and U_3, U_4, U_5 (with medoid U_3) are identified in Partition 2.

Step 3. Compression

After identifying the user clusters (and associated medoids), we analyze each of the clusters in order to create a new compressed dataset. In this dataset, each cluster is represented by just one (virtual) user. The binarization procedure described in Algorithm 4.4 is used to identify the permissions to grant to this virtual user. Essentially, given a cluster of users U, and a binarization threshold t, each permission $p \in PERMS$ is granted to the virtual user if and only if $\texttt{support}_U(p) \geq t$, where $\texttt{support}_U$ indicates the percentage of users possessing the permission p within the cluster U. Thus, the resulting compressed dataset contains one user for every cluster of users identified in Step 2. Further, the virtual user does not necessarily have the same permissions as the medoid. Indeed, the threshold t plays a key role in determining

4.4 Binarization Procedure

```
1: procedure BinarizationProcedure(ClusterOfUsers clust, threshold t)
2:     create new user virtualUser;
3:     for Permission p ∈ PERMS do
4:         if support_U(p) ≥ t then
5:             Grant p to virtualUser
6:         end if
7:     end for
8:     return virtualUser
9: end procedure
```

this: for example, when $t = 1$, only the permissions shared by all the the users in the cluster will be granted to the virtual user, where as with $t = 0$ all the permissions that are granted to at least one user in the cluster will be granted. In the experimental validation, we experiment with different values for the threshold, and provide some guidance on how this can be chosen.

Note that as before, this step can also be fully parallelized thus guaranteeing a noticeable speed-up of the overall analysis. Further, the kind of compression that we perform will not cause the presence of unauthorized permissions inside the final list of roles, indeed the roles will be elicited only after analyzing the compressed dataset (Step 4) and rebuilding portions of the original one (Step 5). Figure 4.9(d) illustrates Step 3 in the context of our example. The compressed dataset is formed of four virtual users, each corresponding to one of the clusters highlighted in Figure 4.9(c). The threshold used was $t = 0.5$, i.e., only those permissions that are supported by at least half of the users of the cluster are granted to the virtual user. For example, the virtual user V_9 represents the cluster U_9, U_6, U_7, U_8, and is granted the permissions P_10, P_9, P_7, P_6 and P_8. This is correct, since only those permissions are also owned by at least two of the users in the cluster. The virtual users V_10, V_3 and V_2 are also generated in the same way from the corresponding user clusters.

Step 4. Analysis of Compressed Dataset

After the prior three steps are completed, a compressed dataset has been built that effectively summarizes the original dataset. Now, we analyze this compressed dataset in order to discover portions of the original dataset that are worth inspecting.

Note that access control datasets can be represented as (often, sparse) binary matrices, where rows represents users, columns represents permissions, and a 1 in a cell indicates the presence of a user-permission assignment, while a 0 represents the absence. If we consider such a matrix, the portions worth inspecting are composed of "dense" and "large" subsets of rows and columns, since they allow us to find roles that can handle many user-permission assignments. Due to the use of the binarization procedure (Algorithm 4.4), there may exist permissions in the original user clusters that have not been granted to the corresponding virtual users because of low support. However, these should still be potentially taken into consideration.

To do so, we use the maximal pseudo-bicluster tool that has been introduced in [12]. A pseudo-bicluster is a pattern that can be seen as a superset of any interesting role. We can then focus the role mining on a subset of these patterns, that identifies portions of the dataset (in this case our compressed dataset) that are worth successively analyzing.

Definition 4.14 (Relevance) The *relevance* of a Maximal Pseudo-Bicluster $B = \langle U, P \rangle$ is defined as the number of generators of B, and indicated with $\varrho(B)$. Formally:

$$\varrho(B) = |\hat{U}| \times |\hat{P}|$$

Since $\hat{U} \in U$ and $\hat{P} \in P$, it turns out that $\varrho(B) < |U| \times |P|$. It means that a Maximal Pseudo-Bicluster that involves few users and few permissions cannot have a high relevance. Further, the relevance of B reaches the maximum when $\hat{U}$ is equal to U, and $\hat{U}$ is equal to P. Indeed, in that case all the users and the permissions of the Maximal Pseudo-Bicluster can be managed with just one role. Since the relevance of a Maximal Pseudo-Bicluster is an absolute value, we use the normalized version:

Definition 4.15 (Normalized Relevance) The *normalized relevance* of a Maximal Pseudo-Biclusters $B = \langle U, P \rangle$ is defined as:

$$\overline{\varrho}(B) = \frac{\varrho(B)}{|UP|}$$

where $UP \in USERS \times PERMS$ is the set of the existing user-permission assignments.

Note that the normalized relevance of a Maximal Pseudo-Bicluster is correlated with the frequency of a pseudo-role (Definition 4.6). It can be shown that $0 < \overline{\varrho}(B) \leq 1$, indeed $\varrho(B)$ is always greater than 0 and lower than, or equal to $|UP|$. Maximal Pseudo-Biclusters that have a high normalized relevance correspond to those portions of a dataset that when inspected are likely

to have roles that can be used to manage many user-permission assignments. Therefore, in this step, we search for the maximal Pseudo-Biclusters having high normalized relevance within the compressed dataset. Note that since the virtual users many not have some of the permissions granted, the maximal pseudo-bicluster may not correspond to a single role, but it does narrow the search area when looking for large roles. Several strategies are possible to select the maximal Pseudo-Biclusters – for example, we can select in descending order of size, up to a fixed number, or we can take a given percentage of the existing maximal Pseudo-Biclusters. Once this subset has been selected, we can proceed to Step 5.

With respect to our expository example (Figure 4.9), when the compressed dataset shown in Figure 4.9(d) is analyzed to identify (and order) maximal pseudo-biclusters, three maximal pseudo-biclusters are identified with the same maximum normalized relevance. The first is composed of the virtual users V_10, V_3, V_2, and the permissions P_4, P_5, P_3, P_2. Its normalized relevance is equal to $3/51$, indeed $|\hat{U}| = 1$, $|\hat{P}| = 3$ and $|UP| = 51$. The second Maximal Pseudo-Biclusters with the same relevance is composed of the virtual user V_9, and the permissions P_10, P_9, P_7, P_6, P_8. In this case, it involves only one user, and if we are searching for roles that extend across partitions, it can be discarded. The third Maximal Pseudo-Biclusters is identified by the virtual users V_10, V_3, V_2, and the permissions P_5, P_3, P_2, P_10 and P_9. This Maximal Pseudo-Biclusters identifies another area that should be further analyzed in the original dataset.

Step 5. Reconstruction

Once the maximal pseudo biclusters with high normalized relevance have been identified, we recover the portions of the original dataset that these maximal pseudo biclusters correspond to. This phase can be seen as the expansion of the portions highlighted in the previous step. Algorithm 4.5 depicts the reconstruction procedure. We assume that a hash map has been maintained (in the prior steps), that, given a virtual user, returns the original users from which the virtual user has been built (i.e., gives the association of virtual users to user clusters). Now, the idea is simple. We start by creating a new Maximal Pseudo-Cluster (Line 2) with no users assigned, and then, using the hash map, we gradually add the original users into the bicluster (Line 3-5). The procedure returns the new maximal pseudo-bicluster. Thus, it identifies an area of the original dataset that is worth inspecting. Therefore, for each maximal pseudo-bicluster selected in Step 4, we reconstruct the corresponding original pseudo-bicluster, and each of them identifies a given subset of the original dataset. Compared with the original dataset, these subsets involve only few

4.5 Reconstruction of the Original Portions

```
1: procedure Reconstruct(MaximalPseudoBicluster B = ⟨U, P⟩)
2:     create new MaximalPseudoBicluster B′ = ⟨∅, P⟩;
3:     for User u ∈ U do
4:         B′.Add(HashMap(u))
5:     end for
6:     return B′
7: end procedure
```

users and few permissions. So, they are easier to analyze using any role mining algorithm, and this analysis can even be executed in parallel.

Figure 4.9(f) shows the result of the reconstruction of the Maximal Pseudo-Bicluster highlighted in Figure 4.9(e). The Maximal Pseudo-Bicluster involves the virtual users V_10, V_3 and V_2. Looking at Figure 4.9(c), it can be seen that they respectively identify the sets: $\{U\_11, U\_10, U\_12\}$, $\{U\_3, U\_4, U\_5\}$ and $\{U\_2, U\_1\}$. For these users, only the permissions P_4, P_5, P_3, P_2 are taken into consideration for the reconstruction, indeed only these permissions belong to the highlighted Maximal Pseudo-Bicluster. It can be seen that, even if the permission P_1 is granted to U_3, U_4, U_5, it will not be considered in the reconstruction phase. Further, the portion of the original dataset that we rebuild is effectively an area where we can find meaningful roles that involves users belonging to different partitions. Also, it has to be considered, that this portion of the original matrix involves the 51% of all the original user-permission assignments, but only 4 permissions out of 10, and 7 users out of 12. This gain will increase when thousands of users and permissions are involved in the original dataset, and with the sparsity of the dataset.

Step 6. Mining

Once the interesting portion of the dataset is reconstructed, any role mining mining algorithm can be used to actually discover the roles. In effect, our procedure is completely agnostic to the actual role mining methodology used and serves to zoom attention to the interesting areas of the original dataset. Therefore, Step 6 of our methodology corresponds to the independent analysis of the reconstructed portions. Since the analysis is executed independently, it can be also executed in parallel, thus further reducing the runtime complexity.

Table 4.2 Properties of the real Datasets analyzed

DATASET	USERS	PERMISSIONS
HP Customers	10021	277
HP Apj	2044	1164
HP Americas Small	3477	1587
HP Americas Large	3485	10127

4.3.2 Experimental Evaluation

We now experimentally validate our approach. To do this, we have evaluated our approach on several datasets of differing characteristics, both real and synthetic. We first discuss these datasets, then introduce the evaluation strategy, and finally present the results achieved.

Datasets

Our primary case study has been carried out on a large private organization that has more than two thousand users, more than six thousand permissions, and a total of more than 100,000 user-permissions assignments. In order to preserve the privacy of the organization it is not possible to reveal more information about the activities performed by the users, the applications used, and the corresponding permissions granted. However, we actually had detailed data on this dataset, including both user and permission attributes such as "Job Title", "Cost Center", "Division", "Organizational Unit", "Application" etc. Without using any compression, the binary matrix representing all of these user-permissions assignments requires almost 12Gb of space. Using other (sparse representation) data structures this requirement can be reduced, but even considering only 2 bytes to store each one of the 100,000 user-permission assignments (1 byte to store the user index, and 1 byte for the permission index), roughly 200Mb of memory space is required. Indeed, this dataset is not the largest that we had available, but we choose it because it was still possible to analyze it using the standard approaches. Thus, we use this dataset to compare the performance of our methodology, with respect to a traditional approach that performs the analysis of the whole dataset. Note that with larger datasets traditional analysis becomes even more infeasible. Indeed, larger datasets in our possession were impossible to analyze, both because computa-

tional time and memory required.

To execute the first step of our methodology, we used a business attribute that was at our disposal, that is the "Job Title". As per this attribute, a total of 95 different partitions have been created during the "Partitioning" step.

In order to confirm the effectiveness of our methodology we analyzed also several other access control datasets that are publicly available [29]. In particular, we analyzed all the dataset at our disposal with more than 1000 users. A description of their main properties can be found in Table 4.2. Since no business information is available for these datasets, we executed the Step 1 randomly partitioning the users in a number of partitions equal to the 1% of the number of users.

To complete the evaluation, we applied our methodology to synthetic datasets as well. Each dataset has been generated using the following procedure. We created an empty matrix composed by 10000 rows and 1000 columns. In each matrix, a given number n of subsets of rows and columns have been randomly chosen. Each subset of rows, and each subset of columns, counts a number of elements that are proportional to $100 \times x^2$ and $20 \times x^2$, where x is a random number uniformly chosen between 0 and 1. The elements of the matrix that belongs to one of such subsets are set to 1, while the other ones are set to 0. Using this procedure we generated different binary matrices, and we used them as access control datasets that have 10000 users, and 1000 permissions. In other words, we granted a permission to a specific user if, and only if, the corresponding cell in the matrix was set to 1.

Experiments Setup

In the following, we will report on several experiments that have been executed to evaluate the performance of our methodology. In particular, since our approach does not depend on any particular role mining algorithm, we will execute our experiments searching for a general pattern inside our data: the maximal biclusters. These patterns are related to the concept of closed itemsets [83], that are used in the discovery of association rules, strong rules, correlations, sequential rules, episodes, multidimensional patterns, and many other important tasks [43].

In order to estimate the performance achieved, we will compare the results with and without utilizing our approach. In particular, given the datasets previously described, we will search for maximal biclusters, and we will compare this list of actual maximal biclusters with the maximal biclusters generated by using our methodology. Since each maximal bicluster identifies a set of users and a set of permissions, we use two similarities indices as metrics: the similarity of two sets of users, and the similarity of two sets of permissions.

Definition 4.16 (Similarity of two sets of users)

$$Sim_u\left(U_1, U_2\right) = \frac{\left|U_1 \cap U_2\right|}{\left|U_1 \cup U_2\right|} \tag{4.31}$$

where $U_1, U_2 \subseteq USERS$

Definition 4.17 (Similarity of two sets of permissions)

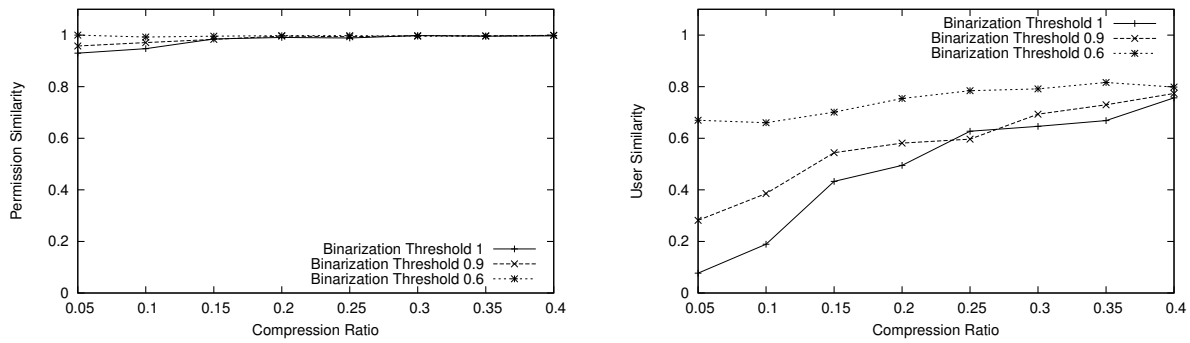
$$Sim_p\left(P_1, P_2\right) = \frac{\left|P_1 \cap P_2\right|}{\left|P_1 \cup P_2\right|} \tag{4.32}$$

where $P_1, P_2 \subseteq PERMS$

Given the two lists of maximal biclusters, each element of the first list is compared with all the elements in the second list. Among all these pairs, the highest user and permission similarities are taken into consideration, and the arithmetic mean of all the similarities (that we indicate with "*Users Similarity*" and "*Permissions Similarity*") is the index that we use to compare the two lists of maximal biclusters . In summary, given the list of maximal biclusters generated by our six-steps methodology and the one generated by considering the whole dataset, high "*Users Similarity*" means that the two lists of roles are very similar considering the users to role assignments. While, high "*Permission Similarity*" means that the two lists of roles are very similar considering the permission to role assignments. If the "*Users Similarity*" and the "*Permission Similarity*" calculated as above are both equal to one, then the accuracy of our six steps methodology is maximal, indeed it is able to elicit the same list of maximal biclusters generated analyzing the dataset as a whole.

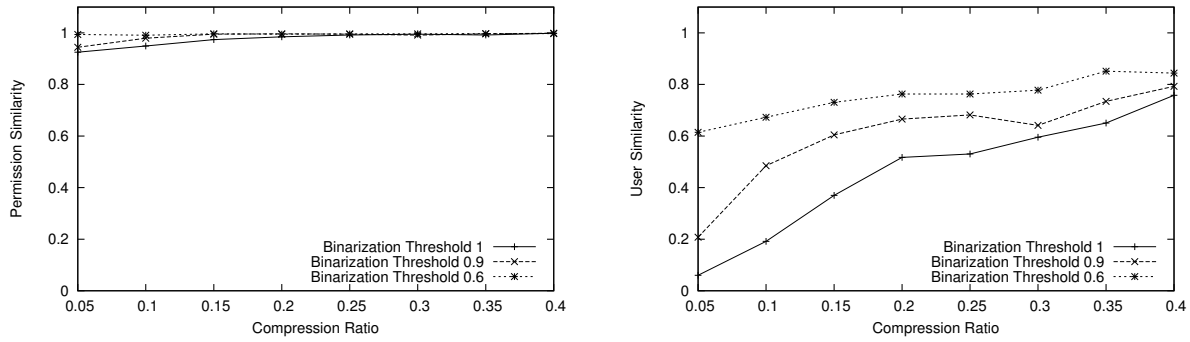
Results

We first discuss the results achieved on our case study and then look at the results with the other datasets. Figure 4.10(a) reports the "Permissions Similarity" achieved. The number of swaps s used in Step 2 has been set equal to 100. The compression ratio used to determine the number of medoids in each partition is depicted on the x-axis, while the values of users and permissions similarities are reported on the y-axis. The results for different binarization thresholds are reported. In particular, it can be noticed that in all the cases the *Permission Similarity* is always greater than 90%(Figure 4.10(a)). It indicates that we loose only around 10% of precision when adopting our methodology, independent of the selected binarization threshold.



(a) Permission Similarity (b) User Similarity

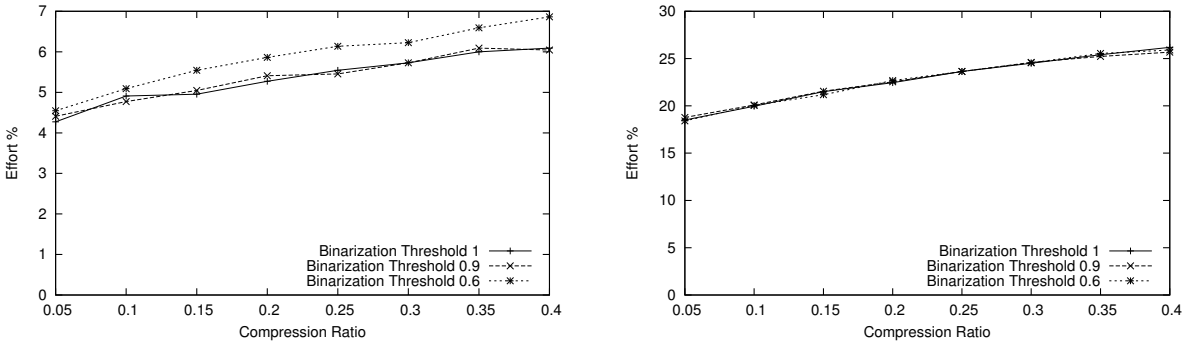
Figure 4.10 Users and Permissions Similarity of the actual list of maximal biclusters, and the one built using our six step methodology. $s = 100$



(a) Permission Similarity (b) User Similarity

Figure 4.11 Users and Permissions Similarity of the actual list of maximal biclusters, and the one built using our six step methodology. $s = 1000$

As showed in Figure 4.10(b), the *User Similarity* is instead more sensitive. This is mainly due do the fact that we are using a compression that involves users instead of permissions. A low binarization threshold allows us to achieve better results, indeed the best users similarity is reached when the binarization threshold is equal to 0.6. In this case, using only around 5% of the users



(a) Effort with s=100 **(b)** Effort with s=1000

Figure 4.12 Percentage of Effort when using our methodology

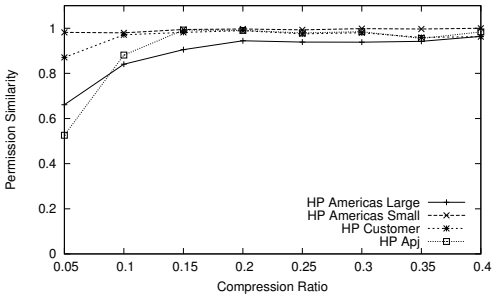
as medoids, we achieve an *User Similarity* higher than the 60%. When the binarization threshold is higher, we reach however outstanding results: Using the 25% of users, in all the cases the users similarity is greater than the 60%, even when the binarization threshold is 1.

The complexity of Algorithm 4.3 is $O\big(sk(n-k)\big)$, therefore, a higher number of swaps s requires more effort. Figure 4.11, like Figure 4.10, reports users and permissions similarities, but in this case the number of swaps executed in Algorithm 4.3 is equal to 1000, that is ten times the case illustrated in Figure 4.10. It can be seen that both the *User Similarity* and the *Permission Similarity*, are not markedly different from the results achieved in Figure 4.10. Therefore, already 100 swaps can be considered sufficient to achieve good results. Figure 4.12 compares the effort needed to search the maximal biclusters without our methodology, with the effort needed to execute our methodology with 100 and 1000 swaps. The effort is shown as a percentage: x% effort indicates that our methodology requires only x% of the computational time required without it. Figure 4.12(a) shows that the effort required is always less than the 7% when $s = 100$. With only 5% of users as medoids, the effort is even less than 5%.

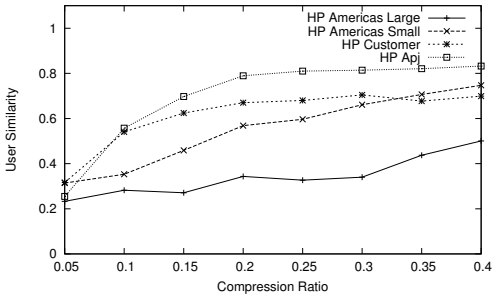
The most important outcome of our six step methodology can be seen by analyzing Figure 4.12(a), and Figure 4.10 together. It turns out that it is possible to achieve the 100% of *Permission Similarity*, and the 80% of *User Similarity* with less than the 7% of effort. This largely confirms the effectiveness and the efficiency of the proposed framework.

As discussed above, we also analyzed several other access control datasets that are publicly available [29]. In the following experiments, the threshold t has been set to 0.6, and the value s to 100. Figure 4.13(a) shows the *Permission Similarity* achieved when analyzing the four dataset described in Table 4.2, while Figure 4.13(b) shows the *User Similarity*. It can be seen that `HP AMERICAS LARGE` is the dataset that had the worst performance. This is mainly due to the fact that this dataset has a larger number of permissions than users. In cases like this, it is better to apply our methodology trying to compress permissions instead that the number of users. In all the other cases the permission similarity is higher than 80% when the compression ratio is higher then 0.05, and the users similarity is higher than 60% when the compression ratio is higher than 0.25. It is worth noticing that the results are slightly worse than in the real case illustrated in figures 4.10(a) and 4.10(b). This is mainly due to the fact that for the public datasets we do not have at our disposal any business information to drive the partitioning step, leading to assigning users randomly to the different partitions, and therefore users that are not similar at all are likely assigned to the same partitions.

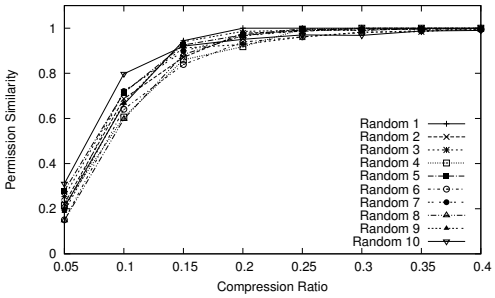
Figures 4.13(c) and 4.13(d) shows the *Permission Similarity* and the *User Similarity* of the datasets that we randomly generated using the procedure described in Section 4.3.2. Even in this case the results are similar to the real datasets that we previously analyzed: the permissions similarity is always higher than the users similarity, and a users similarity higher than roughly the 60% can be achieved when the compression ratio is higher than 0.25.



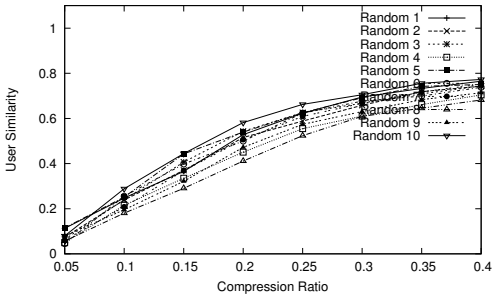
(a) Permission Similarity



(b) User Similarity



(c) Permission Similarity



(d) User Similarity

Figure 4.13 Analysis of several datasets. Since for these datasets no business information are available, we randomly partitioned the users during Step 1, creating a number of partitions equal to the 1% of the number of users. In this simulation $s = 100$ and $t = 0.6$.

5

Data Cleansing in Access Control Datasets

When dealing with any kind of real data, there must always be a phase (usually referred to as *data cleansing*) in which noise and irrelevant data are removed from the collection. As a matter of fact, most algorithms assume the data to be noise-free. This is of course a strong assumption. Most datasets contain exceptions, invalid or incomplete information, etc., which may complicate, if not obscure, the analysis process and in many cases compromise the accuracy of the results. As a consequence, data preprocessing becomes vital. Even if it is sometimes overlooked, data cleaning is one of the most important phases in the knowledge discovery process. Further, reviewing the data during this preliminary step can lead to the generation of more meaningful and human understandable patterns. Indeed, noisy data usually makes classical data mining algorithms to the discovery of multiple small fragments of the real clusters [55]. The problem is even worse for clusters which cover many rows and columns, since they are more vulnerable to noise [57].

The inaccurate data that can be found within a dataset can be usually classified in two classes:

- *missing values*, namely portions of data that are unavailable or unobserved [12].

- *outliers*, namely portions of data that are present but which appears to be inconsistent with the remainder of that set of data [46].

Both missing values and outliers arise in many practical situations. For example, some values can be out of the range of the measuring device, a signal can arrive distorted or modified because of some error in the measurement, indecision could arise due to the rounding of measurement values, etc.. When a clean and complete dataset is required, as is the case of most data mining

and clustering techniques, data analysts typically have three options before performing data analysis: to discard the rows of the matrix that contain missing data and outliers, to replace these data values with some constant, or to estimate their values [74].

One solution for noise identification and removal problem could be to repeat the experiment. However, this approach is not always viable. Indeed, depending on the investigation field and on the data that will be analyzed, repeating the experiment for each suspicious value could either require a high cost or even be impossible. Nevertheless, replacing the missing items with "plausible" values or discarding outliers always gives better results than ignoring them. One possibility is to impute or discard them by analyzing uncovered structures that reveal the nature of the relationships among the rows and the columns of the binary matrix. This approach is rooted on the consideration that in a binary matrix representing a given dataset, many rows and many columns are implicitly bound to one another. For example, in a paleontology dataset many sites (rows) are geographically close to each other, so that it is predictable that they host the same species (columns), even if no evidence have been found. Further, many species share some physical feature that strictly depends on the sites where they lived. In an access control dataset, several users (columns) are similar with respect to the permissions (rows) that are granted to them, so that it is possible that they have the same job position within the organization. Furthermore, if a group of permissions is used by a particular set of users, that may suggest that all of them are related to a particular application.

Therefore, looking at the data makes it possible to uncover their embedded relationships and leverage them to filter noise. Note that both relationships between rows, and those between columns have a particular meaning, and therefore both of them should be considered when trying to discover suspicious values. In the literature, instead, only relationships between rows are used for these purposes [3,33,39,64]. However, caution should be taken: values imputed or removed in this way are *not* real data. It is only an estimation, and it could not reflect the real values.

The detection of suspicious values is not only restricted to data cleansing applications. Indeed, it can be used in many other fields. For example, in a recommendation system, a missing value can represent an item that will probably attract the interest of a user. In an Intrusion Detection System, the imputation of outliers in a dataset that describes network accesses to monitored resources could be used to raise the required alarms. In many of these cases, the matrix can slightly change during the time. For example, considering a matrix that describes friendship between users of a given social network, imputed missing values can be used to suggest new friends of users. Each

user can add or remove friendships at any instant, continuously modifying the underlying dataset. In an Intrusion Detection System, accesses to the protected resources are added or deleted continuously, constantly modifying the dataset over time. To the best of our knowledge, algorithms that aim at finding suspicious data (missing values and outliers) do not take into account these possible slight changes of the dataset. Indeed, they always need to completely reprocess the whole dataset, without leveraging computation carried out over "past" dataset—that could differ from the new one by just a single entry.

In this chapter, we will introduce a set of algorithms that are able to highlight suspicious values within binary matrices: ABBA, ABBA* and Fast-ABBA*. The first is able to find missing values, the second is able to find both missing values and outliers, while the third introduces a fast incremental update operation that can be used when a little modification of the original binary matrix invalidates the previously computed results. We will test the performances of these algorithms on access control datasets, showing that they can be effectively used to isolate noise in this particular context.

5.1 Handling Missing values in binary matrices

An approach that is often used to impute missing values is the *k-nearest neighbors* (KNN) [73]: for each row that has a missing value in the column i, the k-nearest rows that do not have a missing value in the column i are used to impute the missing value. This set of k-nearest rows is found according to some similarity metric. In turn, the missing value is replaced with the average value for the cells on the column i within the k-nearest rows. One of the critical issues using the KNN is the choice of the parameter k. On one hand, if parameter k has a high value, rows that are significantly different from the analyzed ones can decrease the imputing accuracy. Indeed, a "neighborhood" that is too large could decrease the imputing accuracy. On the other hand, if k is too small, an overemphasis is given to small patterns. In fact, the optimal selection of k likely depends on the size of the identifiable clusters within the given dataset. Another aspect of applying KNN is the choice of a threshold t to decide if the imputed value has to be a 0 or a 1. Once each missing value has been imputed, it assumes a value between 0 and 1: the threshold t is used to switch it to 1 or to 0. To the best of our knowledge, the most frequently used approaches for missing value imputation in binary matrices always require that a parameter comparable to k is fixed *a-priori* [51, 62, 73].

In this section we address the challenge posed by the imputation of flagged and not-flagged missing values. A binary matrix with flagged missing values is a matrix where some elements are set to '$*$'. Our target is to impute these val-

ues by leveraging identifiable patterns within available data. In particular, we propose an algorithm referred to as ABBA (*Adaptive Bicluster-Based Approach*) that leverages the identifiable patterns within the data to infer missing values in binary matrices. Our approach provides several distinguishing features when compared to the other approaches similar to the k-nearest neighbors (KNN). The most important one is that ABBA does not require to fix any parameter a-priori. Indeed, the main issue in KNN-like approaches is that a fraction of the rows, *fixed before* running the algorithm, is used to impute a missing value, regardless of the identifiable patterns within the data. Further, our algorithm shows a better computational complexity for a wide range of parameters when compared to KNN. Moreover, the relevance of missing values are inferred by only one algorithm run. Another distinguishing feature is that our approach leverages the actual patterns that are identifiable within the available data, thus making it *adaptive*. Conversely, changing k in KNN requires a new run of the algorithm. Thus, obtaining the desired results with more computational time.

Theorem 5.1 *All the generators of a maximal pseudo-bicluster $B = \langle R, C\rangle$ belong to rows and columns that have 1's in the same positions, formally:*

$$\forall m_{ij}, m_{\ell k} \in M \ : \ m_{ij} \text{ and } m_{\ell k} \text{ generate } B \implies \\ \forall t \in \hat{C},\ \forall s \in \hat{R} \ : \ m_{it} = 1,\ m_{\ell t} = 1,\ m_{sj} = 1,\ m_{sk} = 1$$

PROOF The proof is by contradiction. Let m_{ij} and $m_{\ell k}$ be two elements of the matrix M that generate respectively the maximal pseudo-bicluster $B = \langle R, C\rangle$ and $B' = \langle R', C'\rangle$, and suppose $B = B'$. If m_{ij} and $m_{\ell k}$ belong to two columns that do not have 1's in the same positions, then $R \neq R'$; indeed, $R = \{s \in [n] \mid m_{sj} = 1\}$ and $R' = \{s \in [n] \mid m_{sk} = 1\}$, and so $B \neq B'$. If m_{ij} and $m_{\ell k}$ belong to two rows that do not have 1's in the same positions, then $C \neq C'$; indeed, $C = \{t \in [m] \mid m_{it}\}$ and $C' = \{t \in [m] \mid m_{\ell t}\}$. Therefore, it must be $B \neq B'$, and this is a contradiction. ■

Lemma 5.1 *Let MPBS be the set of all the maximal pseudo-biclusters that exist within the matrix M. Then, $|MPBS| \leq |M|$.*

PROOF Proof follows from the previous observations. Indeed, each cell that is equal to 1 generates exactly one maximal pseudo-bicluster, and each maximal pseudo-bicluster is generated by at least one cell. Since we have exactly $|M|$ cells equal to 1 within the matrix, then $|MPBS| \leq |M|$. ■

The reason why we use the maximal pseudo-bicluster concept is that it can be used to impute missing data, for both flagged and not-flagged missing values. For each element $m_{ij} \in M$ that is equal to 1, it is possible to generate a maximal pseudo-bicluster $B = \langle R, C \rangle$ from m_{ij} by setting $\mathrm{R} = \{\ell \in [n] \mid m_{\ell j} = 1\}$ and $\mathrm{C} = \{k \in [m] \mid m_{ik} = 1\}$. As we have seen before, a bicluster can contain 0, 1, and '$*$' in the case of matrices with flagged missing values. The intuition is that the cells of a maximal pseudo-bicluster equal to 0 (in the not-flagged matrices) or equal to '$*$' (in the flagged matrices) are likely to be '1' since they belong to a pattern. Note that the less cells are equal to 0 and/or '$*$' within a maximal pseudo-bicluster B, the more B is close to being a bicluster, and the more the missing values contained in B are likely to be 1's. This is the rationale that we will use to impute missing data in binary matrices.

We will now introduce another relevance index for each element $m_{ij} \in M$ that is based on the relevance of maximal pseudo-biclusters. With this index, we will be able to evaluate missing values m_{ij} by leveraging all the patterns represented by the maximal pseudo-bicluster which m_{ij} belongs to. This means that each missing value of M will be evaluated using all the patterns (i.e., maximal pseudo-roles) that we are able to discover within the available data, weighted by their relevance.

Definition 5.1 (Relevance of a Cell) Given an element $m_{ij} \in M$, and let *MPBS* be the set of all existing maximal pseudo-biclusters within the data, the relevance of m_{ij} is the sum of the relevances $\varrho(B)$ of all the maximal pseudo-biclusters B which m_{ij} belongs to. Formally:

$$\sigma(m_{ij}) = \sum_{B \in MPBS : m_{ij} \in B} \varrho(B).$$

It is possible to normalize the value of each $\sigma(m_{ij})$ with respect to the maximal index found in the matrix M. In this way, the index value will range from 0 to 1. In the following, we will always consider this normalized version. By evaluating $\sigma(m_{ij})$ for a given m_{ij} that is equal to '$*$' (flagged matrix) or 0 (not-flagged matrix) we can impute the missing value according to the identified patterns within the available data. In this way, each missing value is imputed considering *all* and *only* those patterns that could involve it. This does not happen in other approaches such as KNN. Indeed, in that case each missing value is evaluated using a fixed number of rows (i.e., the k-nearest rows): it may occur that the result is biased, because of not having considered a sufficient number of relevant rows, or, even worse, by averaging rows that are completely unrelated. Conversely, in our approach each missing value is evaluated using a variable number of patterns, that depends on the given data set. A high value for the index $\sigma(m_{ij})$ indicates both a high relevance

and a high number of patterns involved. As for flagged matrices, we will evaluate the relevance index for each element equal to '$*$'. Instead, when we are dealing with not-flagged missing values, we will evaluate the relevance of all the elements $m_{ij} \in M$ such that $m_{ij} = 0$.

5.1.1 ABBA: Adaptive Bicluster-Based Approach

In order to identify an algorithm that evaluates the relevance of missing values, we first make some considerations on the introduced indices. In particular, definitions 4.14 and 5.1 suggest a way to practically compute the relevance values. By simply combining the two indices, the following holds:

$$\sigma(m_{ij}) = \sum_{B \in MPBS : m_{ij} \in B} \varrho(B) = \sum_{B \in MPBS : m_{ij} \in B} \sum_{m_{\ell k} \in B} \gamma(m_{\ell k}, B).$$

where $\gamma(m_{ij}, B)$ is defined as

$$\gamma(m_{ij}, B) = \begin{cases} 1, & m_{ij} \text{ generates } B; \\ 0, & \text{otherwise.} \end{cases}$$

Notice that elements which do not belong to B cannot generate it, thus we can replace B with M in the second sum. Moreover, only elements equal to 1 can be generators. Hence, we can rewrite the previous equation in the following way:

$$\begin{aligned} \sigma(m_{ij}) &= \sum_{B \in MPBS : m_{ij} \in B} \sum_{m_{\ell k} \in M : m_{\ell k} = 1} \gamma(m_{\ell k}, B) \\ &= \sum_{m_{\ell k} \in M : m_{\ell k} = 1} \sum_{B \in MPBS : m_{ij} \in B} \gamma(m_{\ell k}, B), \end{aligned}$$

Since $\gamma(m_{\ell k}, B)$ holds true only when $m_{\ell k}$ generates B, the second sum has non-zero elements only when B is the maximal pseudo-bicluster generated by $m_{\ell k}$, namely $B_{m_{\ell k}}$. Additionally, according to the second sum we have that m_{ij} must belong to maximal pseudo-biclusters $B_{m_{\ell k}}$. Formally:

$$\sigma(m_{ij}) = \sum_{m_{\ell k} \in M : m_{\ell k} = 1} \rho(m_{ij}, B_{m_{\ell k}}) \tag{5.1}$$

where

$$\rho(m_{ij}, B_{m_{\ell k}}) = \begin{cases} 1, & m_{ij} \in B_{m_{\ell k}} \\ 0, & \text{otherwise.} \end{cases} \tag{5.2}$$

5.1 The ABBA algorithm

```
 1: procedure EvaluateMissing(M)
 2:     for all m_lk ∈ M s.t. m_lk = 1 do
 3:         for all m_ij ∈ B_{m_lk} s.t. m_ij = '*' do
 4:             m_ij.count ← m_ij.count + 1
 5:         end for
 6:     end for
 7:     return M
 8: end procedure

 9: procedure Binarize(M, t)
10:     for all m_ij ∈ M s.t. m_ij.count > 0 do
11:         if m_ij.count > t then
12:             m_ij ← 1
13:         else
14:             m_ij ← 0
15:         end if
16:     end for
17:     return M
18: end procedure
```

We used Equation 5.1 to define an algorithm referred to as ABBA (*Adaptive Bicluster-Based Approach*), that is described in Algorithm 5.1. First, we calculate the set of all maximal pseudo-biclusters by scanning all elements $m_{ij} \in M : m_{ij} = 1$. In turn, the relevance of missing values is determined by checking their membership to the generated maximal pseudo-biclusters. In this way, all the identifiable data patterns that could have some relation with the missing value are involved in its imputation, according to their relevance.

Procedure EvaluateMissing is next described. The loop from Line 2 to Line 6 generates a maximal pseudo-bicluster for each element $m_{ij} = 1$ of the matrix M. The loop from Line 3 to Line 5 increases the counter of each missing value contained in the maximal pseudo-bicluster just created. Notice that the condition $m_{ij} =$ '$*$' in Line 3 assumes that we are dealing with a flagged matrix. If this is not the case, we can just replace this condition with $m_{ij} = 0$. At the end of the algorithm, each missing value (i.e., elements with $m_{ij} =$ '$*$' in the flagged version, $m_{ij} = 0$ in the not-flagged version) will contain a value that corresponds to $\sigma(m_{ij})$ in its data field referred to as 'count'. Then, each counter can be optionally normalized with the maximum value found for the index. After having calculated the relevances through EvaluateMissing, the

procedure BINARIZE can be called. It takes in input the matrix M and a threshold t for the relevance index, thus giving back the final binarized version of M.

The correctness of Algorithm 5.1 is guaranteed by Equation 5.1. As for its computational complexity, it is $O(\mu|M|)$, where $|M|$ is the number of elements of the matrix M that are set to 1, and μ is the number of missing values. Indeed, the first loop is executed for each element of the matrix M that is equal to 1. The maximal pseudo-bicluster can be determined in constant time, for example by using an hash table that gives all columns with 1's for a given row, and all rows with 1's for a given column—the hash table can be created in $O(|M|)$. The internal loop is executed at most μ times, and the operation of Line 4 can be executed in constant time. The worst case is when $\mu = |M| = (nm/2)$, namely when half the matrix is filled by 1's half by '$*$' (in the flagged version) or 0's (in the not-flagged version). Yet, this seldom happens. When the number of missing values represents a small fraction of the data, or the matrix is sparse, our approach outperforms other algorithms such as KNN, that has a computational complexity $O(n^2m)$ [73].

Threshold Tuning The *multiple imputation* method [66] is a simulation technique that replaces each missing data with a set of $m > 1$ plausible values. The m versions of the complete data are analyzed by standard complete-data methods. According to this definition, our approach can be considered a multiple imputation method. Indeed, we first impute missing values with the procedure EVALUATEMISSING described in Algorithm 5.1. In turn, by changing a threshold t that does not require the complete re-imputation of missing data, it is possible to generate m versions of the dataset through the procedure BINARIZE. Each version can subsequently be analyzed by trying to find the one that better reaches the target function. The tuning of the threshold t depends on the final objective of the data analysis. First, a metric must be defined to measure how well the objective has been reached. Then, it is possible to use this metric to evaluate the imputed matrix. This can be an iterative process, executed several times with different thresholds, thus choosing the threshold value that provides the best result.

As a possible application of our algorithm, consider the problem of minimizing the number of biclusters required to "cover" all the 1's within the matrix. Many real-world examples require the identification of a "compressed" matrix representation through a list of biclusters, such as minimizing the number of relationships required to manage permissions granted to users [6, 7, 10, 11]. In particular, let us consider the case of a binary access control matrix, where rows represent users, and columns represent permissions. A

cell representing a user-permission assignment is then set to 1 if the user must have the permission granted, 0 if not, and '$*$' in the case that he could have that permission granted but it is not strictly needed to accomplish his work, or in the case that it is not clear whether he should have that permission granted or not. The so called *role mining problem* [9] is to find subsets of users that share the same subset of permissions minimizing a given cost function, thus creating a set of roles that can be efficiently managed by security administrators. Typically, missing values in this scenario are neglected, or more often they are *not* flagged missing values. Thus, the quality of the clustering is severely biased by them. Yet, automatically switching all the imputed values might not always be appropriate. Rather, it would be advisable to submit these results to a checker. In our example, we should not forget that security still remains the main objective. Hence, the system administrator should carefully check the missing values, one by one. Checking all possible missing values could be an unfeasible task. By having a sorted list of the missing user-permission relations based on the computed relevance index, a system administrator can only focus on the most relevant ones, evaluating them in the reverse order. In this case, a reasonable metric for the objective achievement could be the number of generated biclusters. The main requirement is thus identifying which missing values actually reduce the final number of biclusters if switched to 1. In this way, we only focus on the most "useful" values. Once missing values are imputed through EvaluateMissing, it is only necessary to apply a clustering algorithm over the data sets obtained through the procedure Binarize by using given list of thresholds. Then, the threshold that assures the best result (i.e., the minimum number of biclusters) is chosen. Conversely, in KNN requires to choose the threshold t, and the parameter k should be changed as well. Yet, by changing that parameter k we also need to recompute all the missing data, that is definitely more expensive.

5.2 ABBA*: Missing Values and Outliers Detector

In this section we propose a novel algorithm that is able to evaluate both missing values and outliers: ABBA*. The idea behind this algorithm is the same that is behind ABBA: we leverage the pseudo-bicluster concept in order to assign a relevance to the cells of the matrix M. High relevances for cells containing 0's are typical of missing values, low relevances for cells containing 1's are typical of outliers. ABBA* is listed as Algorithm 5.2, and described in the following. The loop from Line 2 to Line 6 generates a maximal pseudo-

5.2 ABBA*: Missing Values and Outliers Detector

```
 1: procedure EvaluateRelevance(M)
 2:     for all m_lk ∈ M s.t. m_lk = 1 do
 3:         for all m_ij ∈ B_{m_lk} s.t. m_ij = '*' do
 4:             m_ij.count ← m_ij.count + 1
 5:         end for
 6:     end for
 7:     return M
 8: end procedure

 9: procedure FindMissingsAndOutliers(M, t_1, t_2)
10:     Missings ← ∅
11:     Outliers ← ∅
12:     for all m_ij ∈ M s.t. m_ij = '*' do
13:         if m_ij.count > t_1 then
14:             Missings ← Missings ∪ {m_ij}
15:         end if
16:         if m_ij.count < t_2 then
17:             Outliers ← Outliers ∪ {m_ij}
18:         end if
19:     end for
20:     return M, Missings, Outliers
21: end procedure
```

bicluster for each element $m_{ij} = 1$ belonging to the matrix M. The loop from Line 3 to Line 5 increases the relevance of each cell of the maximal pseudo-bicluster just created. Notice that the condition $m_{ij} =$ '$*$' in Line 3 assumes that we are dealing with a flagged matrix. If this is not the case, we can just execute this loop for each $m_{ij} \in \mathrm{B}_{m_{\ell k}}$. At the end of the algorithm, each suspicious value (i.e., elements with $m_{ij} =$ '$*$' in the flagged version, or each m_{ij} in the not-flagged version) will contain a value that corresponds to $\sigma(m_{ij})$ in its data field referred to as 'count'.

After having calculated the relevances through EvaluateRelevance, the procedure FindMissingsAndOutliers can be called. It takes in input the matrix M and two thresholds t_1 and t_2. All the cells containing a value greater than t_1 are highlighted as *missing values*, while all the cells containing values lower than t_2 are highlighted as *outliers*.

5.3 Fast-ABBA*: A Fast Incremental Update Operation

The output of ABBA* is used to evaluate suspicious values in binary matrices. Like many other algorithms with the same aim, a little modification of the original binary matrix invalidates the previous algorithm outcome, requiring a new computation that does not leverage previous results. However, the particular structure of ABBA* allows for an incremental update operation that re-evaluates suspicious data considering partially modified data without the need to recompute all the values. This is particularly useful whenever a reactive or a real time algorithm is needed, or simply when few dataset entries are modified.

For example, let us consider the use of ABBA* in the social network field. Suppose to have a binary matrix describing the existing friendship relationships among users. Here rows and columns represent users, while a 1 indicates that the corresponding two users are listed as friends. The matrix can be quite large, counting millions of users. ABBA* can be used to propose possible new friendships to the users. Note that each user adding or removing friends, contributes to the matrix modification. Therefore, a standard algorithm needs to update its results after each friendship modification, making its use almost impracticable in this setting. However, the update operation of ABBA* introduced in the following allows to efficiently execute the re-imputation of all the suspicious values, while incurring only a computational complexity that is linear with the cardinality of the dataset—in contrast with the quadratic complexity required by recomputing all the values.

The idea behind the update operation is explained in the following. ABBA* is based on the maximal pseudo-biclusters concept, and all the maximal pseudo-biclusters generated by each cell $m \in M$ are used to impute suspicious values. Maximal pseudo-biclusters are generated starting from all the cells filled by 1. We only have two possible modifications of the original binary matrix, that is when some 0's is switched to 1's, and/or some 1's is switched to 0's. Let us denote a modified cell with m_{st}. In the first case (0 to 1), a new maximal pseudo-bicluster has to be considered in the imputation of the relevance, namely the one generated by m_{st}. In the second case (1 to 0), a maximal pseudo-bicluster that has been previously used in the computation (the one generated from m_{st}) must not be considered anymore in the computation of the relevance. By switching the value of a cell, the modification of the area of other maximal pseudo-biclusters has to be considered as well. In fact, by switching a cell from 0 to 1 (or from 1 to 0), some maximal pseudo-biclusters could increase (decrease) their area, then covering other cells. The maximal

pseudo-biclusters that can be influenced by a switch are only those generated by cells that are in the same column or in the same row of the switched cell. Indeed, only the area of those maximal pseudo-biclusters can change. In the case of 0 to 1, the maximal pseudo-bicluster generated by a cell that is in the same row of m_{st} will extend its area by also covering some cells in row s, while the maximal pseudo-bicluster generated by a cell that is in the same column of m_{st} will extend its area by also covering some cells in column t. In the case of 1 to 0, the maximal pseudo-biclusters involved are the same, but they will decrease their area not covering anymore row s or column t.

The previous considerations can be formalized with the following two theorems:

Theorem 5.2 *By switching the value of a cell $m_{st} \in M$ from 0 to 1, the relevance of a cell $m_{ij} \in M$ will increase by the quantity*

$$[m_{ij}^{(1)} \in B(m_{st}^{(1)})] + [j = t] \left(\sum_{m_{sk}^{(0)} \in m_{s*} \setminus \{m_{st}\} : m_{sk}^{(0)} = 1} [m_{ik}^{(0)} = 1] \right) + [i = s] \left(\sum_{m_{\ell t}^{(0)} \in m_{*t} \setminus \{m_{st}\} : m_{\ell t}^{(0)} = 1} [m_{lj}^{(0)} = 1] \right) \quad (5.3)$$

where the exponents (0) and (1) indicate respectively the configuration of the matrix before and after the switch.

PROOF The binary value of a cell $m_{ij} \in M$ before and after switching the value of m_{st}, is indicated respectively with $m_{ij}^{(0)}$, and $m_{ij}^{(1)}$. Thus, the relevance of $m_{ij} \in M$ before switching the cell $m_{st} \in M$ can be written as:

$$\sigma(m_{ij}^{(0)}) = \sum_{m_{\ell k}^{(0)} \in M : m_{\ell k}^{(0)} = 1} [m_{ij}^{(0)} \in B(m_{\ell k}^{(0)})], \quad (5.4)$$

while the relevance of m_{ij} after the switch can be denoted as:

$$\sigma(m_{ij}^{(1)}) = \sum_{m_{\ell k}^{(1)} \in M : m_{\ell k}^{(1)} = 1} [m_{ij}^{(1)} \in B(m_{\ell k}^{(1)})]. \quad (5.5)$$

According to the previous observations, the relevance of $m_{ij}^{(1)}$ can change only when it is involved in some pseudo-bicluster influenced by the value switch of m_{st}. Therefore, we only focus on these kinds of pseudo-biclusters. In particular, it can be observed that such pseudo-biclusters are only those generated by

m_{st}, that is by cells that equal 1 and are in the row s, and, by cells that equal 1 and are in the column t. Thus, the previous equation can be written also as:

$$\sigma(m_{ij}^{(1)}) = [m_{ij}^{(1)} \in B(m_{st}^{(1)})] + \sum_{m_{\ell k}^{(1)} \in (m_{s*} \cup m_{*t}) \setminus \{m_{st}\} : m_{\ell k}^{(1)} = 1} [m_{ij}^{(1)} \in B(m_{\ell k}^{(1)})] + \sum_{m_{\ell k}^{(1)} \in M \setminus (m_{s*} \cup m_{*t} \cup \{m_{st}\}) : m_{\ell k}^{(1)} = 1} [m_{ij}^{(1)} \in B(m_{\ell k}^{(1)})] \quad (5.6)$$

The first term corresponds to the contribution of the new maximal pseudo-bicluster introduced, the second one is the contribution of the maximal pseudo-biclusters that could have been changed after the switch of m_{st}, while the last one is the contribution of the maximal pseudo-biclusters that remained unchanged. Indeed, only the maximal pseudo-biclusters generated by a cell that is in the same row, or in the same column of m_{st} are influenced by the switch.

The second term of Equation 5.6 can be further decomposed by considering that the modified maximal pseudo-biclusters extended their area by including at most row s or column t, namely:

$$\sum_{m_{\ell k}^{(1)} \in (m_{s*} \cup m_{*t}) \setminus \{m_{st}\} : m_{\ell k}^{(1)} = 1} [m_{ij}^{(1)} \in B(m_{\ell k}^{(1)})] = \sum_{m_{\ell k}^{(0)} \in (m_{s*} \cup m_{*t}) \setminus \{m_{st}\} : m_{\ell k}^{(0)} = 1} [m_{ij}^{(0)} \in B(m_{\ell k}^{(0)})] + [j = t] \left(\sum_{m_{sk}^{(0)} \in m_{s*} \setminus \{m_{st}\} : m_{sk}^{(0)} = 1} [m_{ik}^{(0)} = 1] \right) + [i = s] \left(\sum_{m_{\ell t}^{(0)} \in m_{*t} \setminus \{m_{st}\} : m_{\ell t}^{(0)} = 1} [m_{lj}^{(0)} = 1] \right) \quad (5.7)$$

In Equation 5.7, the first term represents the contribution of the old maximal pseudo-biclusters, that is those built by considering $m_{st} = 0$, whereas the other two terms only consider the extended portion of the maximal pseudo-biclusters generated by the elements in row s or column t. Indeed, by considering the bicluster generated from a cell m_{sk} that is in the same row of

m_{st}, when m_{st} is switched, it will extend its area to also contain column t. A generic cell m_{ij} will be contained in the extended portion of this maximal pseudo-bicluster only if $j = t$ and the cell m_{ik} is set to 1. In the same way, by considering the bicluster generated from a cell m_{lt} that is in the same column of m_{st}, it will extend his area to also contain row s. A generic cell m_{ij} will be contained in the extended portion of this maximal pseudo-bicluster only if $i = s$ and the cell m_{lj} is set to 1.

Putting all together, the relevance of a cell $m_{ij} \in M$ after the switch of m_{st} can be computed as:

$$\sigma(m_{ij}^{(1)}) = [m_{ij}^{(1)} \in B(m_{st}^{(1)})] + \sum_{m_{\ell k}^{(0)} \in (m_{s*} \cup m_{*t}) \{\setminus m_{st}\} : m_{\ell k}^{(0)} = 1} [m_{ij}^{(0)} \in B(m_{\ell k}^{(0)})]$$

$$+ [j = t] \left(\sum_{m_{sk}^{(0)} \in m_{s*} \setminus \{m_{st}\} : m_{sk}^{(0)} = 1} [m_{ik}^{(0)} = 1] \right)$$

$$+ [i = s] \left(\sum_{m_{\ell t}^{(0)} \in m_{*t} \setminus \{m_{st}\} : m_{\ell t}^{(0)} = 1} [m_{lj}^{(0)} = 1] \right)$$

$$+ \sum_{m_{\ell k}^{(1)} \in M \setminus (m_{s*} \cup m_{*t} \cup \{m_{st}\}) : m_{\ell k}^{(1)} = 1} [m_{ij}^{(1)} \in B(m_{\ell k}^{(1)})].$$

Since we are interested in the difference between $\sigma(m_{ij}^{(1)})$ and $\sigma(m_{ij}^{(0)})$, we have that:

$$\sigma(m_{ij}^{(1)}) - \sigma(m_{ij}^{(0)}) = [m_{ij}^{(1)} \in B(m_{st}^{(1)})] + \sum_{m_{\ell k}^{(0)} \in (m_{s*} \cup m_{*t}) \setminus \{m_{st}\} : m_{\ell k}^{(0)} = 1} [m_{ij}^{(0)} \in B(m_{\ell k}^{(0)})]$$

$$+ [j = t] \left(\sum_{m_{sk}^{(0)} \in m_{s*} \setminus \{m_{st}\} : m_{sk}^{(0)} = 1} [m_{ik}^{(0)} = 1] \right) + [i = s] \left(\sum_{m_{\ell t}^{(0)} \in m_{*t} \setminus \{m_{st}\} : m_{\ell t}^{(0)} = 1} [m_{lj}^{(0)} = 1] \right)$$

$$+ \sum_{m_{\ell k}^{(1)} \in M \setminus (m_{s*} \cup m_{*t} \cup \{m_{st}\}) : m_{\ell k}^{(1)} = 1} [m_{ij}^{(1)} \in B(m_{\ell k}^{(1)})] - \sum_{m_{\ell k}^{(0)} \in M : m_{\ell k}^{(0)} = 1} [m_{ij}^{(0)} \in B(m_{\ell k}^{(0)})]. \tag{5.8}$$

The fourth term of the sum considers all the cells m_{lk} that are not in the same row, or in the same column of m_{st}. Since the maximum pseudo-bicluster generated by these elements does not change after m_{st} has been switched, it

can be written also as

$$\sum_{m_{\ell k}^{(0)} \in M \setminus (m_{s*} \cup m_{*t} \cup \{m_{st}\}) : m_{\ell k}^{(0)} = 1} [m_{ij}^{(0)} \in B(m_{\ell k}^{(0)})].$$

By simplifying Equation 5.8, we have that:

$$\begin{aligned} \sigma(m_{ij}^{(1)}) - \sigma(m_{ij}^{(0)}) = [m_{ij}^{(1)} \in B(m_{st}^{(1)})] \\ + [j = t] \left(\sum_{m_{sk}^{(0)} \in m_{s*} \setminus \{m_{st}\} : m_{sk}^{(0)} = 1} [m_{ik}^{(0)} = 1] \right) \\ + [i = s] \left(\sum_{m_{\ell t}^{(0)} \in m_{*t} \setminus \{m_{st}\} : m_{\ell t}^{(0)} = 1} [m_{lj}^{(0)} = 1] \right). \end{aligned} \quad (5.9)$$

The previous equation corresponds to the relevance increment after m_{st} have been switched from 0 to 1, thus completing the proof. ■

The following theorem can be proved as well:

Theorem 5.3 *By switching the value of a cell $m_{st} \in M$ from 1 to 0, the relevance of a cell $m_{ij} \in M$ will decrease by the quantity*

$$\begin{aligned} [m_{ij}^{(0)} \in B(m_{st}^{(0)})] + [j = t] \left(\sum_{m_{sk}^{(0)} \in m_{s*} \setminus \{m_{st}\} : m_{sk}^{(0)} = 1} [m_{ik}^{(0)} = 1] \right) + \\ [i = s] \left(\sum_{m_{\ell t}^{(0)} \in m_{*t} \setminus \{m_{st}\} : m_{\ell t}^{(0)} = 1} [m_{lj}^{(0)} = 1] \right) \end{aligned}$$

PROOF The proof is a rewriting of the proof of Theorem 5.2. ■

By leveraging theorems 5.2 and 5.3, we are able to exactly compute modifications of the relevances when a change of the matrix occurs. The update function listed in Algorithm 5.3 is a direct application of these two theorems. It is described in the following. From Line 2 to Line 6, it is checked whether m_{st} is switching from 0 to 1, or from 1 to 0. Depending on the kind of switch executed, the relevance will be increased or decreased. The rest of the algorithm is composed by three loops, each one updating the cells involved in the three terms of Equation 5.9. The loop from Line 7 to Line 9 increases (or decreases)

5.3 Fast-ABBA*: Incremental ABBA

```
1:  procedure Update(m_st)
2:      if m_st switches from 0 to 1 then
3:          δ ← +1
4:      else {m_st switches from 1 to 0}
5:          δ ← −1
6:      end if
7:      for all m_ij ∈ B(m_st) s.t. m_ij = '*' do
8:          m_ij.count ← m_ij.count + δ
9:      end for
10:     for all m_ij ∈ m_*t s.t. m_ij = '*' do
11:         for all m_sk ∈ m_s* s.t. m_sk = 1 do
12:             if m_ik = 1 then
13:                 m_ij.count ← m_ij.count + δ
14:             end if
15:         end for
16:     end for
17:     for all m_ij ∈ m_s* s.t. m_ij = '*' do
18:         for all m_ℓt ∈ m_*t s.t. m_ℓt = 1 do
19:             if m_lj = 1 then
20:                 m_ij.count ← m_ij.count + δ
21:             end if
22:         end for
23:     end for
24:     return M
25: end procedure
```

the counter of all the cells that are contained in the maximal pseudo-bicluster generated by m_{st}. It corresponds to the first term of Equation 5.9, executed for all the cells that can be modified by this term. The loop from Line 10 to Line 16 is only executed for the cells that are in the same column of m_{st}. These cells are the only ones involved in the second term of Equation 5.9, and each one is increased (or decreased) by executing the internal loop. In particular, the internal loop increases/decreases the relevance of a cell if the condition is satisfied. In the same way, the third loop is executed for all the cells that are in the same row of m_{st}. Each one is increased/decreased by the internal loop if the condition is satisfied. This third loop corresponds to the third term of Equation 5.9.

Since Algorithm 5.3 is derived from the above equations and theorems, its

correctness is assured. Moreover, since it is only composed by finite loops, than its completeness is also assured. As for the computational complexity, it is $O(n \times m)$. Indeed, the first loop is executed by considering all the cells belonging to the maximal pseudo-bicluster generated by the modified cell, that are at most $n \times m$. The second loop is executed over all the cells in the row s (that are at most m), and for each one the internal loop is executed over all the cells belonging to the column t (that are at most n), giving a complexity of $O(n \times m)$. Also the third loop can be executed in $O(n \times m)$. Indeed, it is executed over all the cells in the column t (that are at most n), and for each one the internal loop is executed over all the cells belonging to the row s (that are at most m). All the other operations can be executed in constant time. In particular, it can be used a hash table that gives all columns with 1's for a given row, and all rows with 1's for a given column—that is, the same hash table defined for ABBA*. Thus, the total computation complexity is equal to $O(n \times m)$.

5.4 Experimental Results

In this section, we test the quality of the results produced by our algorithms and discuss the achieved performances. First, we report on the results of ABBA* on synthetic datasets, considering both missing values and outliers. Then, we test the performances of both ABBA* and Fast-ABBA* on real data, showing that the latter is much faster than the former. Indeed, dealing with real datasets the incurred computational overhead is much lower than in the worst case captured by the big O notation. Further, we directly compare with KNN, showing that ABBA* is much more reliable in all the cases. The hardware used for the experiments was: a notebook with an Intel Pentium M processor at 1.86 GHz, and 2 GB RAM at 782 MHz, the algorithms have been implemented in Java, and the simulations have been executed in a GNU/Linux operating system (Ubuntu 10.04).

To assess the quality of the achieved results, we introduce a measure based on the Jaccard's coefficient that has been already used in [33] to compare the similarity of two matrices. Let n_{ij} be the number of entries on which two matrices M and R have values i and j, respectively. Thus, n_{11} is the number of detected mates, n_{00} is the number of non-mates, while n_{10} and n_{01} count the disagreements between the true and suggested solution. The Jaccard's coefficient is defined as $\frac{n_{11}}{n_{11}+n_{10}+n_{01}}$. It represents the proportion of the correctly identified mates over the sum of the correctly identified mates plus the total number of disagreements. Hence, the Jaccard's coefficient should score one when all the suspicious values are correctly identified. Conversely, the

more this index is closer to zero, the less the two matrices can be considered similar. Thus, we use the Jaccard coefficient to evaluate the similarity between imputed and original matrices, allowing us to compare the performances achieved by different algorithms over the same dataset.

In the following simulations, when dealing with synthetic data, we first generate a matrix M. Then, uniformly at random, we flip a fraction of the elements of M, generating a new matrix M'. In turn, M' is given in input to our algorithm, which generates a matrix R. Using the Jaccard's coefficient to measure the similarity of R (the rebuilt matrix) and M (the original matrix), we capture the quality of the rebuilt matrix, that is, how close it is to the original one (M).

Testing Missing Values Imputation

To implement our experiments on synthetic datasets, we generated sample matrices composed by 600 rows and 100 columns. Each matrix has been generated with the procedure described in the following. First, 20 subsets of rows and columns have been randomly chosen. Each subset of rows, and each subset of columns, counts a number of elements that are proportional to $\texttt{MaxRows} \times x^y$ and $\texttt{MaxCols} \times x^y$, where x is a random number uniformly chosen in the interval [0,1], while y, `MaxRows` and `MaxCols` are integer variables. The elements of the matrix M that belongs to one of such subsets are set to 1, while the other ones are set to 0. The exponent y allows to change the number of small and large patterns created. Note that $y = 1$ corresponds to the uniform distribution. By increasing y we are able to generate a higher number of patterns of small dimension, and some of high dimension. The noise has been introduced randomly selecting cells equals to 1, and setting them to 0. We indicate with `MissingRate` the fraction of cells of M that we flipped using this procedure.

In order to show the advantages provided by our approach, we compared our algorithm with KNN. Table 5.2 shows a summary of the results for several sample matrices. To be fair, we chose to report several results for KNN, obtained using different k. However, we want to highlight that in a real implementation selecting the best k corresponds to an extra overhead that is not needed in our algorithm. Each row of the table corresponds to a sample matrix generated as above using the indicated parameters, and $y = 5$. To have more reliable results, the results correspond to the average of 100 simulations, where the threshold t assures that the best results is reported. The best result for each configuration in a row is highlighted in gray. It can be seen that our algorithm performs better than KNN (for any k used). The difference is more noticeable when the matrix is sparse (lower values for `MaxRows` and `MaxCols`),

Table 5.1 Outliers Imputation with ABBA*. The Jaccard coefficient between the original and the imputed matrix is reported in ABBA* column.

MaxRows	MaxCols	OutliersRate	**ABBA***
20	20	0.1	0.980
		0.2	0.967
		0.3	0.958
		0.4	0.947
		0.5	0.945
30	30	0.1	0.978
		0.2	0.968
		0.3	0.959
		0.4	0.953
		0.5	0.951
60	60	0.1	0.978
		0.2	0.967
		0.3	0.957
		0.4	0.951
		0.5	0.949

and above all when the number of missing values is high.

Testing Outliers Imputation

As for the outliers detection, we conducted an experiment similar to the previous one. In this case, the original matrix is composed by 1,000 rows and 1,000 columns. Table 5.1 reports on the results achieved. Data has been generated using the previously described process. However, the noise has been introduced selecting a fraction of random cells of M equal to 0 (indicated as `OutliersRate` in the table), and setting them to 1. It can be seen that, after executing the outliers imputation using ABBA*, in all the cases the similarity of the imputed matrix and the original one is above the 94%. Notice that this happens also when the noise introduced is extremely high. Even with few patterns of maximum 20×20 cells it is able to distinguish them from the noise. This shows the strong reliability of ABBA* to high rates of random noise.

Table 5.2 A comparison between KNN and ABBA*. Values in KNN and ABBA* columns report the Jaccard coefficient between the original and the imputed matrix.

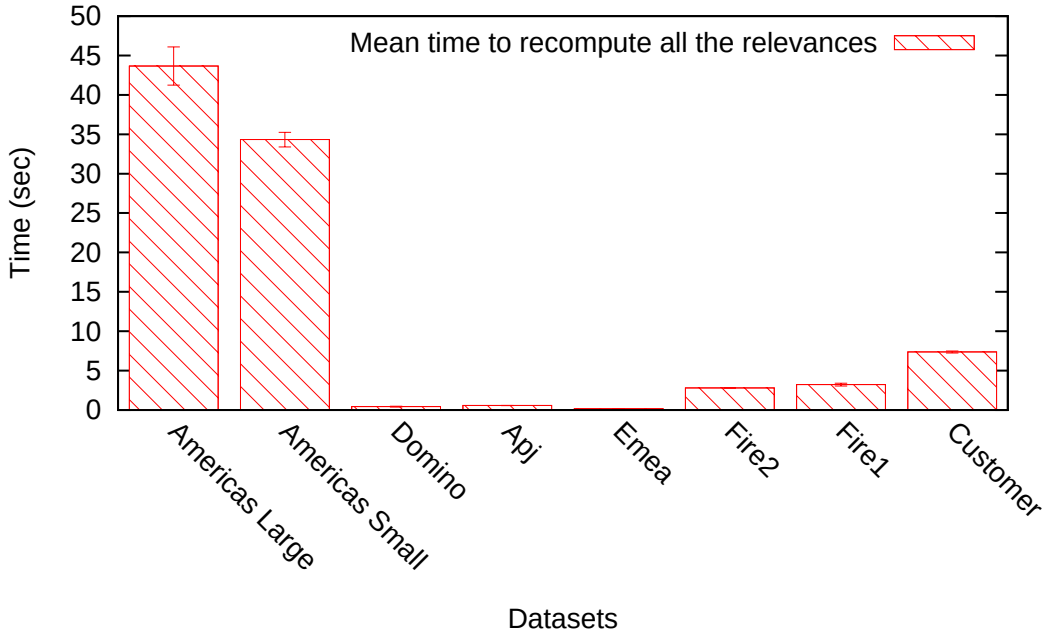
MaxRows	MaxCols	MissingRate	KNN(4)	KNN(16)	KNN(32)	KNN(64)	ABBA*
20	4	0.1	0.910	0.916	0.918	0.919	0.935
		0.2	0.825	0.837	0.844	0.845	0.860
		0.3	0.706	0.717	0.717	0.718	0.777
		0.4	0.543	0.550	0.555	0.556	0.676
		0.5	0.482	0.491	0.492	0.494	0.562
30	6	0.1	0.883	0.893	0.898	0.899	0.945
		0.2	0.783	0.785	0.792	0.798	0.884
		0.3	0.706	0.716	0.717	0.721	0.811
		0.4	0.615	0.621	0.623	0.623	0.801
		0.5	0.500	0.510	0.511	0.511	0.611
60	8	0.1	0.909	0.923	0.926	0.927	0.951
		0.2	0.819	0.834	0.838	0.842	0.890
		0.3	0.708	0.721	0.725	0.725	0.811
		0.4	0.613	0.625	0.625	0.625	0.731
		0.5	0.501	0.508	0.514	0.514	0.648
90	12	0.1	0.939	0.943	0.943	0.943	0.959
		0.2	0.850	0.867	0.870	0.870	0.910
		0.3	0.746	0.765	0.769	0.774	0.838
		0.4	0.616	0.635	0.645	0.651	0.767
		0.5	0.504	0.517	0.517	0.529	0.682
120	16	0.1	0.935	0.938	0.942	0.945	0.949
		0.2	0.826	0.831	0.831	0.831	0.966
		0.3	0.769	0.789	0.796	0.799	0.925
		0.4	0.615	0.637	0.641	0.644	0.855
		0.5	0.555	0.584	0.592	0.592	0.783

Testing on Real Data

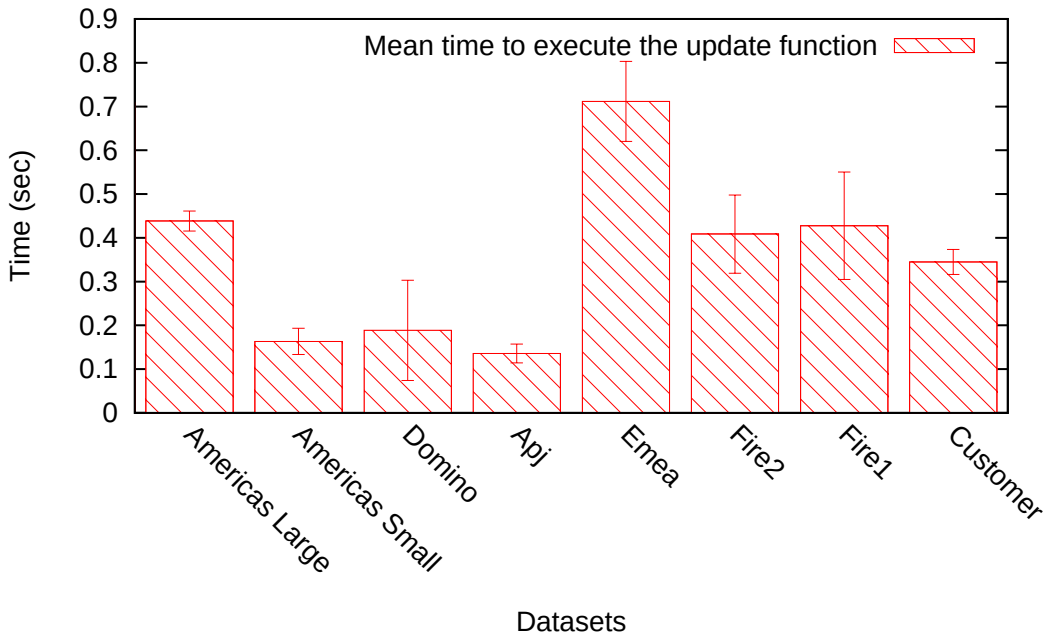
We tested our algorithm on real data as well. In particular: first, we compared the mean time to recompute all the relevances from the beginning, with the mean time required by the update function introduced in Section 5.3, then we compared our algorithm with KNN. Since we ran the experiments in a multitasking environment, the reported results are an upper bound on the real computation time. However, we want to use them to show the performances of the introduced update function. Indeed, even if its computational complexity is linear with the dataset size, the worst case rarely happens in a real scenario. Experiments conducted over many real datasets show that the actual performances are even better than the expected one.

Figure 5.1 reports on the comparison of the computational time needed to update the suspicious values when a modification of the source data occurs. The reported times correspond to the mean of 1000 simulations, while the error bars correspond to their standard deviation. The analysis has been conducted over 8 datasets publicly available concerning several real-world access-control scenarios [59]. In Table 5.3 the main properties of these datasets are summarized. Figure 5.1(a) reports the mean times needed to recompute all the values from scratch. It can be seen that when the dataset is quite large, as in the case of "America Large" and "America Small", the time needed to recompute all the values is around 45 seconds in the first case, and 35 in the second one. However, for all the remaining datasets the mean requested time is lower than 10 seconds. Comparing the properties of the datasets with the computational time needed to find missing values and outliers, it turns out that the time to execute this computation depends on the dataset size. Instead, Figure 5.1(b) shows the time needed to execute the update function introduced in Section 5.3. It can be noticed that in all the cases the mean time is lower than 1 second, and above all not affected by the size of the dataset. Indeed, even if the computational complexity of the update function is $O(nm)$, where n is the number of rows and m is the number of columns, this is only the worst case, that seems to be really unlikely in real world datasets. Considering that a dataset could be even larger than "America Large", it appears noticeable that the introduction of the update function allows the use of ABBA* also for larger datasets, and where the real-time re-computation of the suspicious values is needed.

In addition to the time performances discussed above, we conducted other tests to evaluate the performances of our algorithms against our competitors. Since the experiments that we are going to introduce are computationally intensive, we selected two of the height datasets previously introduced, that is "Domino" and "Emea". We use the following procedure to generate datasets



(a) Time to recompute all the relevances from scratch



(b) Time to execute the update function

Figure 5.1 Comparing the update function with the recomputing time of all the relevances.

Table 5.3 Main properties of the real world datasets used in our tests.

Dataset	Rows	Columns	$\|Dataset\|$
America Small	3,477	1,587	105,205
America Large	3,485	10,127	185,294
Domino	79	231	730
Apj	2,044	1,164	6,841
Emea	35	3,046	7,220
Fire1	365	709	31,951
Fire2	325	590	36,428
Customer	10,021	277	45,427

based on "Domino" and "Emea", but containing missing values and outliers:

1. We add to the existing dataset a number of rows equal to the 33% of the original number of rows. These rows are copies of existing rows randomly selected;

2. We modify the new rows adding missing values, or outliers (depending on the case that we are going to analyze). In the first case, we switch each existing 1 to 0 with a probability of the 5%, in the second case we switch each 0 to 1 with the same probability.

By varying the threshold from 0 to 1, we generate several ROC graphs for both our algorithm and the KNN based algorithms. A ROC graph shows the false positive rate (also called specificity, that is the ratio between true negative and the sum of true negative and false positive) on the X axis, and the true positive rate (also called sensitivity, that is the ratio between true positive and the sum of true positive and false negative) on the Y axis. These graphs are used to estimate the performances of classifiers. Indeed, our test can be read as a classification problem: we are interested in evaluate missing values and outliers correctly classified. In a ROC graph, the point (0,1) is reached by a perfect classifier: it classifies all positive cases and negative cases correctly. In the following, each point that is shown in the graphs is the mean of 10 runs. For each run we generated a new dataset following the procedure described above, and we evaluated missing values and outliers varying the threshold from 0 to 1 with steps of 0.001.

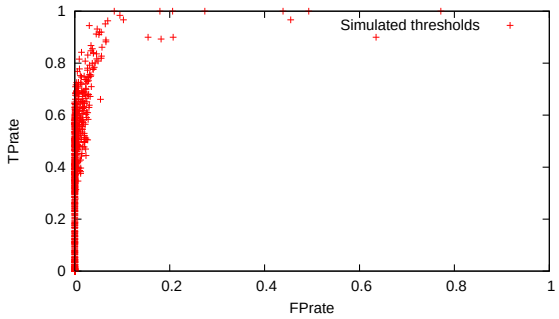
Figure 5.2 shows the results related to the missing values. To be fair, we executed the experiments using several values for the parameter k used in

KNN, and we find out that the best results are achieved for $k = 8$ in both the datasets. However, we want to stress the fact that our algorithm does not need to set up such a parameter, resulting in a more straightforward approach. The figure shows the results achieved by our algorithm and for KNN with $k = 8$ and $k = 32$, that we indicate with $KNN(8)$ and $KNN(32)$. In the case of "Domino", our algorithm performs better than $KNN(32)$, indeed comparing Figure 5.2(a) and Figure 5.2(c), it can be noticed that generally the points are closer to the point (0,1) in the first figure. As for $KNN(8)$, it seems to perform slightly better than our algorithm. This is due to the range used for the threshold, that is $(0, 1)$. Indeed, our algorithm is more sensitive to threshold variation than KNN with small k. Potentially, when a threshold equal to 0 is used, all the "empty" cells can be recognized as missing values with a consequent grown of false positives. The choice of the threshold is therefore quite important, and generally it is preferable a conservative approach: selecting a higher threshold the number of false positive is generally reduced. The result related to the second dataset analyzed ("Emea") are reported in figures 5.2(b), 5.2(d), 5.2(f). In this case our algorithm performs better than KNN.

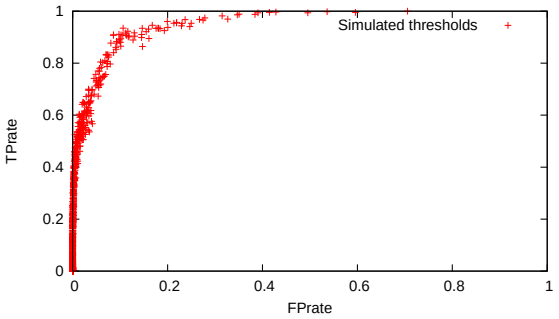
As for the outliers detection, we compared our algorithm against the well known and largely used KNN based algorithm proposed in [4]. A row of the binary matrix represents a point in the space. For each row, the algorithm finds the m-th neighbor and calculates the distance D_m. If this distance is less than a threshold d then the row lies in a sufficiently dense region of the data distribution and is classified as normal. Otherwise the row is classified as outlier. Note that they classify rows as outliers, instead our algorithm is a more fine-grained approach that allows to classify as outliers even individual cells of a row. Figure 5.3 summarizes the results achieved. In all the six sub-figures, the points are quite scattered because we are considering all the possible values for the threshold, in spite of selecting a reasonable one. However, this choice is useful to highlight that it does not exist any threshold t used by the KNN based algorithm that can overcome the performances achieved by our algorithm. Indeed, using $KNN(8)$ or $KNN(32)$, the true positive rate is always less than 0.4 when the false positive rate is less than 0.3, while using our approach we reach a false positive rate higher than 0.6 in both the tested datasets.

A Case Study on Access Control Information

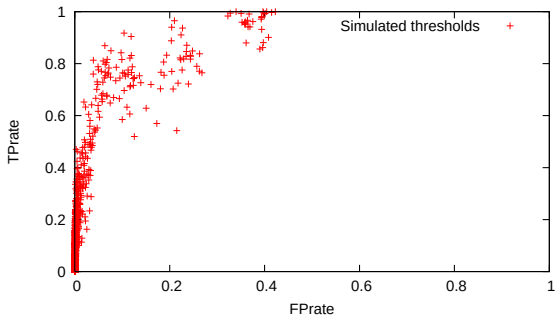
To demonstrate the viability of the proposed framework, we carried out a case study on access control information (user-permission relationships) provided by a large private company. To simplify the analysis, we decomposed the



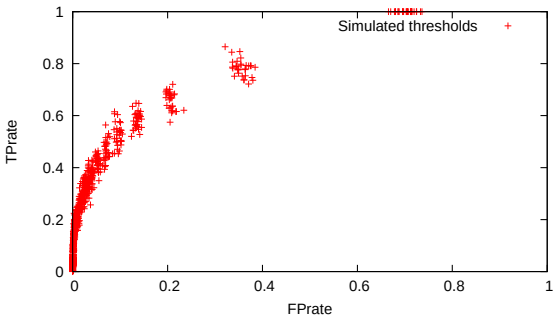
(a) Domino, our approach



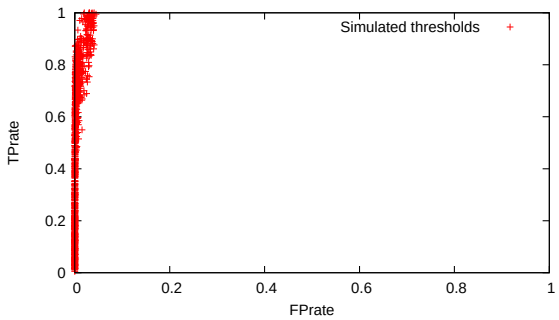
(b) Emea, our approach



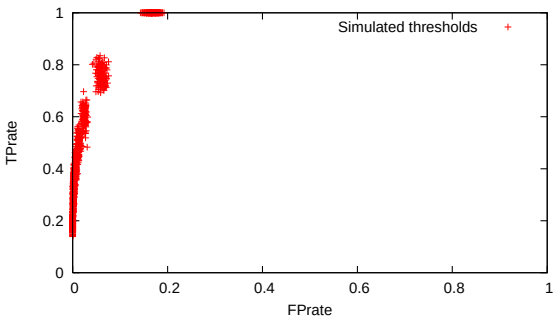
(c) Domino, KNN(32)



(d) Emea, KNN(32)

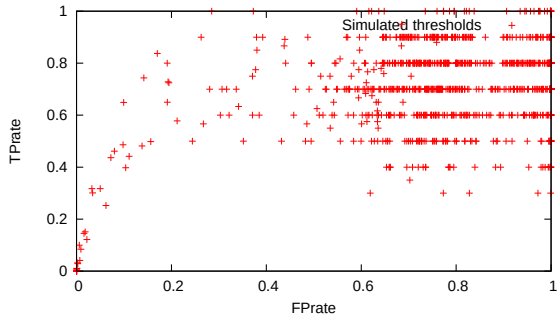


(e) Domino, KNN(8)

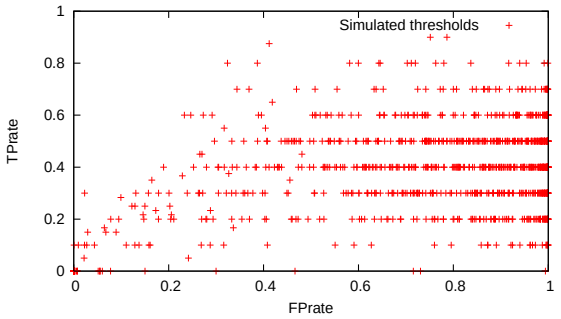


(f) Emea, KNN(8)

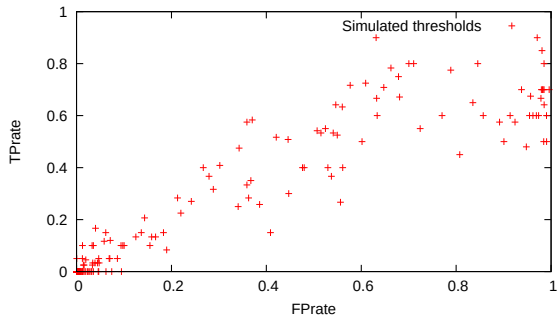
Figure 5.2 Missing values performances on real datasets: ROC Graph



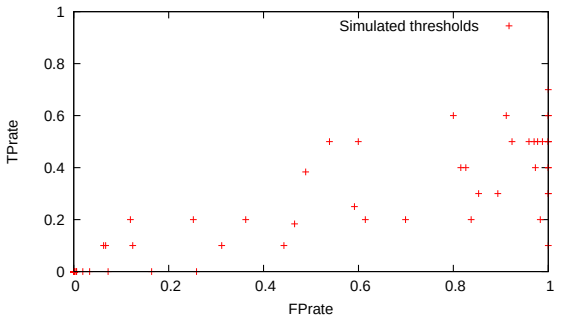
(a) Domino, our approach



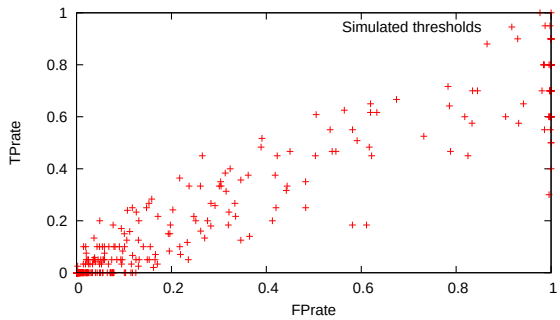
(b) Emea, our approach



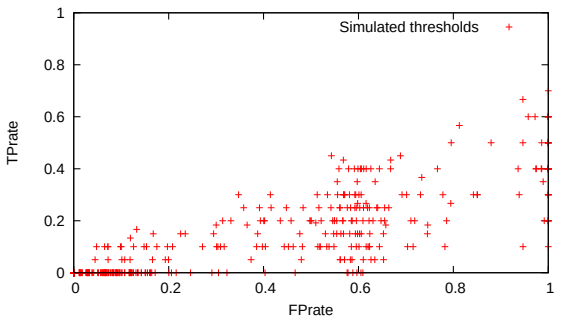
(c) Domino, KNN with k=32



(d) Emea, KNN with k=32

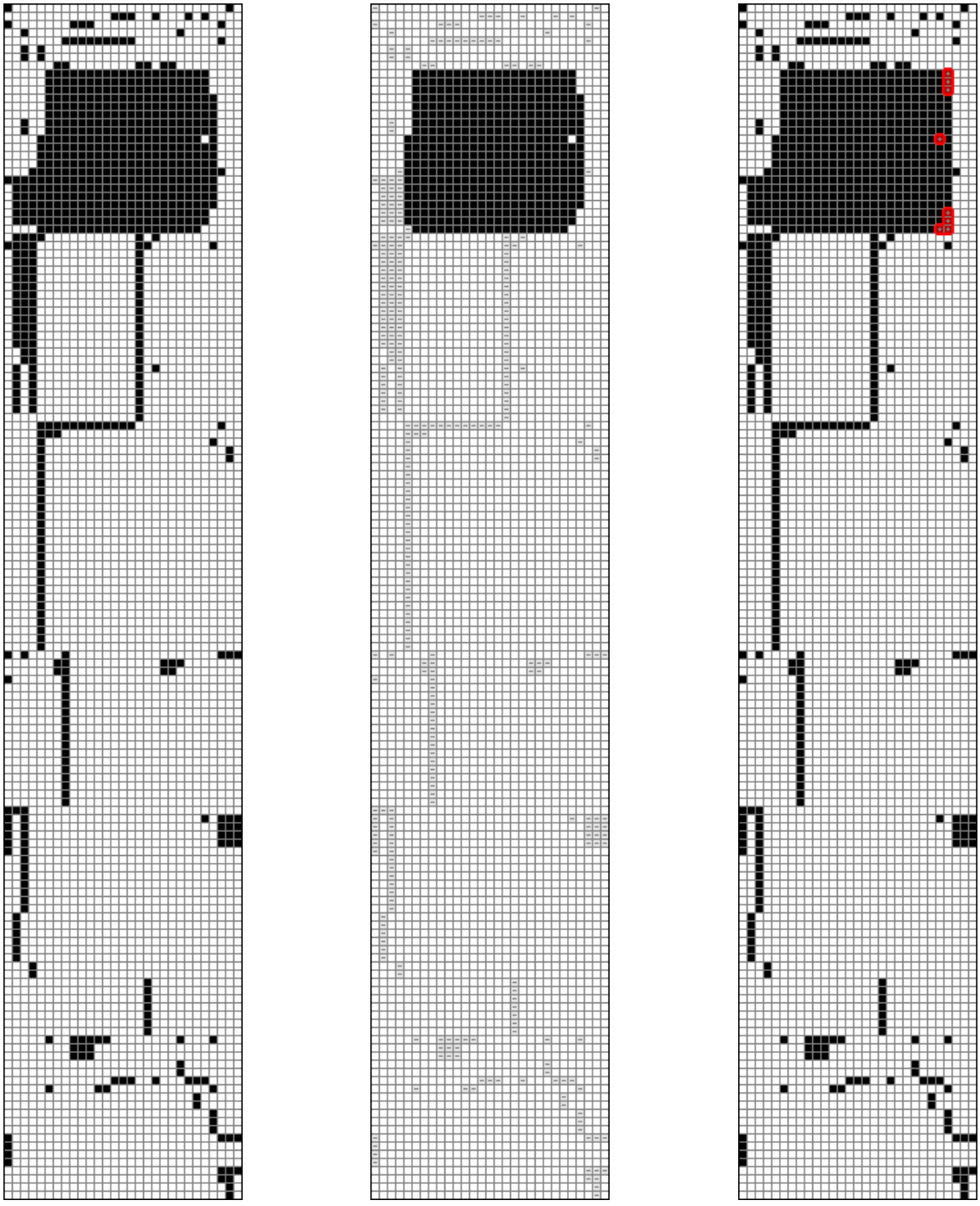


(e) Domino, KNN with k=8



(f) Emea, KNN with k=8

Figure 5.3 Outliers detection performances on real datasets: ROC graph



[tb]

(a) Original

(b) Outliers, $t = 0.4$

(c) Missings, $t = 0.6$

Figure 5.4 Outliers and Missing Values identification in a real dataset

problem into smaller sub-problems, each one represented by the users (and the corresponding permissions) belonging to a representative branch of the organization unit chart. We will only show the outcome of a representative organization unit which numbers 29 users, 146 permissions, and a total of 740 assignments. We will discuss the application of the proposed approach for the identification of missing values and outliers through threshold tuning.

Figure 5.4(a) depicts a binary matrix representing the user-permission relationships involved with the analyzed organization unit. Columns and rows correspond, respectively, to users and permissions, and each cell is filled in black when a certain user has a certain permission granted. We used the algorithm ABBA* to identify the relevance for all the cell of the matrix. To simplify the exposition, we will analyze missing values and outliers separately. Figure 5.4(b) highlights only the outliers proposed by the algorithm when setting up a threshold $t = 0.4$, namely all the black cells that have a relevance value less than 0.4 and thus likely represent exceptional assignments. In the picture, outliers are those represented by light-gray cells with a small "$-$" within them. In this case, IT administrators confirmed the outlier nature of the assignment, that is most of the highlighted assignments were required to access particular exceptional utilities. Figure 5.4(c) highlights only the missing values proposed by the algorithm when setting up a threshold $t = 0.6$, namely all the white cells that have a relevance value greater than 0.6 and thus likely represent missing assignments. In the picture, missing values are those represented by dark-gray cells with a small "$+$" within them and highlighted by a surrounding red line. Even in this case, IT administrators verified that many of the proposed missing values were actually missing user-permission relationships that could be assigned to users without violating the least privilege principle.

6
Visual Role Mining

Visual representations of roles can actually amplify cognition, leading to optimal analysis results [31, 48]. In this chapter we introduce a new approach, referred to as *visual role mining*. We offer a graphical way to effectively *navigate* the result of *any* existing role mining algorithm, showing at glance what it would take a lot of data to expound. Moreover, we allow to *visually identify meaningful roles* within access control data without resorting to traditional role mining tools. Visualization of the user-permission assignments is performed in such a way to isolate the *noise*, allowing role engineers to focus on relevant patterns, leveraging their cognition capabilities. Further, *correlations* among roles are shown as overlapping patterns, hence providing an intuitive way to discover and utilize these relations.

Even though visual approaches sometimes raise some skepticism, they are generally considered to be highly beneficial when used to gain an overview of the underlying dataset. In fact, this chapter shows that a proper representation of user-permission assignments allows role designers to gain insight, draw conclusions, and design meaningful roles from both IT and business perspectives.

6.1 Role Visualization Problem

This section addresses the following problem: given a set of *already* discovered roles of interest, we want to identify the best graphical representation for them. In particular, we want the representation for user-permission assignments that allows for both an intuitive role validation and a visual identification of the relationships among roles. We will show that roles are easier to recognize than describe via a binary matrix representation. The proposed representation can answer questions that classical statistical or mining approaches

cannot (easily) provide. Represented roles can be the outcome of any role engineering process, as well as roles already in place in a RBAC system. Hence, making such a tool an ideal companion for any existing role mining algorithm.

The role mining objective is to analyze access control data in order to elicit a set of *meaningful* roles that *simplify* RBAC management [6, 9, 36]. To this aim, various business information can be analyzed [6, 9], but user-permission assignments are the minimal data-set required. A natural representation for this information is the *binary matrix*, where rows and columns correspond to users and permissions, and each cell is "on" when a certain user has a certain permission granted.

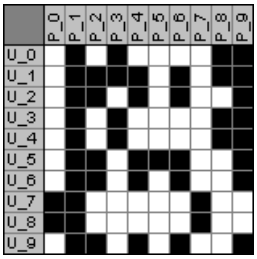
Figure 6.1(a) shows a possible set of user-permission assignments. It is quite clear that it is impossible to analyze such a set without resorting to a more intuitive representation. By reading data in the same order as presented in Figure 6.1(a) we obtain the matrix depicted in Figure 6.1(b). Though this representation is still confusing, it is now possible to observe some patterns. For example, all users possess the permission p_1. Hence, p_1 is likely involved in "base" authorizations to be granted, for example, to new users which join the organization. Practically, we have looked for and found out consecutive cells that are "on". These patterns are usually referred to as *tiles* [38]. Figure 6.1(c) demonstrates that it could be easier to find more patterns if *users and permissions were reordered*. Given the roles listed in Figure 6.1(d), they can be identified more easily in Figure 6.1(c) than in Figure 6.1(b). In particular, Figure 6.1(e) highlights these roles in cyan.

Several considerations can be made from the previous example. First, we can easily deduce all the roles listed in Figure 6.1(d) by only inspecting Figure 6.1(c), namely *without resorting to any role mining algorithm*. Figure 6.1(c) is definitely more communicative: for instance, it is evident that p_1 may be assigned to roles r_2, r_3, r_4, thus making r_1 no longer necessary. Alternatively, if p_1 represents a permission that should *always* be granted to all users, keeping r_1 may be more advantageous. This kind of considerations require additional knowledge that might be hard to translate into structured data. Putting humans in the loop allows for a better correlation of business requirements with IT-related access control data.

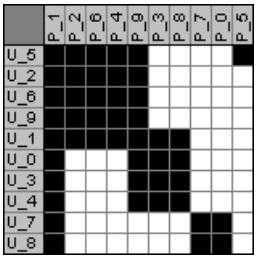
Second, a *visual representation can highlight potential exceptions within data in an effective manner*. For example, the user u_5 is the only one that has permission p_5 granted. This finding warns about a potentially wrong assignment due to causes such as privilege accumulation or illicit authorization. One could observe that this kind of analysis can be performed even without graphically representing user-permission assignments by adopting approximate mining algorithms [42]. However, most algorithms can lead to several false-positive exceptions, degrading the output quality of the automatic analysis.

User-permission Assignments
$\{\langle u_0,p_1\rangle, \langle u_0,p_3\rangle, \langle u_0,p_8\rangle, \langle u_0,p_9\rangle, \langle u_1,p_1\rangle, \langle u_1,p_2\rangle, \langle u_1,p_3\rangle, \langle u_1,p_4\rangle,$ $\langle u_1,p_6\rangle, \langle u_1,p_8\rangle, \langle u_1,p_9\rangle, \langle u_2,p_1\rangle, \langle u_2,p_2\rangle, \langle u_2,p_4\rangle, \langle u_2,p_6\rangle, \langle u_2,p_9\rangle,$ $\langle u_3,p_1\rangle, \langle u_3,p_3\rangle, \langle u_3,p_8\rangle, \langle u_3,p_9\rangle, \langle u_4,p_1\rangle, \langle u_4,p_3\rangle, \langle u_4,p_8\rangle, \langle u_4,p_9\rangle,$ $\langle u_5,p_1\rangle, \langle u_5,p_2\rangle, \langle u_5,p_4\rangle, \langle u_5,p_5\rangle, \langle u_5,p_6\rangle, \langle u_5,p_9\rangle, \langle u_6,p_1\rangle, \langle u_6,p_2\rangle,$ $\langle u_6,p_4\rangle, \langle u_6,p_6\rangle, \langle u_6,p_9\rangle, \langle u_7,p_0\rangle, \langle u_7,p_1\rangle, \langle u_7,p_7\rangle, \langle u_8,p_0\rangle, \langle u_8,p_1\rangle,$ $\langle u_8,p_7\rangle, \langle u_9,p_1\rangle, \langle u_9,p_2\rangle, \langle u_9,p_4\rangle, \langle u_9,p_6\rangle, \langle u_9,p_9\rangle\}$

(a) Input data



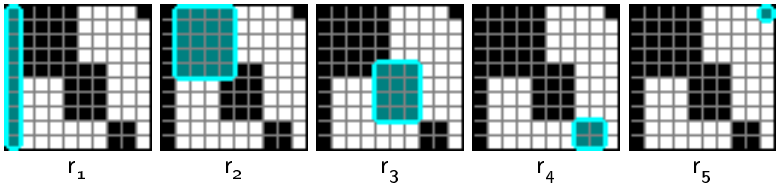
(b) Unsorted matrix



(c) Sorted matrix

Role	*Permissions*	*Users*
r_1	$\{p_1\}$	$\{u_0,u_1,u_2,u_3,u_4,u_5,u_6,u_7,u_8,u_9\}$
r_2	$\{p_2,p_4,p_6,p_9\}$	$\{u_1,u_2,u_5,u_6,u_9\}$
r_3	$\{p_3,p_8,p_9\}$	$\{u_0,u_1,u_3,u_4\}$
r_4	$\{p_0,p_7\}$	$\{u_7,u_8\}$
r_5	$\{p_5\}$	$\{u_5\}$

(d) Candidate roles



(e) Visual representation of roles

Figure 6.1 Visualization examples

Third, a textual role representation (Figure 6.1(d)) reports on information about role-user and role-permission relationships in a less communicative fashion than a graphical representation (Figure 6.1(e)). For instance, Figure 6.1(e) clearly shows that r_2 and r_3 partially overlap, without the need for any additional textual or graphical report. Notice that these representations are not mutually exclusive, but they can coexist in the same role engineering

tool. The tool can also enrich the matrix by providing interactive functionalities such as: drill-down capabilities, highlighting multiple roles, tooltips over cells, etc.. Interactions on the matrix can be turned into intelligence to tune underlying analytical process.

Finally, note that it could be difficult to produce a graphical representation for huge datasets. Yet, scalability is a major problem in both automatic and visual analysis [5]. In fact, a large number of user-permission assignments to analyze usually leads existing role mining algorithms to elicit a large number of roles, thus making hard any kind of data analysis. A viable solution is restricting the analysis to smaller subsets of data that are "homogeneous" with respect to some business-related information as suggested in Chapter 4 (e.g., partitioning users by department, job title, cost center, etc.).

6.1.1 Problem Formalization

In the previous section we intuitively demonstrated that reordering rows and columns of a user-permission matrix can ease the pattern-finding task. We now formalize this problem, offering a tool for the identification of the best representation for a given set of roles.

In addition to RBAC concepts introduced in Chapter 2, we also require the following definitions:

Definition 6.1 (Matrix Permutation) A *matrix permutation* $\sigma_{UP} = \langle \sigma_U, \sigma_P \rangle$ is a pair of bijective functions defined as $\sigma_U : USERS \to \{1, \ldots, |USERS|\}$ and $\sigma_P : PERMS \to \{1, \ldots, |PERMS|\}$.

Matrix permutation is introduced just to provide an ordering for users and permissions by "labeling" them with a number. Note that a matrix permutation uniquely identifies a matrix representation—we will thus use the terms "permutation" and "representation" as synonyms.

Definition 6.2 Given a matrix permutation $\sigma_{UP} = \langle \sigma_U, \sigma_P \rangle$ and a role $r \in ROLES$, the functions $\omega_U : ROLES \to \mathbb{N}$ and $\omega_P : ROLES \to \mathbb{N}$ identify the *height* and *width*, respectively, of r in the given permutation. That is:

$$\omega_U(r) = \max_{u \in ass\_users(r)} \sigma_U(u) - \min_{u \in ass\_users(r)} \sigma_U(u) + 1, \tag{6.1}$$

$$\omega_P(r) = \max_{p \in ass\_perms(r)} \sigma_P(p) - \min_{p \in ass\_perms(r)} \sigma_P(p) + 1. \tag{6.2}$$

In other words, $\omega_U(r)$ (or $\omega_P(r)$) represents the distance between the first and the last user (or permission) of r in the given matrix representation.

Definition 6.3 Given a matrix permutation $\sigma_{UP} = \langle \sigma_U, \sigma_P \rangle$ and a role $r \in ROLES$, the functions $\pi_U : ROLES \to \mathbb{N}$ and $\pi_P : ROLES \to \mathbb{N}$ identify the *number of user and permission fragments* in the given permutation, that is

$$\pi_U(r) = \sum_{u \in ass\_users(r)} \left[\min_{\substack{u' \in ass\_users(r): \\ \sigma_U(u') > \sigma_U(u)}} \sigma_U(u') - \sigma_U(u) \neq 1 \right] + 1, \tag{6.3}$$

$$\pi_P(r) = \sum_{p \in ass\_perms(r)} \left[\min_{\substack{p' \in ass\_perms(r): \\ \sigma_P(p') > \sigma_P(p)}} \sigma_P(p') - \sigma_P(p) \neq 1 \right] + 1, \tag{6.4}$$

where $[b]$ equals 1 when the predicate b is true, and 0 otherwise.

When $\pi_U(r) = 1$ (or $\pi_P(r) = 1$) all the users (or permissions) assigned to the role r are contiguous in the matrix representation. Otherwise, the corresponding rows (or columns) are partitioned into a certain number $\pi_U(r)$ (or $\pi_P(r)$) of subsets of contiguous rows (or columns).

Definition 6.4 Given a matrix permutation $\sigma_{UP} = \langle \sigma_U, \sigma_P \rangle$ and a role $r \in ROLES$, the *Role Visualization-Cost* $\nu_{\sigma_{UP}} : ROLES \to \mathbb{N}$ is:

$$\nu_{\sigma_{UP}}(r) = \big(\pi_U(r) \times \pi_P(r)\big) \times \big(\omega_U(r) \times \omega_P(r) - |ass\_users(r)| \times |ass\_perms(r)|\big). \tag{6.5}$$

The previous definition is a measure of the visual *fragmentation* of a role. It depends on the number of role fragments (i.e., sub-matrices made up of contiguous "on" cells), represented by the quantity $\pi_U(r) \times \pi_P(r)$, weighted by the number of cells "wasted" to represent the role with respect to its compact representation, that is $\omega_U(r) \times \omega_P(r) - |ass\_users(r)| \times |ass\_perms(r)|$. Notice that when all the cells of a role are contiguous, the corresponding cost is zero.

We would like to point out that an alternative visualization-cost that we could have used is the *half-perimeter* [45], defined as:

$$\nu'_{\sigma_{UP}}(r) = \omega_U(r) + \omega_P(r), \tag{6.6}$$

namely the sum of the height and width of roles in the given matrix representation. In our opinion, Equation 6.5 is more straightforward because a high role fragmentation greatly hinders the readability of the matrix, an aspect that the previous Equation does not catch.

Having introduced a visualization cost function makes it possible to define the following problem:

Definition 6.5 Given a set of roles *ROLES*, let $\sigma^*_{\text{UP}} = \langle \sigma^*_{\text{U}}, \sigma^*_{\text{P}} \rangle$ be a matrix permutation, and let $v_{\sigma^*_{\text{UP}}}$ be the corresponding role visualization-cost. We say that σ^*_{UP} is *optimal* when it minimizes the following:

$$\arg\min_{\sigma_{\text{UP}}} \sum_{r \in ROLES} v_{\sigma_{\text{UP}}}(r). \tag{6.7}$$

We refer to the search for the optimal permutation as the *Optimal Matrix-Permutation* (OMP) *optimization problem*. An important property of OMP is:

Theorem 6.1 *The OMP optimization problem is $\mathcal{NP}$-hard.*

PROOF To prove the $\mathcal{NP}$-hardness of OMP, we show a polynomial-time reduction of another $\mathcal{NP}$-hard problem to OMP. In particular, we provide a reduction of the *Minimum Linear Arrangement* (MLA) problem, which is known to be $\mathcal{NP}$-hard [37], to this problem (i.e. proving that MLA $\leq_p$ OMP). The MLA problem can be formulated as follows: given a graph $G = \langle V, E \rangle$, find an ordering σ for V such that $\sum_{\langle i,j \rangle \in E} |\sigma(v_i) - \sigma(v_j)|$ is minimized. This can be reduced in polynomial-time to a special case of the optimal matrix-permutation problem: that is, when we have as many roles as users, each user is assigned to only one role, and each role is assigned with two permissions. The set V represents the permissions, and we put an edge between two permissions if they belong to the same role, namely the edges in E correspond to users (rows) with their own roles.

Please note that minimizing $\sum_{\langle i,j \rangle \in E} |\sigma(v_i) - \sigma(v_j)|$ is equivalent to minimizing $\sum_{\langle i,j \rangle \in E} (|\sigma(v_i) - \sigma(v_j)| - 1)$. In the given OMP instance, this new quantity represents the sum of the number of "off" cells between two "on" cells in a row. Moreover, there can only be one "gap" between the columns of each role (i.e., each role is represented by at most two fragments). Consequently, given this polynomial-time many-one reduction, the identification of the optimal matrix-permutation that sorts permissions (columns) in order to place the two "on" cells as close as possible in each row—corresponds to MLA. Thus, completing the proof. ■

The previous theorem entails no polynomial-time solution for OMP. Hence, the following section describes a fast heuristic algorithm that is able to find an acceptable solution for the problem in many practical scenarios.

6.1.2 Matrix Sorting Algorithm

By leveraging on the observations made in the previous section, we now describe a viable, fast heuristic algorithm called ADVISER (*Access Data VISual-*

```
 1: procedure ADVISER(USERS, PERMS, ROLES, UA, PA)
 2:     σ_u ← SortSet(USERS, UA, ROLES)
 3:     σ_p ← SortSet(PERMS, PA, ROLES)
 4:     return σ_u, σ_p
 5: end procedure

 6: procedure SortSet(ITEMS, IA, ROLES)
 7:     ITEMS‾ ← {I ⊆ ITEMS | ∀i, i' ∈ I, roles(i) = roles(i')}
 8:     σ ← ∅
 9:     for all I ∈ ITEMS‾ sorted by descending areas of roles(I) do
10:         if |σ| < 2 then
11:             σ.append(I)
12:         else
13:             if Jacc(I, σ.first) > Jacc(I, σ.last) then
14:                 p ← 1,          j ← Jacc(I, σ.first)
15:             else
16:                 p ← |σ| + 1,    j ← Jacc(I, σ.last)
17:             end if
18:             for i = 2...|σ| do
19:                 j_prec ← Jacc(I, σ[i − 1]),    j_succ ← Jacc(I, σ[i])
20:                 j_curr ← Jacc(σ[i − 1], σ[i])
21:                 if max{j_prec, j_succ} > j ∧ min{j_prec, j_succ} ≥ j_curr then
22:                     p ← i,    j ← max{j_prec, j_succ}
23:                 end if
24:             end for
25:             σ.insert(p, I) {between the (p − 1)^th and the p^th elements}
26:         end if
27:     end for
28:     return σ.expand
29: end procedure
```

Figure 6.2 The ADVISER algorithm

izER). Given a set of roles, this algorithm is able to provide a compact representation of them. In particular, it reorders rows and columns of the user-permission matrix to minimize the fragmentation of each role. Despite being relatively simple, it provides a good—though not necessarily optimal—and fast solution to the otherwise intractable OMP problem. In particular, its running time is $O\big(n \times (|ROLES| + \log n)\big)$ where $n = max\{|USERS|, |PERMS|\}$.

6.1.3 Algorithm Description

As a heuristic, ADVISER is based on some intuitions, summarized in the following:

- Introducing a "gap" in the visualization of "large" roles (namely, those roles that involve many users and permissions) increases the cost more than introducing gaps on smaller roles. Hence, larger roles should be better represented.
- The more fragments in the visualization of a role, the higher the role visualization-cost.
- Reordering users but not permissions only affects the number of gaps between columns, and so do permissions.

As for the first point, one can argue that small roles can be more important from a business perspective since they likely represent administrative tasks. To focus on exceptions, large roles can be removed after their identification. Notice that searching for large-area tiles is also the choice of many other mining techniques [7, 38, 75, 78].

The algorithms described in Figure 6.2 implements our approach. A detailed description follows:

1. Rows and columns are sorted independently. ADVISER decomposes the optimal matrix-permutation problem into two sub-problems, that is users (Line 2) and permissions (Line 3) are sorted independently. Due to this symmetry, from now on we generically refer to rows and columns as *items*.

2. If some items are assigned to the same set of roles, they are put together. For this reason, the algorithm sorts groups of items, called *itemsets*, instead of individual items. In Figure 6.2 the function $roles\colon IA \to 2^{ROLES}$ identifies all roles associated with an item, namely $roles(i) = \{r \in ROLES \mid \langle i, r\rangle \in IA\}$. Line 7 identifies items assigned to the same roles. Given an itemset $I \in \overline{ITEMS}$, with abuse of notation in the following we refer to $roles(I)$ as the set of roles $roles(i)$ for any $i \in I$.

3. Itemset positions are decided one-by-one. In order to facilitate a better representation of large roles, itemsets involving roles with larger areas are analyzed first. Line 9 implements this behavior. In particular, let $I, I' \in \overline{ITEMS}$ be two itemsets. Then, I is considered before I' only if

$$\max_{r \in roles(I) \setminus roles(I')} |ass\_users(r) \times ass\_perms(r)| > \max_{r' \in roles(I') \setminus roles(I)} |ass\_users(r') \times ass\_perms(r')|.$$

When $roles(I) \setminus roles(I') = \emptyset$ or $roles(I') \setminus roles(I) = \emptyset$ we assume that $max = 0$.

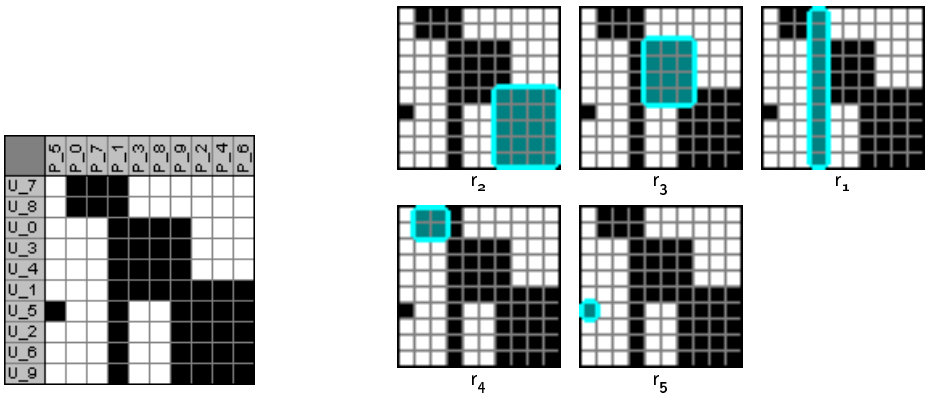
4. The algorithm tries to avoid large gaps by putting itemsets close to each other when they share large roles. First of all, we introduce a metric to rank the similarity of items in terms of shared large-area roles. To do this we resort again to the *Jaccard coefficient*. Given two sets X and Y, $J(X,Y) = |X \cap Y| \,/\, |X \cup Y|$. A natural generalization is to consider n-dimensional non-negative vectors X, Y and define $J(X,Y) = \sum_{i=1}^{n} \min\{X_i, Y_i\} \;/\; \sum_{i=1}^{n} max\{X_i, Y_i\}$. In our context, we measure the similarity of two items $i, i' \in ITEMS$ in terms of their assigned roles, weighted by the "depth" of possible gaps in roles. This is done via the following variation of the Jaccard coefficient:

$$\mathsf{Jacc}(i,\, i') = \frac{\sum_{r \in roles(i) \cap roles(i')} |m(r)|}{\sum_{r \in roles(i) \cup roles(i')} |m(r)|} \quad , \tag{6.8}$$

where $m(\cdot)$ is the membership function $ass\_users(\cdot)$ when we sort permissions, or the function $ass\_perms(\cdot)$ when we sort users. Summarizing, we try to put closer those users (permissions) that share roles with lots of permissions (users). This allows to reduce the number of cells between fragments of large roles. Given $I, I' \in \overline{ITEMS}$, with abuse of notation we refer to $\mathsf{Jacc}(I,\, I')$ as the value of $\mathsf{Jacc}(i,\, i')$ for any $i \in I$ and $i' \in I'$.

5. Each itemset is preferentially positioned at the beginning or at the end of already sorted itemsets. The idea is to avoid to "worsen" already found, high similarities. Having defined the previous similarity metric between items, lines 10–26 implement the itemset-sorting strategy by deciding a position p for the itemset I in an itemset permutation σ. The first two itemsets are just inserted in the first two positions (lines 10–11). Then, subsequent itemsets are inserted among already-sorted itemsets only when this operation actually improves the existing sorting. In particular, an itemset I is put but between two consecutive itemsets $\sigma[i-1], \sigma[i]$ (i.e., the two already-sorted items at positions $i-1$ and i) only when both the similarities between I and $\sigma[i-1]$ and between I and $\sigma[i]$ are below the similarity between $\sigma[i-1]$ and $\sigma[i]$ (lines 19–23). Among all possible positions, the algorithm seeks the one that provides the highest similarity: this is done by updating the variables "j" (maximum similarity value found) and "p" (position where the maximum similarity has been found). If inserting the itemset between already sorted itemsets is not advantageous, the itemset will be inserted at the beginning (lines 13–14) or at the end (lines 15–16) of the permutation σ.

6. Itemset sorting is converted to item sorting. This is the inverse of previous point 2. When all itemsets in $\overline{ITEMS}$ have been sorted, they are "expanded" (see Line 28) to return the ordering of each single item in *ITEMS*—instead of providing an ordering for group of items that share the same roles.



(a) (b)

Figure 6.3 Application of ADVISER on two role sets

We now demonstrate that the complexity of ADVISER as depicted in Figure 6.2 is $O\big(n \times (|ROLES| + \log n)\big)$ where $n = max\{|USERS|, |PERMS|\}$. To prove this, we first show that SortSet has a running time equal to

$$O\big(|ITEMS|\,(|ROLES| + \log |ITEMS|)\big)$$

Indeed, Line 7 requires a running time $O\big(|ITEMS|\,|ROLES|\big)$ because we have to scan all items and, for each item, check the corresponding roles. The set $\overline{ITEMS}$, such that $|\overline{ITEMS}| \leq |ITEMS|$, can be sorted at Line 9 in time $O\big(|ITEMS| \log |ITEMS|\big)$. All the statements of the loop from Line 9 to Line 27 can be executed in a constant time, except for the computation of $\mathsf{Jacc}(\cdot,\cdot)$ that requires a running rime $O\big(|ROLES|\big)$. Consequently, the total computation cost is $O\big(|ITEMS|\,(|ROLES| + \log |ITEMS|)\big)$. The complexity of ADVISER immediately follows.

6.1.4 Example

A simple example can help to better understand the behavior of the algorithm ADVISER. Starting from the user-permission relationships introduced in Figure 6.1, Figure 6.3(a) is obtained by applying ADVISER over the roles depicted in Figure 6.3(b), sorted by descending area. We only describe the sorting of users, since similar considerations can be made for permissions.

- First, the algorithm groups users assigned to the same roles and sort them by descending role areas. In our example, sorted user-sets are:

$\{u_1\}$ (assigned to roles r_2, r_3, r_1), $\{u_5\}$ (assigned to roles r_2, r_1, r_5), $\{u_2, u_6, u_9\}$ (assigned to roles r_2, r_1), $\{u_0, u_3, u_4\}$ (assigned to roles r_3, r_1), and $\{u_7, u_8\}$ (assigned to roles r_1, r_4).

- Then, the first two user-set are just put together, namely $\sigma = \big\{\{u_1\}, \{u_5\}\big\}$.
- In turn, we seek a position for $\{u_2, u_6, u_9\}$. The maximum similarity value is

$$\begin{aligned} &\mathsf{Jacc}(\{u_2, u_6, u_9\}, \{u_5\}) \\ &\quad = \sum_{r \in \{r_2, r_1\}} |ass\_perms(r)| \;/\; \sum_{r \in \{r_2, r_5, r_1\}} |ass\_perms(r)| \\ &\quad = (4+1)/(4+1+1) = 0.83 \end{aligned} \tag{6.9}$$

  Indeed, the first user-set has $\mathsf{Jacc}(\{u_2, u_6, u_9\}, \{u_1\}) = 0.63$ and then the current user-set cannot be inserted at the beginning. Moreover, the similarity between the two already-sorted items is $\mathsf{Jacc}(\{u_1\}, \{u_5\}) = 0.56$; this means that inserting the user-set between them is potentially advantageous, but this would not increase the maximum similarity found at the first position. Hence, $\sigma = \big\{\{u_1\}, \{u_5\}, \{u_2, u_6, u_9\}\big\}$.
- Similarly, $\{u_0, u_3, u_4\}$ is inserted at the beginning because the maximum similarity is $\mathsf{Jacc}(\{u_0, u_3, u_4\}, \{u_1\}) = 0.5$, thus

$$\sigma = \big\{\{u_0, u_3, u_4\}, \{u_1\}, \{u_5\}, \{u_2, u_6, u_9\}\big\}$$

- Finally, $\{u_7, u_8\}$ is inserted at the beginning because of the Jaccard Coefficient $\mathsf{Jacc}(\{u_7, u_8\}, \{u_0, u_3, u_4\}) = 0.16$. Thus, the final ordering is equal to: $\sigma = \big\{\{u_7, u_8\}, \{u_0, u_3, u_4\}, \{u_1\}, \{u_5\}, \{u_2, u_6, u_9\}\big\}$.

Please also note that, in this small example, all roles have been best represented. When roles are not overlapping, namely each role involves different users and permissions, the algorithm always provides good visualization results. Notice that there is no particular strategy in positioning each role within the matrix: the algorithm only strives to reduce the number of fragments required to represent each role.

6.2 Visual Elicitation of Roles

In Section 6.1 we pointed out that a good matrix permutation can help role engineers elicit candidate roles. By just inspecting the matrix—that is, without analyzing the outcome of any role mining algorithm—analysts can intuitively select the more relevant roles. When we want to identify roles through

visual analysis, a natural question is how a role-set for ADVISER should be made in order to facilitate this task. An approach is to first compute all possible *closed permission-sets* and later trying to best represent them. A permission-set is "closed" when no proper supersets of permissions possessed by the same users exist. Examples of algorithms that compute such patterns are [7, 78, 83]. Closed permission-sets provide a compressed representation of *all* possible permission combinations that can be found within users [83]. Closed permission-sets are roles in RBAC terminology.

By feeding ADVISER with closed permission-sets we provide analysts with a matrix visualization that seeks to contextually best depict all identifiable patterns. However, the number of closed permission-sets is often too large when compared to the number of users and permissions [42]. Hence, leading to long running time and huge memory footprint. To reduce the overall problem complexity, we introduce a probabilistic algorithm called EXTRACT (*EXception-Tolerant Role ACTualizer*). It generates a list of *pseudo-roles* used to feed ADVISER in lieu of closed permission-sets. We will show that pseudo-roles and closed permission-sets lead to very similar results. Further, we will demonstrate that computing such pseudo-roles takes just $O(k|UP|)$, where k is a tunable parameter of the algorithm.

6.2.1 Using Pseudo-Roles

Let us now recall the definition of pseudo-role given in Chapter 4:

Definition 6.6 Given a user-permission assignment $\langle u, p\rangle \in UP$, the *pseudo-role generated by* $\langle u, p\rangle$, and hereafter referred to as $\varrho_{\langle u,p\rangle}$, is a role represented by all users having the permission p granted and all permissions granted to the user u, namely $ass\_users(\varrho_{\langle u,p\rangle}) = users(p)$ and $ass\_perms(\varrho_{\langle u,p\rangle}) = perms(u)$.

Several user-permission assignments can generate the *same* pseudo-role. In particular:

Definition 6.7 Let *P-ROLES* be the set of all pseudo-roles that can be generated from user-permission assignments in *UP*. The number of assignments that generates a pseudo-roles $\hat{\varrho} \in P\text{-}ROLES$ is referred to as its *frequency*, defined as a function $f : P\text{-}ROLES \to \mathbb{N}$ such that

$$f(\hat{\varrho}) = \Big|\{\langle u, p\rangle \in UP \mid ass\_users(\hat{\varrho}) = users(p) \wedge ass\_perms(\hat{\varrho}) = perms(u)\}\Big|. \quad (6.10)$$

When using pseudo-roles in place of roles, the main objective is to best represent pseudo-roles that likely have the largest area, and have no or only

(a) (b)

Figure 6.4 Typical configurations for EXTRACT

few non-existing user-permission relations. The frequency concept summarizes both these aspects. An example can support the previous statement. Figure 6.4 shows two possible submatrices of a larger user-permission matrix. All user-permission relationships have been divided in three subsets: A, B, and C. Non-existing user-permission relationships (that is, $USERS \times PERMS \setminus UP$) are indicated with D. In both figures, the same three pseudo-roles can be generated: *i*) every assignment in A generates a pseudo-role made up of A, B, C, D; *ii*) assignments in B generate A, B; and, *iii*) assignments in C generate A, C. In Figure 6.4(a), the most frequent pseudo-role—the one with the highest value for f—is represented by *i*), since generating user-permission relationships belong to the largest area A. Since D is small, it likely represents missing assignments. Hence, it is advantageous to put together all the cells of A, B, C, and D. Conversely, in Figure 6.4(b) the most frequent pseudo-role is *ii*). In this case, assignments in C likely represent exceptions. Hence, representing the cells of C close to A, B is probably not important.

Based on the previous observations, we propose to *represent frequent pseudo-roles better than the infrequent ones*. This can be done by properly adapting Line 9 of Figure 6.2 and sorting pseudo-roles by descending frequency. Moreover, a new definition for the item similarity is required, that is

$$\mathsf{Jacc}(i, i') = \frac{\sum_{\varrho \in roles(i) \cap roles(i')} |m(\varrho)| \times f(\varrho)}{\sum_{\varrho \in roles(i) \cup roles(i')} |m(\varrho)| \times f(\varrho)}, \tag{6.11}$$

where $roles(i) = \{\varrho_{\langle u,p \rangle} \mid \langle u, p \rangle \in UP \ \wedge \ i \in ass\_users(\varrho_{\langle u,p \rangle})\}$ when sorting permissions, while $roles(i) = \{\varrho_{\langle u,p \rangle} \mid \langle u, p \rangle \in UP \ \wedge \ i \in ass\_perms(\varrho_{\langle u,p \rangle})\}$ when sorting items.

```
1:  procedure EXTRACT(UP, k)
2:      P-ROLES, P-UA, P-PA ← ∅
3:      for i = 1...k do
4:          {Identify the current pseudo-role}
5:          Pick ⟨u, p⟩ ∈ UP uniformly at random
6:          U ← users(p)
7:          P ← perms(u)
8:          {Check if the pseudo-role has been previously generated}
9:          if ∃ϱ ∈ P-ROLES : ass_users(ϱ) = U ∧ ass_perms(ϱ) = P then
10:             {Update the frequency of the existing pseudo-role}
11:             ϱ ← existing pseudo-role
12:             count[ϱ] ← count[ϱ] + 1
13:         else
14:             {Add a new pseudo-role to P-ROLES}
15:             ϱ ← new pseudo-role
16:             P-ROLES ← P-ROLES ∪ {ϱ}
17:             P-UA ← P-UA ∪ (U × {ϱ})
18:             P-PA ← P-PA ∪ (P × {ϱ})
19:             count[ϱ] ← 1
20:         end if
21:     end for
22:     return P-ROLES, P-UA, P-PA, count[·]
23: end procedure
```

Figure 6.5 The EXTRACT algorithm

6.2.2 Algorithm Description

Based on Definition 6.6, a naïve approach to generate all pseudo-roles is to scan all assignments in $\langle u, p\rangle \in UP$ and identifying the corresponding pseudo-role by computing $users(p)$ and $perms(u)$. During the scanning, whenever we generate an already existing pseudo-role, we update its frequency. This intuitive and simple algorithm has a running time $O(|UP| \log |UP|)$. Assuming that UP is ordered, the search for all the users possessing p and all the permissions assigned to u can be executed in $O(\log |UP|)$, and this must be done for all assignments in UP. In the worst case we generate $|UP|$ pseudo-roles—i.e., a different pseudo-role for each assignment. Hence, searching and updating the frequency requires $O(\log |UP|)$, for instance by storing pseudo-roles in a self-balancing binary search tree.

Although this algorithm is quite efficient, we can still obtain better results.

In particular, in the following we present a very fast randomized algorithm to generate pseudo-roles called EXTRACT. The key idea is that of *approximating* the frequencies of pseudo-roles by sampling k times a relationship in UP uniformly at random, and then generating the corresponding pseudo-role. In turn, for each pseudo-role we count how many times it has been generated. Figure 6.5 summarizes this approach. The computational cost of EXTRACT is $O(k \log |UP|)$. Indeed, the main loop (lines 3–21) is executed k times. The random selection of $\langle u, p \rangle$ (Line 5) is supposed to be executed in $O(1)$. Moreover, searching all the users possessing p and all the permissions assigned to u (lines 6–7) can be executed in $O(\log |UP|)$. Lines 9–9 can be executed in $O(\log |UP|)$. All the remaining statements can be performed in $O(1)$. Hence, the overall computational complexity is $O(k \log |UP|)$.

The following theorem gives a bound on the approximation introduced by this sampling approach:

Theorem 6.2 *Let k be the number of the randomly chosen user-permission assignments by EXTRACT. Given a pseudo-role ϱ, let $\tilde{f}(\varrho)$ be the actual number of times the pseudo-role has been generated by the algorithm. Hence,*

$$\Pr\left(\left|\frac{\tilde{f}(\varrho)}{k} - \frac{f(\varrho)}{|UP|}\right| \geq \varepsilon\right) \leq 2\exp\left(-2k\varepsilon^2\right). \tag{6.12}$$

PROOF We will use the Hoeffding inequality [44] to prove this theorem. It says that if $X_1 \ldots X_k$ are independent random variables such that $0 \leq X_i \leq 1$, then

$$\Pr\left(\left|\sum_{i=1}^{k} X_i - \mathbb{E}\left[\sum_{i=1}^{k} X_i\right]\right| \geq t\right) \leq 2\exp\left(-\frac{2t^2}{k}\right),$$

where $\mathbb{E}[\cdot]$ indicates the expected value of a random variable. X_i is such that

$$X_i = \begin{cases} 1, & \text{the } i^{\text{th}} \text{ assignment generates the pseudo-role;} \\ 0, & \text{otherwise.} \end{cases}$$

The previous equation can be rewritten as

$$\Pr\left(\left|\frac{1}{k}\sum_{i=1}^{k} X_i - \mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} X_i\right]\right| \geq \varepsilon\right) \leq 2\exp\left(-2k\varepsilon^2\right),$$

where $\varepsilon = t/k$. Note that $\sum_{i=1}^{k} X_i$ is exactly $\tilde{f}(\varrho)$.

To complete the proof, we have to prove that $\mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} X_i\right]$ is equal to $f(\varrho)/|UP|$. Because of the linearity of the expectation, the following equation holds:

$$\mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} X_i\right] = \frac{1}{k}\sum_{i=1}^{k}\mathbb{E}[X_i].$$

Since the user-permission assignments are picked uniformly at random, the probability to choose each of them is $1/|UP|$. Thus,

$$\forall i \in 1 \dots k, \quad \mathbb{E}[X_i] = \sum_{j=1}^{|UP|} \frac{X_j}{|UP|} = \frac{f(\varrho)}{|UP|},$$

completing the proof. ■

Theorem 6.2 proves that there exists a value k such that the matrix permutation obtained by adopting sampled frequencies is, with a given probability, almost the same as using exact frequencies—the absolute difference between the two results is bounded. Both the absolute difference and the given probability are tunable parameters depending on k.

Note that if the visualization quality is poor due to the approximated frequency values, it is possible to improve the quality by performing just additional samples. Suppose to have the matrix representation generated by feeding the algorithm ADVISER with the output of the algorithm EXTRACT with k samples: if we are not satisfied by this matrix, we can use k' additional samples (namely, we run the loop from Line 3 to Line 21 k' more times) in order to have a more accurate frequency estimation.

6.2.3 A Visual Approach to Role Engineering

In the following we will show an application of the visualization and sampling algorithms finalized to a visual role engineering activity. In particular, we will illustrate how to perform role engineering upon the matrix representation obtained through ADVISER when fed by EXTRACT. Further, we will show how to identify potentially wrong or missing assignments. This methodology originates from a real case-study that has been carried out on a large private company. To protect company privacy, we will not reveal any detail of the results, but we limit ourself to summarizing the methodology.

The proposed approach is an iterative method, mainly inspired by the role-finding process proposed by Kuhlmann et al. [52]. First, according to [14,15], we simplify the role-finding task decomposing the problem into smaller sub-problems. Then, for each sub-problem, we suggest to conduct the following activities:

1. Select the most relevant roles by resorting to a visual inspection. Then, the corresponding user-permission assignments should be put aside. Analysts should visually recognize the most clearly visible roles, namely those corresponding to the biggest tiles, and remove them for the next iteration.

2. Identify the business managers responsible for the involved users (typically referred to as "user managers") and the administrators within IT staff responsible for the involved data (typically referred to as "data owners") of each candidate role.

3. In concert with user managers and data owners, understand the meaning of exceptional user-permission assignments. That is, analyze those assignments that are depicted on the "ragged edges" of the main tiles. In particular, analysts should try to understand whether such assignments are actually required or not. Further, they should verify whether "holes" within almost perfect tiles are missing user-permission relationships that could be assigned to users without violating the least-privilege principle.

After having put aside those user-permission assignments covered by already identified roles, the analysis can be iteratively repeated over the remaining data. Notice that discovering exceptional assignments and subsequently removing them is a good way to keep policy engineers in the work loop and still provide valuable feedback. If the feedback of the analysis is fast enough, this is a very effective technique: in real cases, we performed very few iterations (up to 4), eliciting a limited number of roles when compared to the cardinality of the assignment set.

Another important observation relates to the identification of user managers and data owners. This task is often easy whenever the divide-and-conquer approaches proposed in [14, 15] are adopted. The reason is that the identified patterns likely reflect the actual business of the company.

6.2.4 Experimental Results

This section presents practical applications of our visualization methodology. First, we will discuss the efficiency and the quality of our algorithms. Then, we will report on a comparative analysis against a competing approach (i.e., [45]) by using publicly available datasets. The testbed was a notebook equipped with an Intel Pentium Core Duo Pro processor operating at 2.40 GHz, and 3 GB RAM. The operating system was Linux Fedora 8—in order to be compatible with the code provided by the authors of [45]. The algorithms were coded in

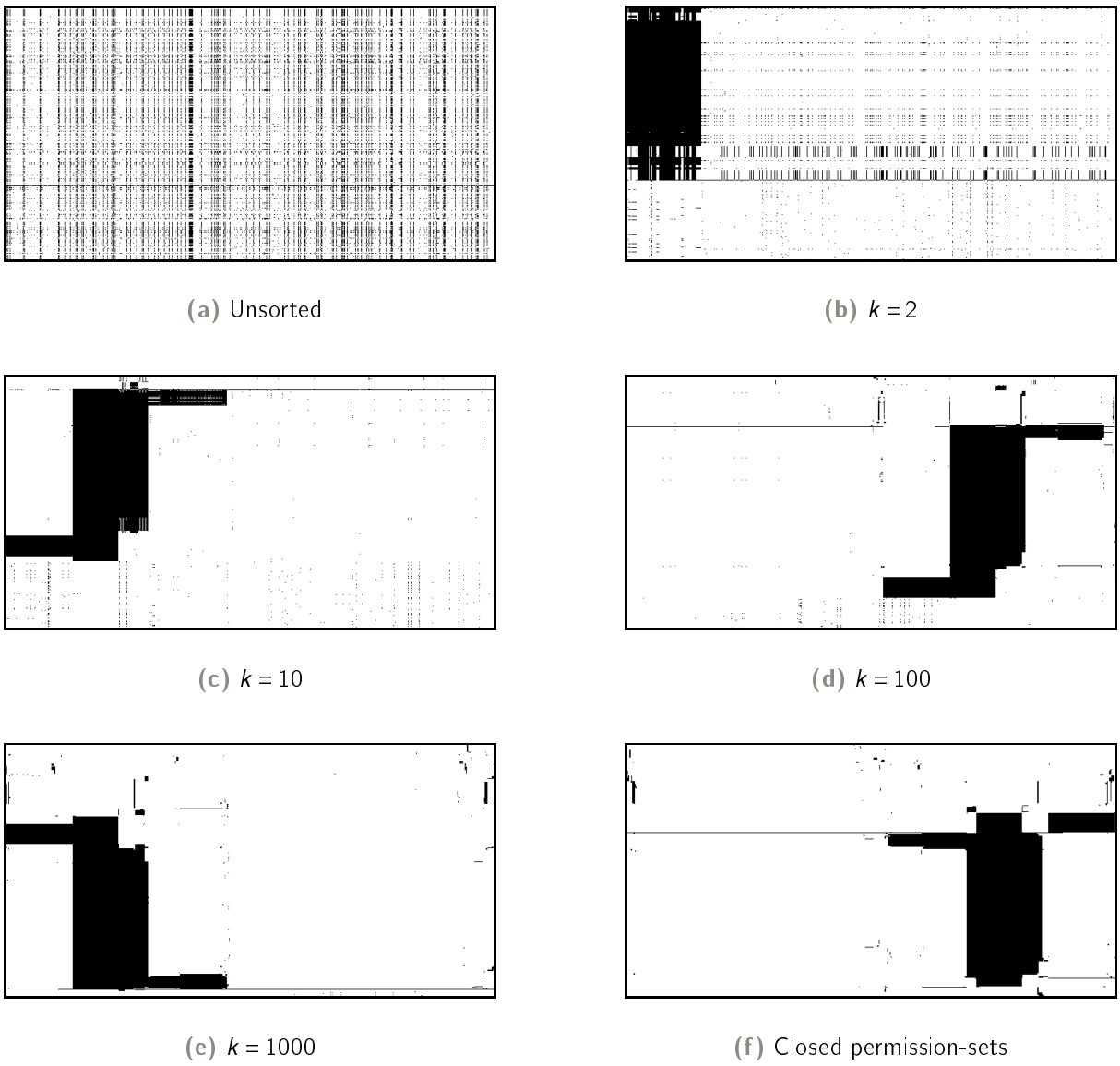
Java. Since we ran the experiments in a multitasking environment, the values provided are an upper bound of the real computation time.

6.2.5 Matrix Sorting

Figure 6.6 shows the application of our algorithms on a given dataset. Figure 6.6(a) depicts the data without any sorting. Instead, figures from 6.6(b) to 6.6(e) show the results obtained when using ADVISER fed with the pseudo-roles generated by EXTRACT, respectively for $k = \{2, 10, 100, 1000\}$. Table 6.1 reports, among other data, the computation time to build each one of these pictures.

For $k = 2$ (Figure 6.6(b)) only some users and some permissions have been sorted, but a candidate role that could manage a large number of user-permission assignments is already clear and visible. The number of "shuffled" rows and column decreases when $k = 10$ (Figure 6.6(c)). By using $k = 100$ (Figure 6.6(d)), most of the main patterns become clearer. By using a larger sampling parameter, namely $k = 1000$ (Figure 6.6(e)), there are very few differences when compared to Figure 6.6(d). The last example (Figure 6.6(f)) shows an application of the sorting algorithm when applied to the outcome of the algorithm proposed in [7], which computes closed permission-sets.

To provide a quantitative analysis of the quality of visualization results, the last column of Table 6.1 indicates the cost of visualizing all possible closed permission-sets. According to our expectation, the visualization cost decreases as the number of samples increases. Moreover, the differences between Figure 6.6(f) and Figure 6.6(d) are minimal. Although the running time of the algorithm proposed in [7] is definitely greater than that of EXTRACT, it does not lead to performance bottlenecks in the case study. Yet, the advantage of adopting EXTRACT is an almost real-time representation of the data to analyze. The situation dramatically changes as the dataset becomes larger. In particular, the time required to generate all possible closed permission-sets grows exponentially as the dataset dimension increases, whereas the generation of pseudo-roles increases according to a logarithmic law. Even though large matrices cannot entirely be represented on a personal computer screen, their construction is useful anyway. For instance, visualizing a small "sliding window" and/or zooming in/out still represents a valuable way of browsing data. As stated before, scalability is a major problem in both automatic and visual analysis.



(a) Unsorted

(b) $k = 2$

(c) $k = 10$

(d) $k = 100$

(e) $k = 1000$

(f) Closed permission-sets

Figure 6.6 Matrix representation of the access control configuration.

6.2.6 Concluding Remarks

To the best of our knowledge, this thesis is the first work in addressing the *visual role mining* problem. That is, visualizing user-permission assignments in an intuitive graphical form that makes it possible to simplify the role engi-

Table 6.1 Comparison among different algorithms and parameters

Figure	Algorithm	Samples	Number of (Pseudo-)Roles	Sampling/Mining Time (nsec)	Sorting Time (nsec)	Total Time (nsec)	Vis. cost $v_{\sigma_{up}}$ on closed perm-sets
6.6(a)	–	–	–	–	–	–	1.35×10^{15}
6.6(b)	Sampling	2	2	2	2	4	1.22×10^{14}
6.6(c)	Sampling	10	6	2	3	5	1.04×10^{13}
6.6(d)	Sampling	100	45	15	4	19	1.07×10^{12}
6.6(e)	Sampling	1000	201	149	22	171	4.55×10^{11}
6.6(f)	Closed perm-sets	-	315	310	23	333	2.33×10^{11}

neering process. The proposed representation of data allows role designers to gain insight, draw conclusions, and ultimately design meaningful roles from both IT and business perspectives.

We provided several contributions. First, we offered a formal description of the visual role mining problem. Second, we demonstrated that constructing the binary matrix representation of user-permission relations that best represents already recognized patterns is $\mathcal{NP}$-hard. Moreover, we proposed a novel heuristic algorithm called ADVISER to generate a matrix representation starting from the outcome of any role mining algorithm. We also described an efficient, tunable, and probabilistic tool referred to as EXTRACT. It produces approximate patterns that can be used in conjunction with ADVISER to obtain high-quality visualization results—the quality of the results produced by EXTRACT is formally proved. Finally, extensive applications over real and public data confirm that our approach is efficient, both in terms of computational time and result quality.

We introduced role engineering as a process which can greatly benefit from the visual approach proposed in this section. Our contributions, other than being useful for role engineering, can have interesting applications in other fields as well. For instance, binary matrices are used to analyze gene-expression data to uncover embedded relationships among DNA fingerprints. In particular, homogeneous submatrices indicate subsets of genes (rows) coexpressed under the same conditions (columns). Another application that could benefit from our approach is the well-known market-basket analysis. In this case, each transaction corresponds to a row and each item corresponds to column of the matrix. One possible application is in the identification of “dense” submatrices, namely approximate frequent itemset pattern identified by “on” cells with a small false-positive rate. In general, whenever there is a need to analyze data representable as a binary matrix, our approach can introduce benefits.

7 Conclusion

Final remarks on the contributions detailed in this thesis are provided in this chapter. The candidate also points out some future research directions that can pave the way to some further research.

Remarks on Contributions In this thesis we addressed the role mining problem, and in particular we looked at how to make it practical and usable for actual deployment. Several aspects regarding the business meaning of the elicited roles have been considered. Furthermore, we highlighted that previously proposed approaches usually lead to candidate role-set that cannot be easily managed by system administrators. We proposed several methodologies that can be used to reduce the role mining complexity, still providing roles that are meaningful from a business perspective. These methodologies can be also combined together in order to provide to the final user the best outcome of each approach.

The following additional observations can be made:

Performances We tackled performance issues from several viewpoints. In Section 4.2 we introduced a divide-and-conquer approach that, by dividing the dataset in smaller parts, allows for reduced running time of mining algorithms. Further, in Section 4.3 we introduced a six step methodology that overcome the drawback of the divide-and-conquer approach.

Data Noise The automatic recognition of exceptional access control data (i.e., exceptionally or erroneously granted or denied entitlements) can greatly simplify the elicitation of meaningful roles. In particular, we proposed the concept of "unstable" assignments in Chapter 4, showing that certain user's entitlements can be discarded from the analysis when they do not benefit from adopting RBAC. Furthermore, in Chapter 5, we showed

that imputing "missing" and "outlier" assignments can also be beneficial.

Role Mining Problem Complexity We accomplished a reduction of the role mining problem complexity in Chapter 4 through two different approaches: on the one hand, we described how to decompose a complex problem in simpler sub-problems. On the other hand, we introduced a methodology that overcomes the limitation of divide et impera approaches.

To conclude, this thesis provided several fundamental results for role mining. Several examples, developed on both synthetic and real data, support our claims.

Future Work Possible extensions of the current work are:

- To investigate new visualization models that differ from the traditional binary matrix. A candidate strategy could be the graph visualization that we already used in this thesis.
- To investigate machine learning algorithms to foresee the permissions to grant to new users when they are firstly introduced in the system. Existing attributes of the new users could be leveraged for this purpose.
- To investigate how to choose an optimal value for the threshold that is required to decide which assignments are stable and which are not. An analogous threshold is used for missing values. Indeed, although not expensive to tune, this parameter can hamper the applicability of the proposed methodologies. If an optimal value were automatically identified, role mining algorithm could directly be applied without any preventive human action, hence putting analysts out of the loop.

The aforementioned extensions strengthen the contributions of this thesis by paving the way to some further research.

Bibliography

[1] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 1994.

[2] American National Standards Institute (ANSI) and InterNational Committee for Information Technology Standards (INCITS). ANSI/INCITS 359-2004, Information Technology – Role Based Access Control, 2004.

[3] Stephen D. Bay and Mark Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 29–38, 2003.

[4] Simon Byers and Adrian E. Raftery. Nearest-Neighbor Clutter Removal for Estimating Features in Spatial Point Processes. *Journal of the American Statistical Association*, 93(442):577–584, 1998.

[5] Chaomei Chen. Top 10 unsolved information visualization problems. *IEEE Trans. Comp. Graph. Appl.*, 25(4):12–16, 2005.

[6] Alessandro Colantonio, Roberto Di Pietro, and Alberto Ocello. A cost-driven approach to role engineering. In *Proceedings of the 23rd ACM Symposium on Applied Computing, SAC '08*, volume 3, pages 2129–2136, Fortaleza, Ceará, Brazil, 2008.

[7] Alessandro Colantonio, Roberto Di Pietro, and Alberto Ocello. Leveraging lattices to improve role mining. In *Proceedings of the IFIP TC 11 23rd International Information Security Conference, SEC '08*, pages 333–347, 2008.

[8] Alessandro Colantonio, Roberto Di Pietro, and Alberto Ocello. *Role Mining in Business—Taming Role-Based Access Control Administration*. World Scientific Publishing Co. Inc, 2011.

[9] 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.

[10] Alessandro Colantonio, Roberto Di Pietro, Alberto Ocello, and Nino Vincenzo Verde. Mining stable roles in RBAC. In *Proceedings of the IFIP TC 11 24th International Information Security Conference, SEC '09*, pages 259–269, 2009.

[11] Alessandro Colantonio, Roberto Di Pietro, Alberto Ocello, and Nino Vincenzo Verde. A probabilistic bound on the basic role mining problem and its applications. In *Proceedings of the IFIP TC 11 24th International Information Security Conference, SEC '09*, pages 376–386, 2009.

[12] Alessandro Colantonio, Roberto Di Pietro, Alberto Ocello, and Nino Vincenzo Verde. ABBA: Adaptive bicluster-based approach to impute missing values in binary matrices. In *Proceedings of the 25th ACM Symposium on Applied Computing, SAC '10*, pages 1027–1034, 2010.

[13] Alessandro Colantonio, Roberto Di Pietro, Alberto Ocello, and Nino Vincenzo Verde. Evaluating the risk of adopting RBAC roles. In *Proceedings of the 24th Annual IFIP WG 11.3 Working Conference on Data and Applications Security, DBSec '10*, pages 303–310, 2010.

[14] Alessandro Colantonio, Roberto Di Pietro, Alberto Ocello, and Nino Vincenzo Verde. Mining business-relevant RBAC states through decomposition. In *Proceedings of the IFIP TC 11 25th International Information Security Conference, SEC '10*, pages 19–30, 2010.

[15] Alessandro Colantonio, Roberto Di Pietro, Alberto Ocello, and Nino Vincenzo Verde. A new role mining framework to elicit business roles and to mitigate enterprise risk. *Decision Support Systems*, Special Issue on "Enterprise Risk and Security Management: Data, Text and Web Mining", 2010. To appear.

[16] Alessandro Colantonio, Roberto Di Pietro, Alberto Ocello, and Nino Vincenzo Verde. Taming role mining complexity in RBAC. *Computers & Security*, 29:548–564, 2010. Special Issue on "Challenges for Security, Privacy & Trust".

[17] Alessandro Colantonio, Roberto Di Pietro, and Nino Vincenzo Verde. A business-driven decomposition methodology for role mining. *Computers & Security*, 2011. Submitted for revision.

[18] Alessandro Colantonio, Roberto Di Pietro, and Nino Vincenzo Verde. A novel approach to impute missing values and detecting outliers in binary matrices. *Data Mining and Knowledge Discovery (DMKD)*, 2011. Submitted for revision.

[19] Alessandro Colantonio, Roberto Di Pietro, and Nino Vincenzo Verde. Privacy preserving role-engineering. 2011. Submitted for revision.

[20] Alessandro Colantonio, Roberto Di Pietro, Alberto Ocello, and Nino Vincenzo Verde. Visual role mining: A picture is worth a thousand roles. *IEEE Transactions on Knowledge and Data Engineering*, 99(PrePrints), 2011.

[21] Edward J. Coyne. Role-engineering. In *Proceedings of the 1st ACM Workshop on Role-Based Access Control, RBAC '95*, pages 15–16, 1995.

[22] Edward J. Coyne and John M. Davis. *Role Engineering for Enterprise Security Management*. Artech House, December 2007.

[23] Robert Crook, Darrel Ince, and Bashar Nuseibeh. Towards an analytical role modelling framework for security requirements. In *Proceedings of the 8th International Workshop on Requirements Engineering: Foundation for Software Quality, REFSQ '02*, 2002.

[24] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 2 edition, 2002.

[25] Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[26] Roberto Di Pietro and Nino Vincenzo Verde. Introducing epidemic models for data survivability in unattended wireless sensor networks. In *Second International Workshop on Data Security and PrivAcy in wireless Networks (D-SPAN 2011)*, Lucca, Italy.

[27] Roberto Di Pietro and Nino Vincenzo Verde. Epidemic data survivability in unattended wireless sensor networks. In *Proceedings of the fourth ACM conference on Wireless network security*, WiSec '11, pages 11–22, Hamburg, Germany, 2011. ACM.

[28] Roberto Di Pietro and Nino Vincenzo Verde. Epidemic Theory and Data Survivability in Unattended Wireless Sensor Networks: Models and Gaps. *Pervasive and Mobile Computing*, 2011. Submitted for revision.

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

[30] P. Epstein and Ravi S. Sandhu. Engineering of role/permission assignments. In *Proceedings of the 17th Annual Computer Security Applications Conference, ACSAC 2001*, pages 127–136, 2001.

[31] Jean-Daniel Fekete, Jarke J. Wijk, John T. Stasko, and Chris North. The value of information visualization. In *Information Visualization: Human-Centered Issues and Perspectives*, pages 1–18, 2008.

[32] E. B. Fernandez and J. C. Hawkins. Determining role rights from use cases. In *Proceedings of the 2nd ACM Workshop on Role-Based Access Control, RBAC '97*, pages 121–125, 1997.

[33] Andres Figueroa, James Borneman, and Tao Jiang. Clustering binary fingerprint vectors with missing values for DNA array data analysis. In *CSB '03: Proceedings of the IEEE Computer Society Conference on Bioinformatics*, pages 38–47, 2003.

[34] Mario Frank, David Basin, and Joachim M. Buhmann. A class of probabilistic models for role engineering. In *Proceedings of the 15th ACM Conference on Computer and Communications Security, CCS '08*, pages 299–310, 2008.

[35] Mario Frank, Joachim M. Buhmann, and David Basin. On the definition of role mining. In *Proceedings of the 15th ACM Symposium on Access Control Models and Technologies, SACMAT '10*, pages 35–44, 2010.

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

[37] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete problems. In *Proceedings of the 6th annual ACM Symposium on Theory of Computing, STOC '74*, pages 47–63, 1974.

[38] Floris Geerts, Bart Goethals, and Taneli Mielikäinen. Tiling databases. In *Discovery Science*, pages 278–289, 2004.

[39] Amol Ghoting, Srinivasan Parthasarathy, and Matthew Otey. Fast mining of distance-based outliers in high-dimensional datasets. *Data Mining and Knowledge Discovery*, 16:349–364, 2008.

[40] William Philip Gramm, James Albert Smith Leach, and Thomas Jerome Bliley. Eu privacy protection directive 2002/58/ec. Directive of the European Parliament and of the Council of 12 July 2002.

[41] William Philip Gramm, James Albert Smith Leach, and Thomas Jerome Bliley. Gramm-Leach-Bliley Financial Services Modernization Act of 1999. Act of the United States Congress. Pub. L. No. 106-102, 113 Stat. 1338. Also known as the "Gramm-Leach-Bliley Act".

[42] Rohit Gupta, Gang Fang, Blayne Field, Michael Steinbach, and Vipin Kumar. Quantitative evaluation of approximate frequent pattern mining algorithms. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '08*, pages 301–309, 2008.

[43] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2 edition, 2006.

[44] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[45] Ruoming Jin, Yang Xiang, David Fuhry, and Feodor F. Dragan. Overlapping matrix pattern visualization: A hypergraph approach. In *Proceedings of the 8th IEEE International Conference on Data Mining, ICDM '08*, pages 313–322, 2008.

[46] R. A. Johnson and D. W. Wichern, editors. *Applied multivariate statistical analysis*. Prentice-Hall, Inc., 1988.

[47] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley, 1990.

[48] Daniel A. Keim, Gennady Andrienko, Jean-Daniel Fekete, Carsten Görg, Jörn Kohlhammer, and Guy Melançon. Visual analytics: Definition, process, and challenges. In *Information Visualization: Human-Centered Issues and Perspectives*, pages 154–175, 2008.

[49] Edward Kennedy and Nancy Kassebaum. Us health insurance portability and accountability act (hipaa). Act of the United States Congress in 1996.

[50] A. Kern, M. Kuhlmann, A. Schaad, and J. Moffett. Observations on the role life-cycle in the context of enterprise security management. In *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies, SACMAT '02*, pages 43–51, 2002.

[51] Hyunsoo Kim, Gene H. Golub, and Haesun Park. Missing value estimation for DNA microarray gene expression data: local least squares imputation. *Bioinformatics*, 21(2):187–198, 2005.

[52] Martin Kuhlmann, Dalia Shohat, and Gerhard Schimpf. Role mining – revealing business roles for security administration using data mining technology. In *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies, SACMAT '03*, pages 179–186, 2003.

[53] Ninghui Li, Ji-Won Byun, and Elisa Bertino. A critique of the ANSI standard on role-based access control. *IEEE Security & Privacy*, 5(6):41–49, 2007.

[54] Jinze Liu, Susan Paulsen, Xing Sun, Wei Wang, Andrew B. Nobel, and Jan Prins. Mining approximate frequent itemsets in the presence of noise: Algorithm and analysis. In *Proceedings of the 6th SIAM International Conference on Data Mining*, pages 405–416, 2006.

[55] Jinze Liu, Susan Paulsen, Wei Wang, Andrew Nobel, and Jan Prins. Mining approximate frequent itemsets from noisy data. In *Proceedings of the 5th IEEE International Conference on Data Mining, ICDM '05*, pages 721–724, 2005.

[56] Haibing Lu, Jaideep Vaidya, and Vijayalakshmi Atluri. Optimal boolean matrix decomposition: Application to role engineering. In *Proceedings of the 24th IEEE International Conferene on Data Engineering, ICDE '08*, pages 297–306, 2008.

[57] Mohamed A. Mahfouz and M. A. Ismail. BIDENS: Iterative density based biclustering algorithm with application to gene expression analysis. In *Proceedings of World Academy of Science, Engineering and Technology, PWASET*, volume 37, pages 342–348, 2009.

[58] Ian Molloy, Hong Chen, Tiancheng Li, Qihua Wang, Ninghui Li, Elisa Bertino, Seraphin Calo, and Jorge Lobo. Mining roles with semantic meanings. In *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies, SACMAT '08*, pages 21–30, 2008.

[59] Ian Molloy, Ninghui Li, Tiancheng Li, Ziqing Mao, Qihua Wang, and Jorge Lobo. Evaluating role mining algorithms. In *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies, SACMAT '09*, pages 95–104, 2009.

[60] Ian Molloy, Ninghui Li, Yuan (Alan) Qi, Jorge Lobo, and Luke Dickens. Mining roles with noisy data. *Proceeding of the 15th ACM symposium on Access control models and technologies - SACMAT '10*, page 45, 2010.

[61] Gustaf Neumann and Mark Strembeck. A scenario-driven role engineering process for functional RBAC roles. In *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies, SACMAT '02*, pages 33–42, 2002.

[62] Shigeyuki Oba, Masa-Aki Sato, Ichiro Takemasa, Morito Monden, Ken-Ichi Matsubara, and Shin Ishii. A bayesian missing value estimation method for gene expression profile data. *Bioinformatics*, 19(16):2088–2096, November 2003.

[63] Alan C. O'Connor and Ross J. Loomis. 2010 economic analysis of role-based access control. Technical report, National Institute of Standards and Technology (NIST), 2010.

[64] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. *SIGMOD Rec.*, 29(2):427–438, 2000.

[65] H. Röckle, G. Schimpf, and R. Weidinger. Process-oriented approach for role-finding to implement role-based security administration in a large industrial organization. In *Proceedings of the 5th ACM Workshop on Role-Based Access Control, RBAC 2000*, volume 3, pages 103–110, 2000.

[66] Donald B. Rubin. *Multiple imputation for nonresponse in surveys*. Wiley, 1987.

[67] Ron Rymon. Method and apparatus for role grouping by shared resource utilization, September 2003. United States Patent Application 20030172161.

[68] Paul Spyros Sarbanes and Michael Garver Oxley. Sarbanes-Oxley Act of 2002. United States federal law. Pub. L. No. 107-204, 116 Stat. 745. Also known as the "Public Company Accounting Reform and Investor Protection Act of 2002".

[69] J urgen Schlegelmilch and Ulrike Steffens. Role mining with ORCA. In *Proceedings of the 10th ACM Symposium on Access Control Models and Technologies, SACMAT '05*, pages 168–176, 2005.

[70] Jürgen Schlegelmilch and Ulrike Steffens. Role mining with ORCA. In *Proceedings of the tenth ACM symposium on Access control models and technologies - SACMAT '05*, page 168, New York, New York, USA, June 2005. ACM Press.

[71] Dongwan Shin, Gail-Joon Ahn, Sangrae Cho, and Seunghun Jin. On modeling system-centric information for role engineering. In *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies, SACMAT '03*, pages 169–178, 2003.

[72] Daluss J. Siewert. *Biclique Covers and Partitions of Bipartite Graphs and Digraphs and Related Matrix Ranks of* $\{0,1\}$ *Matrices*. PhD thesis, The University of Colorado at Denver, 2000.

[73] Olga G. Troyanskaya, Michael Cantor, Gavin Sherlock, Patrick O. Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B. Altman. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17(6):520–525, 2001.

[74] Johannes Tuikkala, Laura L. Elo, Olli S. Nevalainen, and Tero Aittokallio. Missing value imputation improves clustering and interpretation of gene expression microarray data. *BMC Bioinformatics*, 9(202), 2008.

[75] Jaideep Vaidya, Vijayalakshmi Atluri, and Qi Guo. The role mining problem: finding a minimal descriptive set of roles. In *Proceedings of the 12th ACM Symposium on Access Control Models and Technologies, SACMAT '07*, pages 175–184, 2007.

[76] 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.

[77] Jaideep Vaidya, Vijayalakshmi Atluri, Qi Guo, and Haibing Lu. Role mining in the presence of noise. In *Proceedings of the 24th annual IFIP WG 11.3 working conference on Data and applications security and privacy*, pages 97–112, Rome, Italy, 2010. Springer.

[78] Jaideep Vaidya, Vijayalakshmi Atluri, and Janice Warner. RoleMiner: mining roles using subset enumeration. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS '06*, pages 144–153, 2006.

[79] Mark van der Laan, Katherine Pollard, and Jennifer Bryan. A new partitioning around medoids algorithm. *Journal of Statistical Computation and Simulation*, 73(8):575–584, 2003.

[80] Nino Vincenzo Verde, Jaideep Vaidya, Vijay Atluri, and Alessandro Colantonio. Role mining: From theory to practice. In *Proceedings of Second ACM Conference on Data and Application Security and Privacy, CODASPY 2012, San Antonio, TX, USA, February 21-23*. ACM, 2012.

[81] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[82] Mohammed J. Zaki and Ching-Jui Hsiao. Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 17(4):462–478, 2005.

[83] Mohammed J. Zaki and Ching-Jui Hsiao. Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 17(4):462–478, 2005.

[84] Mohammed J. Zaki and Mitsunori Ogihara. Theoretical foundations of association rules. In *In 3rd ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 1998.

[85] Dana Zhang, Kotagiri Ramamohanarao, and Tim Ebringer. Role engineering using graph optimisation. In *Proceedings of the 12th ACM Symposium on Access Control Models and Technologies, SACMAT '07*, pages 139–144, 2007.

[86] Nan Zhang, Mark Ryan, and Dimitar P. Guelev. Synthesising verified access control systems through model checking. *J. Comput. Secur.*, 16(1):1–61, 2008. IOS Press.