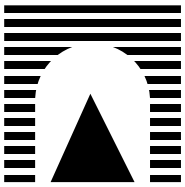


UNIVERSITÀ DEGLI STUDI "ROMA TRE"
FACOLTÀ DI SCIENZE MATEMATICHE FISICHE E NATURALI
XXIII Ciclo del Dottorato di Ricerca in Matematica

Role Mining Techniques To Improve RBAC Administration

Candidato
Alessandro COLANTONIO

Tutor
Roberto DI PIETRO

Coordinatore
Luigi CHIERCHIA

A.A. 2010/2011

To my daughter, amazing, lovely, and true.

To my wife, for her loving concerns.

To my brother, who always held me in high esteem.

To my parents, to whom I owe the greatest debt.

Abstract

Access control is currently one of the most important topics in ICT security. The main areas of research related to access control concern the identification of methodologies and models to efficiently administer user entitlements. With the ever-increasing number of users and IT systems, organizations have to manage large numbers users' permissions in an efficient manner. *Role-based access control* (RBAC) is the most wide-spread access control model. Yet, companies still find it difficult to adopt RBAC because of the complexity of identifying a suitable set of roles. Roles must accurately reflect functions and responsibilities of users in the organization. When hundreds or thousands of users have individual access permissions, adopting the best approach to engineer roles saves time and money, and protects data and systems. Among all *role engineering* approaches, searching legacy access control systems to find *de facto* roles embedded in existing permissions is attracting an increasing interest. Data mining techniques can be used to automatically propose candidate roles, leading to a class of tools and methodologies referred to as *role mining*.

This thesis is devoted to role mining techniques that help security analysts and administrators maximize the benefits of adopting RBAC. To this aim, we consider the role mining problem from several viewpoints. First, we propose a *cost-driven approach* to identify candidate roles. This approach measures and evaluates cost advantages during the entire role-set definition process. This allows to easily integrate existing bottom-up approaches to role engineering with top-down information. Second, we provide a new formal framework to *optimize role mining algorithms*. Applying this framework to real data sets consistently reduces running time and often improves output quality. Another key problem that has not previously been adequately addressed is how to automatically propose *roles that have business meaning*. To do this, we provide a formal framework that leverages business information, such as business processes and organization structure, to implement role mining algorithms. Fur-

thermore, we address the problem of reducing the role mining complexity in RBAC systems by *removing "noise" from data*; i.e., permissions exceptionally or accidentally granted or denied. We propose a new methodology to elicit stable candidate roles, by contextually simplifying the role selection task. Finally, we address the problem of effectively managing the *risk associated with granting access* to resources. We propose a new divide-and-conquer approach to role mining that facilitates attributing business meaning to automatically elicited roles and reduces the problem complexity.

Each of the above results is rooted on a sound theoretical framework and supported by extensive experiments on real data.

Acknowledgements

This thesis is the outcome of a magnificent as well as challenging experience. Looking back, I am surprised and at the same time very grateful for all I have received throughout these years. It has certainly shaped me as a person and has led me where I am now. Moreover, many people were instrumental directly or indirectly in reaching the end of this path. It was hardly possible for me to thrive in my doctoral work without the precious support of these personalities. Here is a small tribute to all those people.

My first thoughts necessarily go to my advisor, Roberto Di Pietro. He introduced me to the charming world of computer security when I attended the Master in IT Security and Management at “La Sapienza” University, in 2006. After proposing this exciting experience, he always guided me in the right direction. His encouragement and valuable advises made me feel confident to fulfill my desire and to overcome every difficulty I encountered. I have learned a lot from him, without his help I could not have concluded this work successfully. Anyhow, it is not sufficient to express my gratitude with only a few words. Roberto is great to work with and I sincerely hope to continue collaborating with him for a long time.

Special thanks are also given to Alberto Ocello, general director of Engiweb Security—the company I work for. He gave me the possibility to enjoy my doctoral studies in parallel with my ordinary work. He also provided me all the necessary tools. Most of my ideas were born from stimulating technical discussions with him. His remarkable insights into identity and access management systems were absolutely precious. I owe him all my gratitude for always believing in me since the beginning.

There are two people I need to mention especially: Nino Vincenzo Verde and Riccardo Mariani. This work largely benefited from my collaboration with them. Their friendship and unselfish help have made this experience a great

pleasure. I owe them both my sincere gratitude.

I would also like to convey my heartfelt thanks to all my colleagues of Engiweb Security, for offering me an ideal environment in which I felt free and could concentrate on my research. They demonstrated a sincere friendship in so manifold occasions. I give my sincere thanks to all these people.

I am very grateful for my family and, in particular, my parents. Their understanding and love always encouraged me to work hard. Their firm and kind-hearted personality has affected me to be steadfast and never bend to difficulty. They always lets me know that they are proud of me, which motivates me to work harder and do my best.

Finally, I owe my gratitude and love to my wife Rosa and my daughter Vittoria. They have lost plenty of my time due to my research. Conversely, the time they spared for me was just precious.

Roma, December 2010

Alessandro Colantonio

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

In computer security, *access control* represents the process of mediating requests to data and services, and determining whether the requests should be granted or denied [32]. In recent years significant research has focused on providing formal representations of access control models [104]. In this context, *role-based access control* (RBAC) [19] has become the norm for managing entitlements within commercial applications. RBAC simplifies entitlement management by using *roles*. A role uniquely identifies a set of permissions, and users are assigned to appropriate roles based on their responsibilities and qualifications. When users change their job function, they are assigned new roles and old roles are removed from their profile. This results in users' entitlements matching their actual job functions. While RBAC is not a panacea for all ills related to access control, it offers great benefits to users managers and administrators, especially non-technical people. First, RBAC helps business users define security policies [45]. Second, RBAC implements the security engineering principles that support risk reduction, such as separation of duties (SoD) and least privilege [26]. Finally, roles minimize system administration effort by reducing the number of relationships among users and permissions [5].

Despite the widespread adoption of RBAC-oriented systems, organizations frequently implement them without due consideration of roles. To minimize deployment effort or to avoid project scope creep, organizations often neglect role definition in the initial part of the deployment project. Very often, organizations do not invest enough time to define roles in detail; rather, they define high-level roles that do not reflect actual business requirements. The result of this careless role definition process is that deployed RBAC systems do not deliver the expected benefits. Additionally, it also leads to role misuse [26]. This is the main reason why many organizations are still reluctant to adopt RBAC. The *role engineering* discipline [28] addresses these problems. Its aim is to

properly customize RBAC systems in order to capture the needs and functions of the organizations. Yet, choosing the best way to design a proper set of roles is still an open problem. Various approaches to role engineering have been proposed, which are usually classified as: *top-down* and *bottom-up*. Top-down requires a deep analysis of business processes to identify which access permissions are necessary to carry out specific tasks. Bottom-up 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. In practice, top-down approaches may produce results that conflict with existing permissions, while bottom-up approaches may not consider the high-level business structure of an organization [54]. For maximum benefit, therefore, a *hybrid* of top-down and bottom-up is often the most valid approach.

The bottom-up approach has recently attracted many researchers, since it can be easily automated [40, 69]. Indeed, companies which plan to deploy RBAC-oriented systems usually find themselves migrating from several "conventional" access control systems [58]. Thus, role mining uses data mining techniques to generate roles from the access control information of this collection of legacy and standard systems. Current role mining approaches, however, must deal with some practical issues:

Meaning of Roles Automatically elicited roles often have no connection to business practice [5]. Existing role mining algorithms can be classified in two different classes [69], both of which suffer from the same problem. The first class seeks to identify complete RBAC states, minimizing the resulting system complexity [3, 41, 68, 92]. It is dubious, however, that automated techniques can overcome and replace the cognitive capacity of humans. This is particularly true when complex security policies also allow for exceptions. As a result, organizations are unwilling to deploy automatically elicited roles that they cannot fully understand or trust. To gain greater flexibility, a second class of algorithms proposes a complete list of roles [4, 94], so role designers can manually select the most relevant ones. However, there is the risk of missing the complete view of data due to the typically large number of candidate roles and unavoidable exceptions.

Algorithm Performance 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 [6, 7, 98] to name a few. As a result, many role mining algorithms entail long running times and huge memory footprints. Moreover, when hundreds of thousands of existing user-permission assignments need to

be analyzed, the number of candidate roles may be so high that trying to analyze them is often impractical.

Noise Within Data The number of elicited roles is often very large mainly due to "noise" within the data—namely, permissions exceptionally or accidentally granted or denied. In such a case, classical role mining algorithms elicit multiple small fragments of the true role [49]. Recently, a number of methods have been proposed to discover approximate patterns in the presence of noise [39, 49]. However, they usually require tuning several parameters that can greatly affect algorithm performance and the quality of results. Another problem is that the adopted noise model may not fit real cases, mainly when exceptions are legitimate and cannot be avoided. Further, the number of exceptions may be so high to make it difficult to navigate and analyze them.

Problem Complexity Introducing new users, permissions, or relationships between them into the access control system may require reassessing the role-set in use. In other words, the RBAC system could require a complete re-design of its roles in order to reduce the overall administration cost. Another important observation is related to the typically large number of candidate roles, which hampers the selection of the most meaningful ones for the organization. Furthermore, matching top-down information with bottom-up results is often impracticable.

Risk of Unmanageable Roles As stated before, a superficial application of standard data mining approaches often yields roles that are merely a set of permissions, with no connection to the business practices. It is difficult to incorporate such roles into an organization's risk management framework. Thus, poorly designed roles directly increase the risk of incorrectly authorized users [26].

1.1 Candidate's Contribution

This thesis collects and harmonizes all the contributions of the candidate to address the aforementioned issues. Such contributions are witnessed by several conference and journal papers. These publications are summarized in Table 1.1. For each paper, the table indicates: the bibliographical reference that gives full details of the conference/journal, the paper title, an outline of the contribution, and the chapters of this thesis to delve into work details. In the bibliography at the end of the document, the authors of papers cited by

Table 1.1 Summary of Candidate's Contribution

Ref.	Title	Contribution	Ch.
[3]	*A Cost-Driven Approach to Role Engineering*	The cost-function to measure the system complexity, as well as a role mining algorithm to use it	3
[4]	*Leveraging Lattices to Improve Role Mining*	A formal framework to improve the performance of role mining algorithms such as [3]	4
[7]	*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	4
[5]	*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 [3]	5
[10]	*Mining Business-Relevant RBAC States Through Decomposition*	An entropy-based measure of the expected uncertainty in locating homogeneous users and permissions and its applications	5
[6]	*Mining Stable Roles in RBAC*	Easing the mining task by discarding user-permission assignments that lead to "unstable" roles	6
[12]*	*Taming Role Mining Complexity in RBAC*	Efficient algorithms to implement the approach proposed in [6]	6
[8]	*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	6
[11]*	*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	7
[9]	*Evaluating the Risk of Adopting RBAC Roles*	A framework to rank users and permissions by the risk related to markedly deviating from "peers"	7
[1]*	*CONCISE: COmpressed 'N' Composable Integer SEt*	Compressed data structure to increase the performance of role mining algorithms, to use with [3]	–
[13]*†	*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	–
[14]*†	*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	–
[15]*†	*Privacy Preserving Role-Engineering*	An approach for a privacy-preserving outsourcing of the role engineering task	–
[2]†	*An Activity-Based Model for Separation of Duty*	Activity model to define SoD constraints, also used to model business processes in [5]	–

*Journal paper
†Unpublished and under review process

Table 1.1 are reported in alphabetical order, but the candidate is actually the first author for the majority of papers, and the second one for the remaining.

More details about the cited work follows. First, in [3] we tackle the problem of designing meaningful roles by providing a *cost-driven approach* to role engineering. The main idea is to identify candidate roles based on the measurement and evaluation of "cost" advantages during the entire role-finding process. This allows to effectively integrate existing bottom-up approaches to role engineering with top-down information to match organization needs. In particular, the cost concept allows to evaluate the good quality of roles by leveraging the available modeling of business. Additionally, in [5] we describe how to design a cost function that measures the meaning of roles. This is done by evaluating the *spreading* of roles among business processes or organization structure. The business process model is mainly inspired by [2], a work that, while being an additional candidate's contribution, is not strictly related to role engineering topic. Hence, it is not further detailed in this thesis.

Second, the candidate devised another solution for the problem of eliciting meaningful roles. This contribution is complementary to the previous one. Both [11] and [10] 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. Partitioning data also introduces benefits in terms of execution time of algorithms, hence providing a solution to the complexity problem mentioned in the previous section. To apply this divide-and-conquer strategy, in [11] we describe two indices that measure the expected complexity to find roles with clear business meaning. Alternatively, in [10] we describe an 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.

Third, the candidate addressed the performance problem of role mining algorithms. In [4] a new formal framework to *optimize role mining algorithms* is proposed. In particular, we describe how to efficiently enumerate interesting patterns among user-permission assignments by discarding redundant information. We demonstrate that, when applied to real data sets, it consistently reduces the running time of certain role mining algorithms and, in some cases, also greatly improves output quality. In addition to this fundamental result, in [7] we address the issue of *minimizing the number of roles*. In this case, our main contribution is to prove that the minimum number of roles is sharply concentrated around its expected value. This result can be applied as a stop

condition when striving to minimize the number of roles through fast approximation algorithms. The proposal can also be used to decide whether it is advisable to undertake the efforts to renew a RBAC state. To further improve the performance of role mining algorithms, in [1] we describe a novel compression scheme that allows for fast set operations (i.e., intersection, union, difference) over a compressed representations of integral numbers. Hence, reducing the memory required to store data. We demonstrate that this scheme is more efficient than other well-known data structures such as arrays, lists, hash tables, and self-balancing binary search trees. Since this contribution is not strictly related to the role engineering problem, we will not further describe it in this thesis—as indicated in Table 1.1.

As for the identification of *exceptionally/accidentally granted/denied permissions*, in [6] we propose a new methodology that allows role engineers to elicit "stable" candidate roles, namely roles that likely remains unchanged during their lifecycle. To obtain this result, we offer a theoretical framework that allows to identify and then discard assignments that can only be managed via "unstable" roles. Then in [12] we introduce fast algorithms that implement such a strategy. Hence, avoiding the generation of unstable roles during the application of any role mining algorithm. The dual problem of selecting assignments related to unstable roles is represented by the identification of *missing* assignments: those permissions that, if granted to certain users, would simplify the mining task. In [8] we describe a viable and effective approach to identify such missing values. Furthermore, [14] gather all the contributions related to exception identification in access control by offering a holistic approach to this problem. Since [14] has been submitted for publication to a journal and the corresponding review process is not yet complete, we will not further describe this contribution in the thesis.

Finally, we confront the question of effectively managing the *risk derived from granting access* to resources. In particular, [9] describes a framework to highlight users and permissions that markedly deviate from those "similar" to them. That is, bringing out users that have a permission-set different from other users with the same business attributes (i.e., job title, department, cost center, etc.) since they are likely prone to error when roles are operating. Similarly, permissions that shares the same attributes (e.g., application, supported business activity, etc.) granted to different user-sets are reported. Focusing on such users and permissions during the role definition process likely mitigates the risk of unauthorized accesses and role misuse. Moreover, in [11] we discuss how a divide-and-conquer approach to role mining allows for better enforcement and actually reduces the risk related to illegal accesses. In particular, we describe a risk model that assesses the risk related to entitling users to not expected operations, or hampering their jobs by not granting required

permissions. The likelihood of an administration error is estimated through two indices provided in the paper.

As reported in Table 1.1, there are two additional papers that have been submitted for publication to journals. Since the review processes are not yet complete, we will just outline the contribution without further describing them in the reminder of this thesis. The first one is [13], where a new approach referred to as *visual role mining* is delineated. The key idea is that of adopting a visual representation of existing user-permission assignments, hence allowing for a quick analysis and elicitation of meaningful roles. We formally introduce the problem by defining a metric for the quality of the visualization. We also prove that finding the best representation according to the defined metric is $\mathcal{NP}$-hard. In turn, we propose novel algorithms that seek to best represent user-permission relationships. The second paper is represented by [15]. We face up the problem of outsourcing the role engineering task to consultants external to the analyzed organization, while preserving the privacy of the organizations itself. In particular, we demonstrate that it is possible to use specific data encryption techniques to guarantee the confidentiality of data without compromising the ability of conducting the role engineering task.

1.2 Outline of the Thesis

The remainder of the thesis is organized as follows. Chapter 2 offers some preliminaries required for the subsequent chapters. In particular, the basics of RBAC are first provided. The chapter also offers a formal description of the RBAC model through the ANSI/INCITS 359-2004 standard. Subsequently, we briefly survey the state-of-the-art in role engineering. In particular, the chapter also introduces the automated part of role engineering, namely *role mining* techniques. Binary matrices are introduced as a tool to formally describe the role mining problem. The identification of exceptional values in binary matrices is also discussed.

Chapter 3 fills a gap of several role engineering methods, by providing a metric to measure the "quality" of candidate roles. We introduce the concept of measuring the "cost" of RBAC administration. Further, we describe an algorithm that leverages the cost metric to find candidate role-sets with the lowest possible administration cost. We provide several examples showing the sensibility of assumptions made by the algorithm.

Chapter 4 offers a rigorous analysis of identifiable patterns in access permission data. We show that this analysis can practically be applied to optimize role mining algorithms. Moreover, we prove that the minimum number of roles is sharply concentrated around its expected value. We show how this

result can be used to decide whether it is advisable to undertake the efforts to renew an RBAC state.

Chapter 5 copes with the problem of assigning a *business meaning* to roles. To this aim, two different approaches are detailed. First, we introduce a new metric to assess how "good" are roles from a business perspective, by measuring the *spreading* of a role among business processes or the organization structure. Second, we propose a methodology that helps analysts decompose the dataset into smaller subsets homogeneous from a business perspective. By using practical examples, we demonstrate that this approach likely leads to roles with a clear business meaning.

Chapter 6 addresses the problem of reducing the role mining *complexity* in RBAC systems. The chapter introduces the meaning of *exceptional* assignments. We also propose a three-steps methodology to identify and discard user-permission assignments that cannot belong to so-called "stable" roles. Furthermore, we also consider the possibility that analyzed data present some *missing* user-permission assignment. Thus, we propose an algorithm that to efficiently impute such missing values.

Chapter 7 describes a risk model to analyze the proneness of RBAC configurations to allow for unauthorized access to resources. To measure the likelihood of having meaningful and manageable roles within the system, we recall the methodology described in Chapter 5 where data to analyze are decomposed into smaller subsets according to the provided business information. We show how this processing decreases the probability of making errors in role management, and consequently reduces the risk of role misuse. Additionally, we introduce an approach to highlight users and permissions that markedly deviate from others, and that might consequently be prone to error when roles are operating.

To conclude, Chapter 8 offers final remarks.

2 Background and Related Work

The purpose of this chapter is to acquaint the reader with the prerequisite knowledge and required mathematical formalism. It starts with a brief overview of RBAC and role-based concepts, as well as a formal description according to the ANSI/INCITS 359-2004 standard. 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. The chapter also introduces the automated part of role engineering, namely *role mining* techniques. To conclude, *binary matrices* are introduces by showing the relationship with role mining. A short survey of existing methodologies for exception identification in binary matrices is also presented.

2.1 Role-Based Access Control

In ICT security, we refer to *access control* as the process of mediating requests to data and services maintained by a system, and determining whether the requests should be granted or denied [32]. 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 [82], US Health Insurance Portability and Accountability Act (HIPAA) [53], Gramm-Leach-Biley Act (GLBA) [48], and EU Privacy Protection Directive 2002/58/EC [47], 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.

Role-based access control (RBAC) [19] 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 [19] 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 [31] 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

[1] The RBAC chapters that mention role hierarchy most often treat it as a partial order. By maintaining only a partial order it is not possible to distinguish role dominance relationships explicitly added from those implied [59]. Since consensus (on this matter) has yet to be reached among researchers, we only consider hierarchical relationships derived from permission-set inclusion.

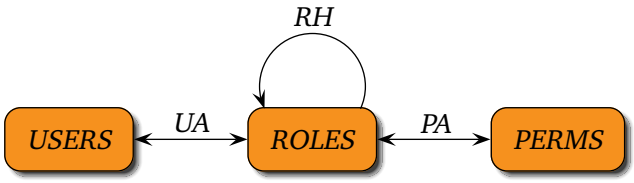


Figure 2.1 RBAC entities

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 to as an *immediate junior* of r_1. In Chapter 4 we will extensively use the proposed mathematical formalism. We will also use other well-known concepts such as the partial order, lattice, powerset, Hasse diagrams, and directed acyclic graphs. Further details on these subjects can be found in [31].

The following functions are also provided by the ANSI standard:

- *ass_users*: $ROLES \rightarrow 2^{USERS}$ to identify users assigned to a role and to none of its senior roles.[2]
- *auth_users*: $ROLES \rightarrow 2^{USERS}$ to identify users assigned to a role or to at least one of its seniors.
- *ass_perms*: $ROLES \rightarrow 2^{PERMS}$ to identify permissions assigned to a role and to none of its senior roles.[3]
- *auth_perms*: $ROLES \rightarrow 2^{PERMS}$ to identify permissions assigned to a role or to at least one of its seniors.

[2]The RBAC standard does not make a clear distinction between base and derived relations [59]. We therefore consider the functions *ass_users* as derived from *UA*, that is $ass\_users(r) = \{u \in USERS \mid \langle u, r\rangle \in UA\}$. We also assume that users assigned to a role are not assigned to its seniors.

[3]Analogous to *ass_users*, we consider the function *ass_perms* as derived from *PA*, that is $ass\_perms(r) = \{p \in PERMS \mid \langle p, r\rangle \in PA\}$. We also assume that permissions assigned to a role are not assigned to its juniors.

The following relation holds true:

$$\forall r_1, r_2 \in ROLES : r_1 \succeq r_2 \implies auth\_users(r_1) \subseteq auth\_users(r_2) \wedge auth\_perms(r_1) \supseteq auth\_perms(r_2). \quad (2.1)$$

2.2 Role Engineering

As organizations start deploying RBAC-oriented access control solutions, it is becoming increasingly important to devise a common set of roles that can be reused and easily maintained over the time. One of the challenges often faced is that, if defined incorrectly, roles are ineffective and fail to meet the organization's requirements. Roles can be defined at an abstract level from a business perspective, or context-specific to an application or system from a technology perspective. At an abstract level, a role can be a simple "label" that defines the corresponding job function and the set of business activities that users have to perform to accomplish their responsibilities. Notice that at an abstract level, there is no enforcement capability. Each role has specific entitlements that enable a user to execute transactions with certain limits. How this is configured within the application and how it is enforced is specific to the individual application capability. Whether an organization looks at defining roles either abstract or specific to a context, the requirements to define roles are important and role definition is a critical step in deploying any RBAC system.

Role engineering [28, 29] is the process of defining roles and related information, such as permissions, constraints, and role hierarchies, as they pertain to the user's functional use of systems, applications, and business processes. It is one of the critical steps in deploying role-based access control systems. Organizations often implement RBAC systems without much consideration for roles. To minimize deployment effort, role definition is often not considered as a part of the deployment project. Organizations frequently do not invest enough time to define roles in sufficient detail; rather, they tend to define high-level roles that do not reflect actual organizational job functions. Permissions mapped to high-level roles are usually generic in nature. The result of this "random" process is that additional efforts are required to manage job-specific permissions manually, outside the RBAC system. Hence, without exploiting all the RBAC benefits. The process of defining roles should be based on a complete analysis of the given organization, including the input from a wide spectrum of users such as business line managers and human resources. In general, role definition and management requires *alignment between business and IT*. It requires a strong commitment and cooperation among the business units, as a role engineering initiative could transcend the enterprise.

Choosing the best role engineering approach is still an open problem. Various approaches can be found in the current literature, 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* [58] is often used as a synonym for bottom-up. Both top-down and bottom-up approaches have pros and cons. To maximize benefits, bottom-up should be used in conjunction with top-down, leading to an *hybrid* approach. As a matter of fact, top-down may ignore existing permissions and exceptions, whereas bottom-up may not consider the business functions of an organization [54]. Hence, hybrid approaches leverage normalized roles derived from role mining and align them to job functions, with the involvement of the business staff. In the following, we summarize the state-of-the-art for both top-down and bottom-up approaches.

Top-Down This 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 mapping the permissions for each job function. This approach provides business oversight and alignment of roles with business functions and reusability.

Top-down role engineering was first illustrated by Coyne [28]. 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 [37] propose a similar approach where use-cases are used to determine the needed permissions. Röckle et al. [78] 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. [30] leverage organizational theory to elicit role-based security policies. Neumann and Strembeck [70] 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. [87] 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 [35] also use UML to address role engineering. Kern et al. [54] 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 [29] is a practical reference that helps to assess some of the previously cited role engineering approaches.

Bottom-Up This 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. [58] 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 particular 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 [17] 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 [86] which applies hierarchical clustering techniques on permissions. However, this approach does not allow for permission overlapping among roles that are not hierarchically related. Vaidya at al. [94] 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 [81]. More recently, the same authors of [94] also studied the problem of finding the minimum number of roles that cover all permissions possessed by users [92, 93]. By leveraging binary integer programming, Lu et al. [63] presented a unified frame-

work for modeling the role number minimization problem. Ene et al. [34] 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. [103] provide an attempt to contextually minimize the number of user-role, permission-role, and role-role relationships. Frank et al. [39] 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 6) to cite a few. Recently, Frank et al. [40] 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.

The main limitation of all the cited role mining approaches is that they do not always lead to the optimal set of roles from a business perspective. To the best of our knowledge, the work described in thesis and first introduced in [3] represents the first 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. [68]. 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.

2.3 Candidate Role-Sets

In addition to RBAC concepts, this thesis introduces other entities required to formally describe the proposed approach. In particular, we define:

- $UP \subseteq USERS \times PERMS$, the set of the existing user-permission assignments to be analyzed;
- $perms$: $USERS \rightarrow 2^{PERMS}$, 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 $users(p) = \{u \in USERS \mid \langle u, p \rangle \in UP\}$.

After having introduced the entities above, we can formally define the main objective of role mining. In particular, the outcome of any role mining tool is a set of *candidate roles*, that is roles which requires an attestation by business people, according to the observations of the previous section. More formally:

Definition 2.1 (System 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 is the users' authorization state before migrating to RBAC, or the authorizations derivable from the current RBAC implementation.

Definition 2.2 (RBAC State) An RBAC *state* is represented by tuple $\psi = \langle ROLES, UA, PA, RH \rangle$, namely an instance of all the sets that characterize the RBAC model.

An RBAC state is used to obtain a system configuration. Hence, the role engineering goal is to find a state that correctly describes a given configuration. In particular we are interested in the following:

Definition 2.3 (Candidate Role-Set) Given a system configuration φ, a *candidate role-set* is the set of roles of a given 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:

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

Please also note that every system configuration may allow for multiple candidate role sets. In Chapter 3 we will explain how to select the "best" one according to organization requirements.

Definition 2.4 (Support of a Role) Given a role $r \in ROLES$, its *support* indicates the percentage of users possessing all permissions assigned to that role, that is $support(r) = |auth\_users(r)|/|USERS|$.

Definition 2.5 (Degree of a Role) The *degree* of a candidate role indicates the number of permissions assigned to it, that is $degree(r) = |auth\_perms(r)|$.

Definition 2.6 (Confidence Between Roles) Given a pair of hierarchically related candidate roles, *confidence* is the percentage of users possessing permissions assigned to both parent and child roles, that is $confidence(r_2 \succeq r_1) = |auth\_users(r_2)|/|auth\_users(r_1)|$.

Lemma 2.1 *Given $r_1, r_2 \in ROLES$ such that $r_2 \succeq r_1$, the confidence between r_1, r_2 is given by the ratio between supports of child and parent roles:*

$$confidence(r_2 \succeq r_1) = support(r_2)/support(r_1).$$

PROOF By definition, $confidence(r_2 \succeq r_1)$ is equal to:

$$\frac{|auth\_users(r_2)|}{|auth\_users(r_1)|} \cdot \frac{|USERS|}{|USERS|} = \frac{support(r_2)}{support(r_1)}$$

for any given role pair r_1, r_2. ■

Lemma 2.2 *Given $r_1, r_2, \ldots, r_{n-1}, r_n \in ROLES$ such that $r_n \succeq r_{n-1} \succeq \ldots \succeq r_2 \succeq r_1$, the confidence between r_1, r_n is equal to the product of confidences between intermediate roles of the given hierarchical path between those two roles:*

$$confidence(r_n \succeq r_1) = \prod_{i=2}^{n} confidence(r_i \succeq r_{i-1}).$$

PROOF It follows for Lemma 2.1 that:

$$\begin{aligned} &confidence(r_n \succeq r_{n-1}) \cdot \ldots \cdot confidence(r_2 \succeq r_1) = \\ &\quad = \frac{support(r_n)}{support(r_{n-1})} \cdot \frac{support(r_{n-1})}{support(r_{n-2})} \cdot \ldots \cdot \frac{support(r_2)}{support(r_1)} = \\ &\quad = \frac{support(r_n)}{support(r_1)} = confidence(r_n \succeq r_1) \end{aligned}$$

for any chosen intermediate roles $r_2, \ldots, r_{n-1}$. ■

Let us consider Equation (2.1). Given two roles $r_1, r_2 \in ROLES$ such that $r_2 \succeq r_1$, if $r_1 \neq r_2$ then the role r_2 adds permissions to role r_1. Instead, the users possessing the permissions assigned to role r_2 can be the same as those possessing the permissions assigned to role r_1. Moreover, we can have the following case:

Definition 2.7 (Role-Role Equivalence) Given the roles $r_1, r_2 \in ROLES$, we say that they are *equivalent*, and indicate this with $r_1 \equiv r_2$, if $auth\_users(r_1) = auth\_users(r_2)$.

Lemma 2.3 *The equivalence relation is transitive, meaning that* $\forall r_1, r_2, r_3 \in ROLES : r_1 \equiv r_2 \wedge r_2 \equiv r_3 \implies r_1 \equiv r_3$.

PROOF According to Definition 2.7, we have that $ass\_users(r_1) = ass\_users(r_2)$ and $ass\_users(r_2) = ass\_users(r_3)$, thus $ass\_users(r_1) = ass\_users(r_3)$. ■

Lemma 2.4 *Given* $r_1, r_2 \in ROLES : r_1 \succeq r_2$, *if* $confidence(r_1 \succeq r_2) = 1$ *then* $r_1 \equiv r_2$.

PROOF From Definition 2.6, $confidence(r_1 \succeq r_2) = 1 \implies |ass\_users(r_1)| = |ass\_users(r_2)|$. Moreover, from Equation (2.1), we have that $ass\_users(r_1) \subseteq ass\_users(r_2) \implies ass\_users(r_1) = ass\_users(r_2)$. ■

In addition to the equivalence between a role pair, referred to as "1:1", it is possible to consider an equivalence relationship between a role and a set of roles, that is "1:n". In particular:

Definition 2.8 (Role-Roleset Equivalence) Given a role $r \in ROLES$ and a set of roles $\{r_1, \ldots, r_n\} \subseteq ROLES$ they are *equivalent*, and thus indicated as $r \equiv \{r_1, \ldots, r_n\}$, when

$$auth\_users(r) = \bigcup_{i=1}^{n} auth\_users(r_i).$$

2.4 Binary Matrices and Exceptions

As described in [63, 92, 103], the problem of finding a candidate role-set can be mathematically expressed as the solution of the equation $C = A \otimes B$, where A, B, C are *binary matrices*, that is

- $C \in \{0,1\}^{m \times n}$, the matrix representation of *UP*, where $m = |USERS|$, $n = |PERMS|$, and $c_{ij} = 1$ when the i^{th} user of *USERS* has the j^{th} permission of *PERMS* granted;
- $A \in \{0,1\}^{m \times k}$, the matrix representation of *UA*, where $m = |USERS|$, $k = |ROLES|$, and $a_{i\ell} = 1$ when the i^{th} user of *USERS* is assigned to the ℓ^{th} role of *ROLES*;
- $B \in \{0,1\}^{k \times n}$, the matrix representation of *PA*, where $k = |ROLES|$, $n = |PERMS|$, and $b_{\ell j} = 1$ when the ℓ^{th} role of *ROLES* has the j^{th} permission of *PERMS* assigned;
- the operator "$\otimes$" is such that $c_{ij} = \bigvee_{\ell=1}^{k} (a_{i\ell} \wedge b_{\ell j})$.

The set RH can be represented by decomposing the matrix B. For instance, if there is a two-level hierarchy of roles, two matrices B' and B'' can be identified such that $B = B' \otimes B''$.

Having multiple candidate role-sets for a given system configuration equals to stating that several values for A and B are solution of the equation $C = A \otimes B$. According to Definition 2.3, the goal is to identify candidate role-sets such that it is always possible to identify a suitable subset of roles whose union of permissions matches exactly with the permissions possessed by each user. This is equivalent to decomposing $A \otimes B$ in order to exactly have the given matrix C.

Besides role mining, binary matrices abound in a large variety of fields: market basket data analysis [16], ecology and paleontology [75], just to cite a few. In particular, they have been largely studied in genetic [21, 56, 88]. *Outliers* and *missing values* are two of the main aspects to consider when analyzing this kind of data. Indeed, data collected for a number of applications can be subject to random noise or measurement error. Missing values are incomplete data, namely portions of data that are unavailable or unobserved. Outliers are values far from the rest of the data according to a given distance measure, that is a value that appears to deviate markedly from other values of the dataset in which it occurs. We generically refer to this "noise" as *suspicious* values. In an access control context, suspicious values are represented by permissions exceptionally or accidentally granted or denied to users. This poses new challenges for the efficient discovery of useful information within noisy data. Especially when dealing with large amount of data, or when time constraints do matter, automatic mechanisms for the recognition of inaccurate values are of utmost importance. Indeed, the presence of noise makes classical data mining algorithms (roles mining included) elicit multiple small fragments of the true, interesting patterns [49]. Chapter 6 describes our contribution to solve the noise isolation problem in an access control scenario. In the following we report on many imputation methods that have been developed to deal with suspicious values in binary matrices.

Missing Values Since the seminal paper of Rubin [79], much work has been done in the statistical field about imputing missing values [60,83]. A complete survey can be found in [84]. In [38], Figueroa *et al.* propose to apply a discrete approach to the clustering problem of DNA array hybridization experiments. The authors first highlight the strength of using the binarized version of a dataset of real numbers that describes microarrays data, and then they define the binary clustering with missing values problem. The target of this problem is to set missing values in such a way as to find a minimum cardinality partition of the rows.

When dealing with missing values, one of the best established method is *multiple imputation* [80], that 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. In Chapter 6 we will describe a multiple imputation method for an access control scenario. To date, several algorithms have been proposed to evaluate missing values. The most frequently used algorithms are probably the *k-nearest neighbors* (KNN) [90], and the *local least squares imputation* [55]. In KNN, missing values are imputed by averaging over the corresponding values of the k-neighboring rows. The metric used to calculate the distance between rows is typically the average Euclidean distance. The problem in this algorithm is that the missing values are imputed by considering a fixed number of rows that have to be chosen depending on the dataset being analyzed. The local least squares imputation method is a regression based estimation method that takes into account the local correlation of data. It is made up of two steps: the first one is to select k genes by the L2-norm or by Pearson correlation coefficients. The second one is regression and estimation, regardless of how the k genes are selected. Unfortunately, also in this case, the parameter k is not adaptive. Another algorithm has been proposed by Oba *et al.* [71], based on [24], that uses the bayesian principal component analysis [77]. Also in this case a fixed parameter comparable with the k of KNN is used, that is the number of principal axes (eigenvectors).

Outliers One possible approach to outlier detection is based on the analysis of the *distribution* of datasets. Outliers are determined according to a given probability distribution. In [99], Yamanishi et al. proposes a Gaussian mixture model to present the normal behaviors of data. The main problem with these kinds of approaches is that the user might simply not have enough knowledge about the underlying data distribution. To tackle this issue, *distance-based* methods have been proposed. A distance-based outlier in a dataset D is a data object with a given percentage of the objects in D having a distance greater than d_{min} away from it. The definition of distance can be extended by considering the distance of a point from its k-nearest neighbor [57, 76], or as the sum of distances from its k-nearest neighbors [20]. It has been shown that dealing with large datasets nested loop algorithms, that in the worst case have quadratic complexity, perform better than other approaches [44]. The algorithm proposed by Bay and Schwabacher in [22] is probably the state-of-the-art distance-based outlier detection algorithm. It is based on a nested loop, but a randomization procedure is used to process non-outlier points relatively quickly.

2.5 Graph Theory

We now summarize some graph-related concepts that are required by chapters 4 and 6. In particular, 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 chapter 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 [102] 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 [34, 63, 68, 94].

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 [63, 89] and tiling database [43] to cite a few. Several role mining approaches leverage these concepts [34, 63, 92].

2.6 Posets, Lattices, Hasse Diagrams, and Graphs

This section introduces some concepts required by Chapter 4. In computer science and mathematics, a *directed acyclic graph* (DAG) is a directed graph with no directed cycles. For any vertex v, there is no non-empty directed path starting and ending on v, thus DAG "flows" in a single direction. Each DAG provides a *partial order* to its vertices. We write $u \succeq v$ when there exists a directed path from v to u. The *transitive closure* is the reachability order "$\succeq$". A *partially ordered set* (or *poset*) formalizes the concept of element ordering [31]. A poset $\langle S, \succeq\rangle$ consists of a set S and a binary relation "$\succeq$" that indicates, for certain element pairs in the set, which element precedes the other. A partial order differs from a total order in that some pairs of elements may not be comparable. The symbol "$\succeq$" often indicates a *non-strict* (or *reflexive*) partial order. A *strict* (or *irreflexive*) partial order "$\succ$" is a binary relation that is irreflexive and transitive, and therefore asymmetric. If "$\succeq$" is a non-strict partial order, then the corresponding strict partial order "$\succ$" is the reflexive reduction given by: $a \succ b \iff a \succeq b \wedge a \neq b$. Conversely, if "$\succ$" is a strict partial order, then the corresponding non-strict partial order "$\succeq$" is the reflexive closure given by: $a \succeq b \iff a \succ b \vee a = b$. An *antichain* of $\langle S, \succeq\rangle$ is a subset $A \subseteq S$ such that $\forall x, y \in A : x \succeq y \implies x = y$. We write $x \parallel y$ if $x \not\succeq y \wedge y \not\succeq x$. A *chain* is a subset $C \subseteq S$ such that $\forall x, y \in C : x \succeq y \vee y \succeq x$. Given a poset $\langle S, \succeq\rangle$, the *down-set* of $x \in S$ is $\downarrow x = \{y \in S \mid x \succeq y\}$, while the *up-set* of $x \in S$ is $\uparrow x = \{y \in S \mid y \succeq x\}$. Given $a \succeq b$, the interval $[a, b]$ is the set of points x satisfying $a \succeq x \wedge x \succeq b$. Similarly, the interval (a, b) is set of points x satisfying $a \succ x \wedge x \succ b$.

The *transitive reduction* of a binary relation R on a set S is the smallest relation R' on S such that the transitive closure of R' is the same as the transitive closure of R. If the transitive closure of R is antisymmetric and finite, then R' is unique. Given a graph where R is the set of arcs and S the set of vertices, its transitive reduction is referred to as its *minimal representation*. The transitive reduction of a finite acyclic graph is unique and algorithms for finding it have the same time complexity as algorithms for transitive closure [18]. A *Hasse*

diagram is a picture of a poset, representing the transitive reduction of the partial order. Each element of S is a vertex. A line from x to y is drawn if $y \succ x$, and there is no z such that $y \succ z \succ x$. In this case, we say y *covers* x, or y is an *immediate successor* of x, also written $y \succcurlyeq x$. A *lattice* is a poset in which every pair of elements has a unique *join* (the least upper bound, or *lub*) and a *meet* (the greatest lower bound, or *glb*). The name "lattice" is suggested by the Hasse diagram depicting it. Given a poset $\langle L, \succeq \rangle$, L is a lattice if $\forall x, y \in L$ the element pair has both a join, denoted by $x \curlyvee y$, and a meet, denoted by $x \curlywedge y$ within L. Let $\langle L, \succeq, \curlyvee, \curlywedge \rangle$ be a lattice. We say that $\langle \Lambda, \succeq, \curlyvee, \curlywedge \rangle : \Lambda \subseteq L$ is a *sublattice* if and only if $\forall x, y \in \Lambda : x \curlyvee y \in \Lambda \ \wedge \ x \curlywedge y \in \Lambda$. In general, we define:

- $\bigvee \Lambda = \{x \in L \mid \forall \ell \in L, \forall \lambda \in \Lambda : \ell \succeq \lambda \implies \ell \succeq x\}$, the join of Λ (lub);
- $\bigwedge \Lambda = \{x \in L \mid \forall \ell \in L, \forall \lambda \in \Lambda : \lambda \succeq \ell \implies x \succeq \ell\}$, the meet of Λ (glb).

In particular, $x \curlyvee y = \bigvee\{x, y\}$ and $x \curlywedge y = \bigwedge\{x, y\}$. Both $\bigvee \Lambda$ and $\bigwedge \Lambda$ are unique.

3 Cost-Driven Role Engineering

The objective of this chapter is to fill a typical gap existing in several role engineering methods, which lack a metric for measuring the "quality" of candidate roles produced. To this aim, we propose a new approach guided by a *cost-based metric*, where "cost" represents the effort to administer the resulting RBAC configuration. Further, we propose RBAM (*Role-Based Association-rule Mining*), an algorithm that leverages the cost metric to find candidate role-sets with the lowest possible administration cost. We provide several examples showing the sensibility of assumptions made by the algorithm. Further, applications of the algorithm to real data highlight the improvements over other solutions. This chapter summarizes the contribution previously published in [3].

3.1 Cost Analysis

As stated in Chapter 1, the RBAC model must be properly customized in order to maximize the its advantages. Moreover, in Section 2.3 we showed that for a given access control configuration, several RBAC states—that is, several candidate role-sets—can be found. Therefore, the objective of any role mining algorithm should be to select the "best" candidate role-set among all possible solutions to the problem. Surprisingly, existing role engineering approaches lack a formal metric to capture the "interest" or "quality" of candidate roles. Although this problem has recently been addressed by other authors, there exist some limitations in the suggested resolutions. For instance, in [103]

only the number of role-user and role-permission relationships are analyzed. [92] restricts the analysis to the number of roles and shows that identifying the optimal solution has exponential complexity. Unfortunately, no heuristic method is proposed to attack the problem.

In the following we propose a *cost-driven approach* for the identification of candidate role-sets based on the measurement and evaluation of cost advantages during the entire role-set definition process.

3.1.1 Problem Description

A cost function is a combination of several *cost elements*, each of them considering a particular business- or IT-related aspect. Among the data available to the organization, it is possible to find information that might either directly influence the required system administration effort (e.g., number of roles, number of role-user relationships to be administered, etc.) or information that might help role engineers assign business meaning to roles (e.g., business processes, organization structure, etc.). Once an organization has identified the relevant data for access control purposes, this data should be "translated" into cost elements and then combined into a cost function. This makes it possible to identify the *optimal* candidate roles which best describes the actual needs of the organization. More specifically, minimizing the cost function can simultaneously optimize the administration effort as well as the business meaning related to the elicited roles. Hence allowing for a hybrid approach to role engineering.

Starting from the concepts introduced in Section 2.3, the proposed approach is founded on the following definitions:

Definition 3.1 (Cost Function) Let Φ, Ψ be respectively the set of all possible system configurations and RBAC states. We define the *cost function* as

$$cost\colon \Phi \times \Psi \rightarrow \mathbb{R}^{+}$$

where $\mathbb{R}^{+}$ indicates positive real numbers including 0. It represents an administration cost estimate for the state ψ used to obtain the configuration φ.

Leveraging the cost metric enables to find candidate role-sets with the lowest effort to administer them:

Definition 3.2 (Optimal Candidate Role-Set) Given a configuration φ, an *optimal candidate role-set* is the corresponding configuration ψ that simultaneously represents a candidate role-set for φ and minimized the cost function $cost(\varphi, \psi)$.

In order to find the optimal candidate role-set, different kinds of cost functions could be proposed. Selecting a cost function that better fits the needs of an organization is further discussed in Chapter 5. In particular, finding the optimal candidate role-set can be seen as a *multi-objective optimization problem* [33]. An optimization problem is multi-objective when there are a number of objective functions that are to be minimized or maximized. Multi-objective optimization often means to trade-off conflicting goals. In a role engineering context, possible objectives are: minimizing the number of roles; minimizing the number of role possessed by each user; maximizing the business meaning of roles. The dependencies among role engineering objectives might be quite complex and not obvious[1]. For example, if on one side $|ROLES|$ decreases, there is a strong chance that more roles will be required to cover all the permissions possessed by users, causing $|UA|$ to increase. On the other hand, if we want to reduce the average number of roles per user, we will need more *ad-personam* roles, then $|ROLES|$ will likely increase.

For the sake of simplicity, in the following we just consider a "flat" RBAC model, in which permission inheritance between roles does not exist. In this case, $auth\_perms(r) = ass\_perms(r)$, while $auth\_users(r) \supseteq ass\_users(r)$. Given this hypothesis, a reasonable cost function could be:

$$f = \alpha|UA| + \beta|PA| + \gamma|ROLES| + \delta \sum_{r \in ROLES} c(r) \tag{3.1}$$

where $\alpha, \beta, \gamma, \delta \geq 0$. A linear combination of the cost factors is only one of many possibilities, even if the simplest. The following analysis, however, can be easily adapted to other types of costs. The function $c\colon ROLES \to \mathbb{R}$ expresses an additional cost related to other business information different from $|ROLES|$, $|UA|$ and $|PA|$. For example, the cost function can be used in a bottom-up approach to discard roles which increment the administration cost of the candidate role-set. In this case, when a complete or partial role design is available from a top-down engineering process, we can avoid deletion of all pre-defined roles representing them with $c(r) \to -\infty$. Thus, creating a hybrid role engineering method. Similarly, we could think about a "blacklist" in which all roles have $c(r) \to +\infty$. This could be useful in implementing separation of duties rules, assigning an infinite cost to combinations of permissions which allow incompatible activities. Another important aspect to be taken

[1]Arguably, reasonable cost metric formulas are difficult to obtain synthetically: they must result from data gathered from real-world RBAC settings, e.g., companies that convert to RBAC and have experience in post-conversion management. If defining a sensible cost function is not feasible for the given company, Section 5.4 describes a more viable solution to crafting meaningful roles. Additionally, [13] offers a new, promising approach to role engineering.

into account via $c(r)$ relates to user attributes. For example, a role exclusively used within a given organizational unit may have a higher cost than a role used across multiple organizational units, as it requires the co-ordination of various user managers. The actual utilization of permissions, derivable from system log analysis, can also influence the administration cost. Furthermore, permission validity is often time limited [23]. Permanently and temporarily assigned permission could be distinguished during the role mining process by $c(r) \to +\infty$ for those roles containing permissions with a set expiration time for some user. Alternatively, a lesser value of $c(r)$ could be given to roles with assigned profiles of longer duration. Finally, when hierarchical RBAC is adopted, $c(r)$ could take into account the number of hierarchical associations.

The following section describes how to identify possible complete candidate role-sets starting from the set *UP*, analyzing how the cost changes as roles are deleted.

3.1.2 Permission Lattice and Roles

Suppose we define a role for each possible permission combination, that is, a candidate role-set based on the *PERMS* lattice. Such a set is interesting in that it is a superset of every possible candidate role-set.

The administration cost of the role-set built upon the *PERMS* lattice is neither a maximum nor a minimum of the cost function. In fact, it is possible to increase the cost by increasing the number of role-user relationships. For example, let $PERMS = \{1, 2, 3\}$ so that $ROLES = \{\{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$. If the role $\{1, 2, 3\}$ is removed from *ROLES*, a combination of the remaining candidate roles must be used to cover its permissions, such as $\{1, 2\}$ and $\{1, 3\}$. This doubles the number of relationships in *UA*. Depending on α, β, γ, δ, $c(r)$ and the number of users assigned to $\{1, 2, 3\}$, this could increase the cost even if *ROLES* and *PA* are smaller. Moreover, the cost is greater than the optimal. In fact, if we delete all roles representing combinations of permissions not possessed by any user, the cardinality of *ROLES* and *PA* diminishes while *UA* remains the same. If $c(r) \geq 0$, the cost diminishes as well.

Therefore, the objective is to analyze which combinations should be removed from the lattice in order to converge toward the optimal candidate role-set.

3.1.3 Discarding Candidate Roles

Suppose that we want to delete a role r whose permissions can be obtained from the union of permissions assigned to a subset of *ROLES*. Removing r

does not alter the completeness property of the role-set, but could lead to positive or negative fluctuation of the cost function value. In particular, the new administration cost would be changed by:

$$-\alpha u_r + \alpha \mu_r u_r - \beta p_r - \gamma - \delta c(r) \tag{3.2}$$

in that:

- $-\gamma$ indicates a role is removed from *ROLES*;
- $-\beta p_r$ indicates that p_r relationships are removed from *PA*, where $p_r = |auth\_perms(r)|$;
- $-\alpha u_r$ indicates that u_r relationships are removed from *UA*, where $u_r = |ass\_users(r)|$;
- $+\alpha \mu_r u_r$ indicates that role r must be replaced with different μ_r roles, so that each user having a relationship with role r would now have new relationships with μ_r different roles;
- $-\delta c(r)$ indicates that the cost related to other business information about r is no longer needed.

In order to reduce the administration cost, Equation (3.2) must thus be negative, that is:

$$(\mu_r - 1)u_r \leq \sigma p_r + \tau + \upsilon c(r) \tag{3.3}$$

where $\sigma = \beta/\alpha$, $\tau = \gamma/\alpha$ and $\upsilon = \delta/\alpha$. According to Equation (3.3), deleting a role from the candidate role-set is advantageous when the role is not supported by a sufficient number of users. The deletion becomes more advantageous as the role's degree increases or as the number of roles needed to replace it decreases. It should be noted that:

- when $\tau = 0$ (that is $\gamma = 0$) no weight is given to the number of roles;
- when $\sigma = 0$ (that is $\beta = 0$) no weight is given to the management of role-permission assignment;
- when $\sigma, \tau, \upsilon \to \infty$ (that is $\alpha \to 0$) no weight is given to the management of role-user assignment. Since Equation (3.3) is always true, deleting roles not affecting the completeness property is always worthwhile.

The choice of parameters σ, τ and υ is clearly dependent on the company we analyze, while $c(r)$ fluctuates based on its specific definition. However, p_r is constant respecting the role, while u_r and μ_r vary according to deletion of other roles, as follows:

u_r analysis Given two roles $r_1, r_2 \in ROLES$ such that $r_1 \succ r_2$, it is more advantageous that all users possessing all the child r_1 permissions be assigned to r_1 and not to the parent r_2. In fact, in the latter case further user-role assignments would be necessary. If r_1 is deleted, users assigned to r_1 would probably be assigned to r_2, resulting in an increase in the variable u_{r_2}. Only direct parents of the deleted role are affected. In the case in which role r_1 has one or more 1:1 or 1:n equivalent children, no user will be assigned to role r_1, thus we will have $u_{r_1} = 0$. This implies that all candidate roles having equivalent children can always be deleted since Equation (3.3) is always true.

μ_r analysis The roles which may replace r have to be exclusively sought among direct parents, as it is preferable to assign a user to the deepest child role in a hierarchy. Deleting a parent role can result in growth of μ_r if, for instance, instead of such a parent we choose roles with lower degrees. In other cases, deletion of a parent role may prevent the deletion of role r.

3.1.4 Finding Optimal Candidate Role-Sets

In the previous section we detailed how the administration cost can vary after the deletion of a candidate role. The optimal set is certainly to be a subset of the *PERMS* lattice. The aim is thus to identify which combinations must be deleted from the *PERMS* lattice to obtain the minimum cost while maintaining the completeness property.

The order in which the deletions take place is relevant when testing Equation (3.3). In particular, deleting a role exclusively affects the cost of direct parents and children. However, deleting a parent or child role can affect the deletion of other roles. Thus, the problem of identifying the correct set of roles to delete is not trivial. One possible solution is to scan the whole solutions space in search of a set that minimizes administration cost. Although such an algorithm leads to the optimal set, it is unfeasible as it has exponential complexity. In fact, given that $\alpha, \gamma = 1$, and $\beta, \delta = 0$ in Equation (3.1), the identification of the optimal set is equivalent to the *Role Mining Problem* (RMP) described in [92], which has proved to be $\mathcal{NP}$-complete.

Next section introduces an algorithm that allows us to approximate the optimal solution.

3.2 Approximating the Optimum

The following algorithm, called RBAM (*Role-Based Association-rule Mining*) offers a sub-optimal solution for the problem of identifying the optimal candidate role-set. Approximations introduced by RBAM are discussed throughout the section, highlighting how these do not invalidate the quality of the obtained output. Supporting examples will be given at the end of this section.

3.2.1 Lattice Generation

The generation of the candidate role-set based on the *PERMS* lattice is derived from the Apriori algorithm [17]. Apriori can be seen as an algorithm for the generation of a partial lattice. The solutions space is obtained by pruning the combinations whose support is lower than a pre-established and constant minimum. We now provide the following definition:

Definition 3.3 Among all users possessing the permissions assigned to role r, only a subset will likely be assigned to r. Therefore, we define *actual support*, as $actual\_support(r) = |ass\_users(r)|/|USERS|$.

Equation (3.3) shows that the gain introduced by a role is related to the number of users assigned to such a role. Since $u_r = |ass\_users(r)|$ then $actual\_support(r) = u_r/|USERS|$, so that Equation (3.3) becomes:

$$(\mu_r - 1)\, actual\_support(r) \leq \bar{\sigma} p_r + \bar{\tau} + \bar{\upsilon} c(r) \tag{3.4}$$

where $\bar{\sigma}$, $\bar{\tau}$ and $\bar{\upsilon}$ are obtained from σ, τ and υ dividing them by $|USERS|$. Equation (3.4) presents the advantage of being normalized by the number of users, thus $\bar{\sigma}$, $\bar{\tau}$ and $\bar{\upsilon}$ may be specified as parameters independent from the set *UP* to be analyzed.

Equation (3.4) may be used to implement an Apriori version with *variable minimum support*. Only roles not increasing the administration cost of previously calculated roles will be generated. The proposed RBAM algorithm is consequently composed of the following steps:

Step 1 An initial analysis of the set *UP* provides the set R_1 containing candidate roles of degree 1 with a support greater than the minimum.

Step k When $k \geq 2$ set R_k is generated merging all possible role pairs in R_{k-1} (*join step*). In order not to generate roles with the same permission set, a lexicographical order for the permission is given. Thus, only role pairs differing in the greater permission are considered. Combinations not meeting minimum support constraints are rejected (*prune*

step). Hierarchical association set H_k is also identified, relating roles in R_k whose assigned permissions are a superset of permissions assigned to roles in R_{k-1}.

Stop The algorithm completes when $R_k = \emptyset$, returning *ROLES* as the union of all calculated R_i and *RH* as the union of all calculated H_i.

Compared to the Apriori algorithm, the produced set *RH* is equivalent to "association-rules" between itemsets. Since we supposed a non hierarchical RBAC model, *RH* is not intended for inheritance of role-permission or role-user assignments. Nevertheless, identifying hierarchical relationships is still worthwhile, since it provides a means to "navigate" the candidate roles.

The first difference from Apriori is represented by the pruning operation. The minimum support constraint used in the prune step is a particular case of Equation (3.4). In fact:

- During the step k, level-$(k+1)$ roles are not yet generated. Thus, we have that $support(r) = actual\_support(r)$ for each $r \in R_k$.
- The join step combines role pairs calculated in the previous step which differ by only one permission. Then, all roles in R_k have degree k. This means that $p_r = k$ for each $r \in R_k$.
- Each role in R_k is established by merging a pair of roles in R_{k-1}, therefore it is always $\mu_r = 2$.

Starting from Equation (3.4), we can define the following pruning condition:

$$support(r) > \bar{\sigma}k + \bar{\tau} + \bar{\upsilon}c(r). \tag{3.5}$$

In order to preserve all the properties of the Apriori algorithm, the correctness of the prune step must be ensured through the following theorem:

Theorem 3.1 *Given $r_1, r_2 \in ROLES$ such that $r_1 \succeq r_2$ and $c(r_1) \geq c(r_2)$, if a role does not satisfy Equation* (3.5)*, then none of its child roles will be generated.*

PROOF If in step k the role r is not generated, then its support is such that $support(r) \leq \bar{\sigma}k + \bar{\tau} + \bar{\upsilon}c(r)$. Any child role r' will be generated from step $k+1$ onward. Furthermore, because of Equation (2.1) at page 12, we have $auth\_users(r') \subseteq auth\_users(r)$ and thus $support(r') \leq support(r)$. Furthermore, $c(r') \geq c(r)$. Therefore, $support(r') \leq \bar{\sigma}k + \bar{\tau} + \bar{\upsilon}c(r) \leq \bar{\sigma}(k+1) + \bar{\tau} + \bar{\upsilon}c(r')$. ∎

On the basis of Theorem 3.1, if a role r is rejected because of minimal support, then all its child roles will certainly not be generated (see Figure 3.1).

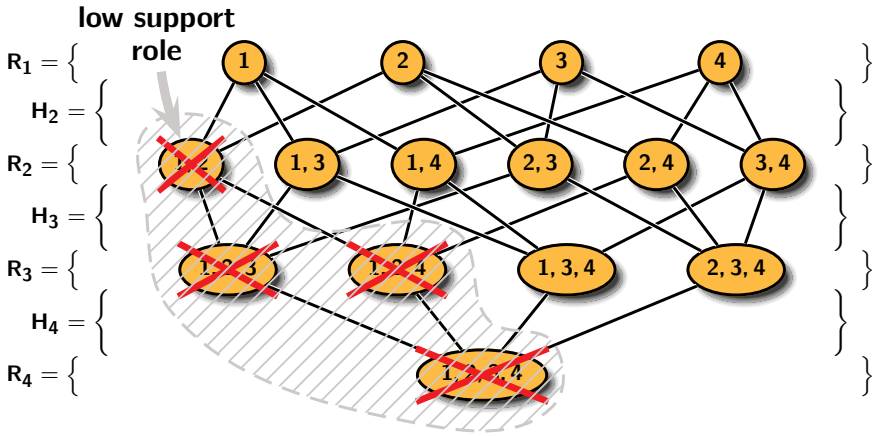


Figure 3.1 An instance of Theorem 3.1

This is coherent with the Apriori approach, but it does not always lead to the optimal administration cost. In fact, a role r with degree $k+1$, by construction of the algorithm, should be generated at step $k+1$. The generation of r could result in the deletion of level-k roles as it decreases users assigned to parents. However, Equation (3.5) does not take into account that generation of a role may result in the deletion of other roles.

Another important observation is that applying Equation (3.5) when $k = 1$ implies that all permissions whose support is too low are rejected. This means that the candidate role-set *is not always complete*. Rejected permissions must be individually managed through the creation of an *ad hoc* role for each permission. The number of users with such permissions is low, hence the approximation is usually acceptable.

Notice that when children of r are generated in step $k+1$, the value of u_r will decrease. This means that immediately after the generation of level-$(k+1)$, Equation (3.4) must be checked again for all level-k roles. At the end of level k the RBAM-purge procedure performs such an operation. This represents the other main difference from Apriori. RBAM-purge is described in the following section.

3.2.2 Removing High-Cost Roles

Since the generation of level-k roles influences the variable u_r of level-$(k-1)$ roles, further deletion of generated roles may thus be necessary. Algorithm 3.1 details the RBAM-purge procedure used to do so. The algorithm performs

3.1 RBAM-purge procedure, used to implement RBAM as a customized version of Apriori

procedure RBAM-purge($R_{k-1}, H_k, H_{k-1}, PA, UA, \bar{\sigma}, \bar{\tau}, \bar{\upsilon}$)
  {*Remove from parents the users also assigned to children*}
  $UA \leftarrow \{\langle u, r\rangle \in UA \mid u \notin \bigcup_{h \in H_k : h.\text{prnt} = r} ass\_users(h.\text{child})\}$
  for all $r \in R_{k-1}$ **do**
    $r.\text{act\_supp} \leftarrow |\{\langle u, r'\rangle \in UA \mid r' = r\}|/|USERS|$
  end for
  {*Identify removable roles with low support*}
  $\Delta \leftarrow \{r \in R_{k-1} \mid r.\text{act\_supp} = 0 \ \vee \ (r.\text{act\_supp} \leq \bar{\sigma}(k-1) + \bar{\tau} + \bar{\upsilon}c(r) \wedge$
    $r.\text{supp} \cdot |USERS| = \left|\bigcup_{h \in H_{k-1} : h.\text{child} = r} ass\_users(h.\text{prnt})\right|)\}$
  {*Remove roles with low support*}
  for all $r \in \Delta$ **do**
    {*Transfer only direct hierarchies*}
    for all $h_p \in H_{k-1}, h_c \in H_k : h_p.\text{child} = h_c.\text{prnt} = r$ **do**
      if $\nexists h' \in H_k : h'.\text{child} = h_c.\text{child} \ \wedge \ h'.\text{prnt} \notin \Delta \ \wedge$
        $\wedge \ ass\_perms(h'.\text{prnt}) \supseteq ass\_perms(h_p.\text{prnt})$ **then**
        $h.\text{prnt} \leftarrow h_p.\text{prnt}$
        $h.\text{child} \leftarrow h_c.\text{child}$
        $h.\text{conf} \leftarrow h_p.\text{conf} \cdot h_c.\text{conf}$
        $H_k \leftarrow H_k \cup \{h\}$
      end if
    end for
    {*Transfer users to parents, then remove r*}
    $UA \leftarrow \{\langle u, r'\rangle \mid \exists h \in RH, u \in USERS : h.\text{prnt} = r' \wedge h.\text{child} = r \ \wedge \ \langle u, r\rangle \in UA\}$
    for all $r' \in \{h.\text{prnt} \mid h \in RH \wedge h.\text{child} = r\}$ **do**
      $r'.\text{act\_supp} \leftarrow |\{\langle u, r''\rangle \in UA \mid r'' = r'\}|/|USERS|$
    end for
    $R_{k-1} \leftarrow R_{k-1} \setminus \{r\}$
    $H_{k-1} \leftarrow \{h \in H_{k-1} \mid h.\text{child} \neq r\}$
    $H_k \leftarrow \{h \in H_k \mid h.\text{prnt} \neq r\}$
    $PA \leftarrow \{\langle p, r'\rangle \in PA \mid r' \neq r\}$
    $UA \leftarrow \{\langle u, r'\rangle \in UA \mid r' \neq r\}$
  end for

  return $\langle R_k, R_{k-1}, H_k, H_{k-1}, PA, UA\rangle$
end procedure

operations upon the following data structures:

- The set *ROLES*. It represents the union of all the sets R_k. For each $r \in$ *ROLES* are identified:
  - r.supp: role r support;
  - r.act_supp: role r actual support;
  - r.degree: the number of permissions assigned to r.
- The set *RH* that hierarchically links candidate roles to one another. It represents the union of all sets H_k. This means that *only direct relationships are determined*. For each $h \in$ *RH* are identified:
  - h.prnt and h.child: parent and child roles hierarchically related;
  - h.conf: confidence value between roles.
- The set *PA*. This set merely correlates candidate roles with their assigned permissions.
- The set *UA*. It contains the *proposed role-user assignments*. At the end of step k, relationships between users and permissions assigned to the level-k roles are added to the set.

Algorithm 3.1 provides the following steps:

- *Lines 3–6*: Among the users assigned to level-$(k-1)$ roles, those assigned to level k children are removed from *UA*, thus updating the actual support.
- *Lines 8–9*: All level-$(k-1)$ roles that meet Equation (3.5) are identified after level-k role generation. Such roles can be deleted only if they preserve the completeness property, that is $u_r = 0$ (there is a 1:1 or 1:n equivalence with their children) or there are suitable sets of parent roles to replace them.
- *Lines 11–32*: Users assigned to roles being deleted are transferred to their parents. Hierarchies are transferred to children, ensuring that indirect hierarchy relationships are not generated. Then roles are deleted.

The algorithm introduces some approximations. Indeed, if a level-k role is deleted, the value of u_r relating to a level-$(k-1)$ role r could grow, so that Equation (3.4) is not likely to be satisfied. However, before step k is performed, Equation (3.4) could have been satisfied for role r. Thus it could have been deleted. If this is the case, the algorithm does not undelete r.

Another approximation is represented by lines 8–9. Variable μ_r does not explicitly appear as it is presumed to always be equal to 2. This gets rid of the need to calculate the actual value of μ_r, but it can result in a higher number of deletions with respect to the optimal solution.

Another simplification is introduced in lines 23–23, which "transfer" the users assigned to deleted roles to *all* their parents, without identifying a minimum set of parent roles to replace deleted roles. In this way, it is likely to have proposed more role-user assignments than actually needed, leaving to the administrator the burden of selecting the best subset among all proposed roles. This cannot be avoided without providing further elements defining role semantics.

Regarding hierarchical relationships, when a role is deleted all the hierarchical relationships with the parents must be "transferred" to the child roles. When a path between the child role and the inherited parent already exists, this operation can create indirect hierarchical relationships. Analyzing the set PA helps us determine if a hierarchical relationship must (or must not) be inherited. Given $r_1, r_2 \in ROLES$ such that $r_1 \rightarrow r_2$, where r_2 is the role being deleted, r_1 inherits $p \in ROLES$ such that of $r_2 \rightarrow p$ if and only if: $\nexists p' \in ROLES : r_1 \rightarrow p' \wedge ass\_perms(p') \supseteq ass\_perms(p)$. This is reflected in lines 14–15. In this way, we guarantee that RH only contains direct relationships. According to Lemma 2.2, the confidence of a new relationship is calculated as the product of confidences along the hierarchical path.

3.2.3 Examples

Two instances of the RBAM algorithm are provided below and summarized in Figure 3.2. Edges in the represented graph show confidence values, while values next to nodes show *support*() in bold and *actual_support*() between brackets.

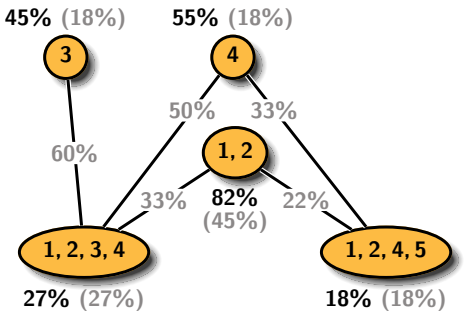
Figure 3.2(a) shows the application of the algorithm to the data in Figure 3.2(c), given $\tau = 1/11$ and $\sigma, \upsilon = 0$. The cost of this model is:

$$|UA| + \sigma|PA| + \tau|ROLES| = 14 + 0 \times 12 + 1 \times 5 = 19$$

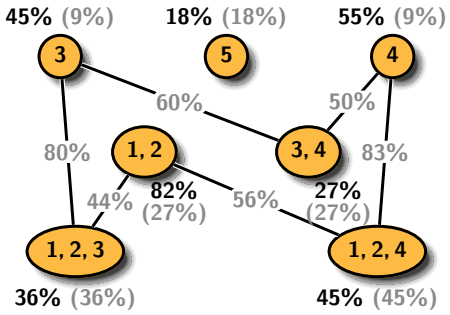
where the obtained value is divided by the parameter α. Since $\tau \neq 0$, the algorithm produces a low number of roles and attempts to minimize the number of user-role relationships. In fact, each user is assigned to a single role except for F, I, and J who are assigned to two roles. To understand the minimality of the result, it should be noted that the set $ROLES = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}\}$ also consists of five roles. In this case, the number of user-role assignments would be higher as each user should be assigned to a number of roles equal to the permissions held, that is $|PA| = 31$.

Figure 3.2(b) details the case in which $\tau, \upsilon = 0$ and $\sigma = 1/11$. The cost of the obtained model divided by α is:

$$|UA| + \sigma|PA| + \tau|ROLES| = 19 + 1 \times 13 + 0 \times 7 = 32.$$



(a) $\bar{\sigma} = 0, \bar{\tau} = 1/11$



(b) $\bar{\sigma} = 1/11, \bar{\tau} = 0$

User	Permissions	Roles (a)	Roles (b)
A	3	{3}	{3}
B	4	{4}	{4}
C	1, 2	{1, 2}	{1, 2}
D	1, 2	{1, 2}	{1, 2}
E	1, 2	{1, 2}	{1, 2}
F	1, 2, 3	{1, 2} + {3}	{1, 2, 3}
G	1, 2, 3, 4	{1, 2, 3, 4}	{1, 2, 3} + {1, 2, 4} + ~~{3, 4}~~
H	1, 2, 3, 4	{1, 2, 3, 4}	{1, 2, 3} + {1, 2, 4} + ~~{3, 4}~~
I	1, 2, 3, 4, 5	{1, 2, 3, 4} + {1, 2, 4, 5}	{1, 2, 3} + {1, 2, 4} + ~~{3, 4}~~ + {5}
J	1, 2, 4	{1, 2} + {4}	{1, 2, 4}
K	1, 2, 4, 5	{1, 2, 4, 5}	{1, 2, 4} + {5}

(c) Input user-permission assignment and proposed role-user assignment for examples (a) and (b)

Figure 3.2 Application of RBAM to toy examples

Since $\sigma \neq 0$, the algorithm generates roles with lower degree than the previous example. However, due to the approximations introduced by the algorithm, three roles are proposed for G, H, and I, even if only two are strictly necessary. Indeed, for these users it is sufficient to assign only one couple among $\{\{1,2,3\},\{3,4\}\}$, $\{\{1,2,4\},\{3,4\}\}$ and $\{\{1,2,3\},\{1,2,4\}\}$. The final decision can only be made by an administrator who knows the actual meaning of the permissions grouping. By deleting the redundant role $\{1,3\}$, the administration cost would become $16 + 1 \times 11 + 0 \times 6 = 27$.

The cost advantage obtained through the RBAM algorithm can be seen in comparison to the administration cost of an access control system which does not rely on the RBAC model. We can think about an RBAC model consisting of roles having only one permission, simulating a situation in which the permis-

sions are directly assigned to users. The administration cost of role-permission relationships can be overlooked (that is $\sigma = 0$) as it could be easily automated. Given $\tau = 1/11$ the cost without RBAC would be:

$$|UA| + \tau|ROLES| = 31 + 1 \times 5 = 36.$$

Using RBAM the cost is almost half compared to the result provided in Figure 3.2(a).

3.2.4 Comparative Analysis

The analysis proposed in this chapter includes and extends similar studies. It can in fact be lead to [103] given that $\alpha, \beta, \gamma = 1$ and $\delta = 0$ in Equation (3.1), while the analogy with [92] derives from $\alpha, \gamma = 1$ and $\beta, \delta = 0$. Compared to [92,103], the RBAM algorithm additionally offers a way to obtain a reasonable approximation of the optimal solution. The algorithm also allows higher level data to be taken into consideration, such as the roles designed with a top-down approach, permissions incompatibility data, user attributes etc., offering a chance to implement a hybrid role engineering approach. Another advantage concerns the possibility of choosing whether to implement or not implement a hierarchical RBAC model, given that not all commercial access control products offer this option. Furthermore, the RBAM algorithm also proposes possible users to roles assignment, further simplifying the work of the system administrator.

3.2.5 Testing With Real Data

To assess the efficiency of the RBAM algorithm, many tests have been conducted using real data. In order to highlight the properties of the algorithm, consider analysis results of data from an application presenting a heterogeneous distribution of permissions among users. Thus, the resulting authorization situation was virtually unanalyzable using standard role mining tools. In total, 4743 users possessing 2907 permissions were analyzed. By applying the RBAM algorithm with $\bar{\tau} = 200/4743 = 4.2\%$ and $\bar{\sigma} = 0$, the total of 42 roles were generated. The related cost is:

$$|UA| + \sigma|PA| + \tau|ROLES| = 6547 + 0 \times 100 + 200 \times 42 = 14947.$$

Not all the 2907 permissions were analyzed by RBAM, since only 23 permissions were held by at least 200 users (that is $\bar{\tau} = 4.2\%$). In particular, 4161 users (88%) were possessing these permissions. Without RBAC, the administration cost of the aforementioned 23 permissions would be:

$$|UA| + \sigma|PA| + \tau|ROLES| = 31557 + 0 \times 23 + 200 \times 23 = 36157$$

reducing the cost by 59%. Of course, the result can drastically improve if more complex cost functions are used.

The remaining $2907 - 23 = 2884$ permissions were scattered over 4654 users through 32182 user-permission associations. Consequently, permissions must be individually managed. Alternatively, the value of τ should be decreased so that more permissions can be considered. Otherwise, the analysis can be restricted to only those permission belonging to a single application module.

Finally, we consider computational complexity. It could be showed that in the worst case scenario RBAM is $\mathcal{NP}$-complete, as it derives from Apriori [100]. However, performing tests on real data sets has shown the algorithm to be quite efficient when using suitable values for parameters $\bar{\sigma}$ and $\bar{\tau}$ despite the tens of thousands of users and thousands of permissions. For example, if the value of $\bar{\tau}$ is high (i.e. we find only permission sets belonging to a large number of users) most of the permission lattice is pruned, thus reducing the combinations being analyzed.

3.3 Final Remarks

In this chapter we described a formal model to derive optimal role-users assignment. This model is driven by a cost-based function, where the cost is expressed in terms of the administration effort in managing the resulting RBAC model.

Further, we proposed the RBAM algorithm, that approximates the optimal solution (known to be $\mathcal{NP}$-complete) for this cost-based model. For specific parameter settings, the proposed algorithm even emulates other known algorithms. Further, as preliminary testing over real data sets shows, tuning its parameters it achieves even better performances than such known algorithms.

4 Pattern Identification in Users' Entitlements

In the following we describe a new formal framework applicable to role mining algorithms. This framework is based on a rigorous analysis of identifiable patterns in access permission data. In particular, it is possible to derive a *lattice of candidate roles* from the permission powerset. We formally prove some interesting properties about such lattices in a role engineering context. These properties can be used to optimize role mining algorithms. Data redundancies associated with co-occurrences of permissions among users can be easily identified and eliminated, allowing for increased output quality and reduced processing time. Moreover, we leverage the equivalence of the role identification problem above with the vertex coloring problem. Our main result is the proof that the minimum number of roles is sharply concentrated around its expected value. A further contribution is to show how this result can be applied to decide whether it is advisable to undertake the efforts to renew an RBAC state. This chapter summarizes the contribution previously published in [4, 7].

4.1 Patterns and Redundancies

As stated in Chapter 2, several role mining techniques proposed to date in literature seek to derive candidate roles through the identification of data patterns in existing access rights. Despite important differences among the various techniques, almost all can take advantage of some common principles summarized by the following:

- *If two access permissions always occur together among users, these should*

simultaneously belong to the same candidate roles. Without further access data semantics, a bottom-up approach cannot differentiate between a role made up of two permissions and two roles containing individual permissions [94]. Moreover, defining roles made up of as many permissions as possible minimizes the administration cost by reducing the number of role-user assignments (see Chapter 3).

- *If no user possesses a given combination of access permissions, it makes no sense to define a role containing such combination.* Similar to the previous point, if no user actually performs a task for which a certain permission set is necessary, it is usually better not to define a role containing such an unassignable set.

- *It is quite common within an organization to have many users possessing the same set of access permissions.* This is one of the main justifications that brought about the RBAC model. The creation of a role in connection with a set of co-occurring permissions is typically more advantageous since the number of relationships to be managed is reduced.

The following example clarifies the assertions just made, particularly that of the first point presented. If of the given four permissions p_1, p_2, p_3, p_4, the pair p_1, p_2 is always found together with p_3, p_4, it is advisable not to define two distinct roles $\{p_1, p_2\}$ and $\{p_3, p_4\}$ but, rather, a single role $\{p_1, p_2, p_3, p_4\}$. This is different from saying that no user possesses only p_1, p_2 without also having some other permission. Suppose some users possess only p_3, others only p_4, others p_1, p_2, p_3 and still others p_1, p_2, p_4. In this case, even if p_1, p_2 never occur "by themselves", it could be convenient to define the role $\{p_1, p_2\}$ since roles $\{p_3\}$ and $\{p_4\}$ will certainly already exist individually. Thus, avoiding roles $\{p_1, p_2, p_3\}$ and $\{p_1, p_2, p_4\}$.

Existing role mining techniques do not always exploit the above-mentioned observations, even though analyzing such data "recurrences" could improve the quality of proposed candidate roles or increase computational efficiency of the algorithms. To this aim, this chapter provides a new model capable of increasing output quality and reducing process time of role mining algorithms. The model revolves around identifiable patterns in access permissions data. Through analysis of user permissions, a *lattice* [31] of candidate roles can be constructed from the permission powerset. Notable properties of this lattice will be discussed to substantiate their effectiveness in optimizing role mining algorithms. Leveraging our results, data redundancies associated with co-occurrence of permissions among users can be easily identified and eliminated, thus improving the role mining output. To prove the merit of our proposal, we have applied our results to two algorithms: Apriori [17] and RBAM

(see Chapter 3). Applying them to a realistic data set yielded drastic reductions in running time and often provided significant redundancy elimination.

4.2 Roles Based on Permission-Powerset Lattice

In this section we shall investigate the relationship among candidate role-sets and permission powerset. Moreover, we shall formally prove some interesting properties about the lattice of such a powerset.

4.2.1 Mapping Patterns to Roles

We now introduce the model on which the following analysis is based. Consider the powerset of a set S (the set of all subsets of S) written as 2^S. The set 2^S can easily be ordered via subset inclusion "$\supseteq$". It can be demonstrated that $\langle 2^S, \supseteq, \cup, \cap \rangle$ is a lattice [31]. Setting $S = PERMS$ makes it possible to build an RBAC model based on all derivable roles from a given permission set. As the operator "$\succeq$" (see Section 2.1) is based on the inclusion operator "$\supseteq$" applied to permissions assigned to roles, it is thus natural to map the operators "$\curlyvee$" to "$\cup$" (the join of two roles represented by the union of all assigned permissions) and "$\curlywedge$" to "$\cap$" (the meet of two roles represented by shared permissions). Every permission combination of the lattice $\langle 2^{PERMS}, \succeq, \curlyvee, \curlywedge \rangle$ identifies the following: 1) an element of *ROLES*, 2) its corresponding relationships in *PA* to such permissions, 3) all permission inclusions in *RH* which involve the role and 4) all relationships in *UA* to users possessing such combination. *RH* is defined to represent the transitive reduction of the graph associated to the lattice. Moreover, if a user is assigned to a role r, then *UA* will contain relationships between r, its juniors and users assigned to them, namely $\forall r \in ROLES, \forall j \in \downarrow r : ass\_users(r) \subseteq ass\_users(j)$.

For simplicity sake, from now on the lattice $\langle 2^{PERMS}, \succeq, \curlyvee, \curlywedge \rangle$ is identified only with the set *ROLES*. The following are some basic properties of this lattice:

Lemma 4.1 *Removing a role r from ROLES, and its corresponding relationships in PA, UA, RH, such that $ass\_perms(r) \neq \bigcap_{r' \in ROLES} ass\_perms(r')$ and $ass\_perms(r) \neq \bigcup_{r' \in ROLES} ass\_perms(r')$, the resulting set ROLES is still a lattice.*

PROOF The role r such that $ass\_perms(r) = \bigcap_{r' \in ROLES} ass\_perms(r')$ represents a lower bound for any role pairs. Analogously, we have that $ass\_perms(r) = \bigcup_{r' \in ROLES} ass\_perms(r')$ represents an upper bound, thus lattice properties are preserved. ■

Note 4.1 Given $r \in ROLES$ then $\forall s \in \uparrow r : support(r) \geq support(s)$. In fact, users possessing permission combination $ass\_perms(r)$ do not necessarily possess other permissions. Analogously, $\forall j \in \downarrow r : support(r) \leq support(j)$. Apriori [17] and RBAM (see Chapter 3) algorithms use this property as a pruning condition to limit the solution space.

Based on the initial hypothesis of Section 4.1, *roles to which unused permission combinations are assigned do not represent significant candidate roles*. Such roles have support equal to 0 and can be eliminated from *ROLES*, except for the meet and join which are required to preserve lattice properties (see Lemma 4.1). Removing such roles results in a lattice that satisfies the following property:

Lemma 4.2 *The immediate seniors of a role $r \in ROLES$ differ from r by a single permission, that is* $\forall r,s \in ROLES : s \gg r \implies degree(s) = degree(r) + 1$.

PROOF For Equation (2.1) at page 12, any role represented by a subset of $ass\_perms(s)$ has support > 0 and is at least assigned to users $ass\_users(s)$. Thus, *ROLES* contains all roles obtained by removing a single permission from $ass\_perms(s)$, including r. ■

4.2.2 Equivalent Sublattices

Let *ROLES* be the lattice based on 2^{PERMS} in which roles with support equal to 0 have been eliminated, except for the meet and join. Such set has a very simple property: *every candidate role set is contained within*, since it provides all user-assignable permission combinations. Beyond eliminating roles having support equal to 0, this section shows that *it is also possible to remove roles presenting equivalence with other roles*, as they do not belong to any "reasonable" candidate role set.

Table 4.1 shows an example of *UP* presenting equivalence relationships. By observing the data, it can be noted that all users simultaneously possessing permissions 3 and 4 also always have permissions 2, 5 and 6. Figure 4.1 shows the role lattice built on the given set *UP* with junior roles above and senior roles below. Despite this being a directed graph, direction indicators are absent (from top to bottom) to avoid complicating the figure. Thicker lines represent hierarchical relationships with confidence equal to 1, namely equivalence relationships (see Lemma 2.4).

Next, we want to demonstrate that *when a role has more equivalent seniors, the combination of its assigned permissions still represents an equivalent role*.

Table 4.1 An example for *UP*

User	*Perms*	*User*	*Perms*	*User*	*Perms*	*User*	*Perms*	*User*	*Perms*
u_1	$\{1\}$	u_8	$\{1,3\}$	u_{15}	$\{3,6\}$	u_{22}	$\{1,3,5\}$	u_{29}	$\{2,4,6\}$
u_2	$\{2\}$	u_9	$\{1,4\}$	u_{16}	$\{4,5\}$	u_{23}	$\{1,3,6\}$	u_{30}	$\{1,2,3,5\}$
u_3	$\{3\}$	u_{10}	$\{1,5\}$	u_{17}	$\{4,6\}$	u_{24}	$\{1,4,5\}$	u_{31}	$\{1,2,3,6\}$
u_4	$\{4\}$	u_{11}	$\{1,6\}$	u_{18}	$\{1,2,3\}$	u_{25}	$\{1,4,6\}$	u_{32}	$\{1,2,4,5\}$
u_5	$\{5\}$	u_{12}	$\{2,5\}$	u_{19}	$\{1,2,4\}$	u_{26}	$\{2,3,5\}$	u_{33}	$\{1,2,4,6\}$
u_6	$\{6\}$	u_{13}	$\{2,6\}$	u_{20}	$\{1,2,5\}$	u_{27}	$\{2,3,6\}$	u_{34}	$\{2,3,4,5,6\}$
u_7	$\{1,2\}$	u_{14}	$\{3,5\}$	u_{21}	$\{1,2,6\}$	u_{28}	$\{2,4,5\}$	u_{35}	$\{1,2,3,4,5,6\}$

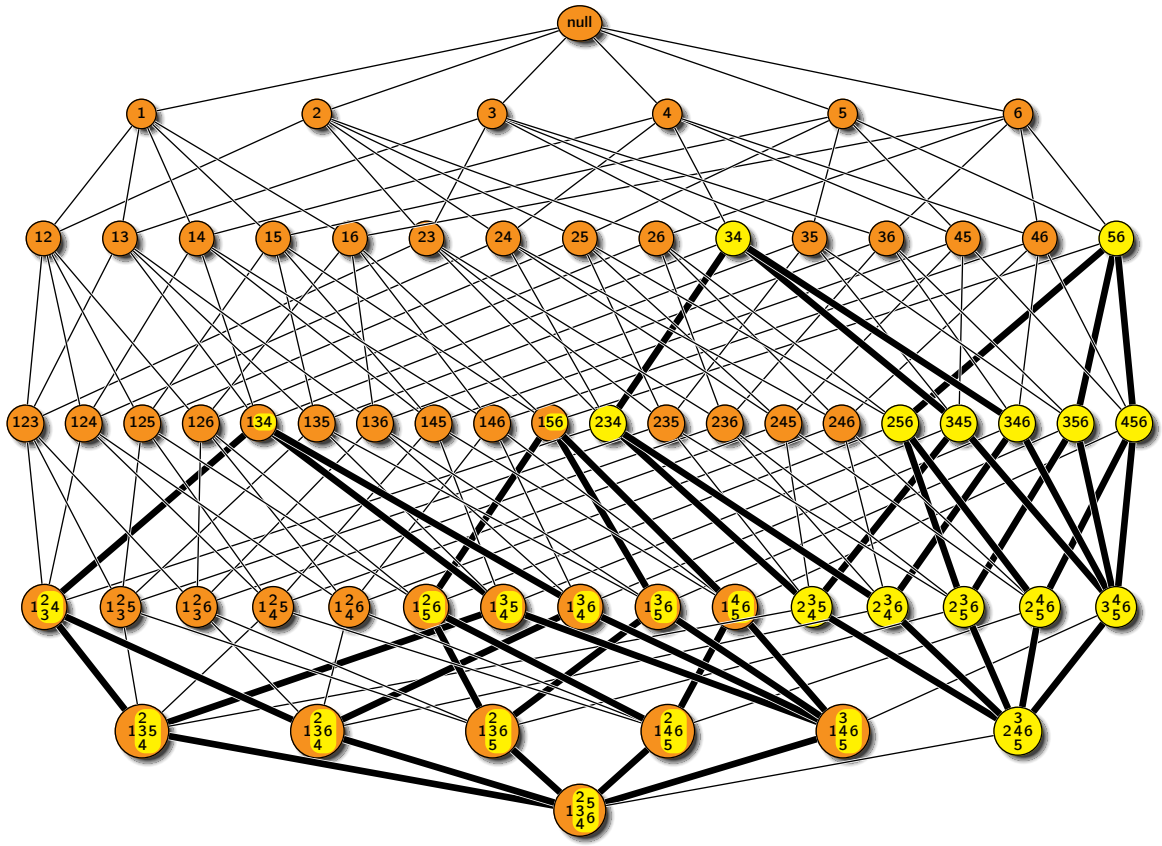


Figure 4.1 Hasse diagram of permission powerset derived from Table 4.1

For example, $\{3,4\} \equiv \{2,3,4\}$, $\{3,4\} \equiv \{3,4,5\}$ and $\{3,4\} \equiv \{3,4,6\}$ implies $\{3,4\} \equiv \{2,3,4,5,6\}$. Moreover, the set of equivalent seniors forms a sublattice. We will now formalize this with a series of theorems demonstrating that: (1) given an interval of roles, if the bounds are equivalent then all roles on the interval are equivalent with each other; (2) by analyzing immediate equivalent seniors, the equivalent role with the maximum degree can be determined; (3) an interval of equivalent roles having the equivalent role with the maximum degree as upper bound is a sublattice of *ROLES*; (4) such sublattice is replicated in *ROLES* with the same "structure".

Theorem 4.1 *Given a role pair $r_1, r_2 \in ROLES$ such that $r_2 \succeq r_1$ and $r_1 \equiv r_2$, then all roles on the interval $[r_1, r_2]$ are equivalent to each other:*

$$\forall r, r_1, r_2 \in ROLES : r_2 \succeq r \succeq r_1 \ \wedge \ r_1 \equiv r_2 \quad \Longrightarrow \quad r \equiv r_1 \equiv r_2.$$

PROOF According to Equation (2.1) at page 12, the following equation holds true: $ass\_users(r_2) \subseteq ass\_users(r) \subseteq ass\_users(r_1)$. But we also have that $ass\_users(r_1) = ass\_users(r_2)$, therefore the following equality holds as well: $ass\_users(r_2) = ass\_users(r) = ass\_users(r_1)$. ■

Theorem 4.2 *A role $r \in ROLES$ is equivalent to the role represented by the union of permissions assigned to any set of its equivalent seniors:*

$$\forall r \in ROLES, \forall R \subseteq \uparrow r, \forall r' \in R : r' \equiv r \quad \Longrightarrow$$
$$\Longrightarrow \quad \exists s \in ROLES : r \equiv s \ \wedge \ ass\_perms(s) = \bigcup_{r' \in R} ass\_perms(r').$$

PROOF Users possessing a role are those possessing all the permissions assigned to that role, namely

$$\forall r' \in ROLES : ass\_users(r') = \bigcap_{p \in ass\_perms(r')} perm\_users(p).$$

According to the hypothesis, $\forall r_i \in R : r_i \equiv r$, so all roles in R are assigned with the same users. Then

$$\bigcap_{r' \in R} \left(\bigcap_{p \in ass\_perms(r')} perm\_users(p) \right) = ass\_users(r).$$

Such an equality can also be written as

$$\bigcap_{p \in \bigcup_{r' \in R} ass\_perms(r')} perm\_users(p) = ass\_users(r)$$

but the set $\bigcup_{r' \in R} ass\_perms(r')$ represents the set of all permissions assigned to the role s. ■

Definition 4.1 Given $r \in ROLES$, the *maximum equivalent role* of r, written $\bar{r}$, is the role represented by the union of permissions of its immediate equivalent seniors:

$$ass\_perms(\bar{r}) = \bigcup_{r' \in ROLES \mid r' \gg r \wedge r' \equiv r} ass\_perms(r').$$

The name attributed to the role $\bar{r}$ is justified by the following theorem:

Theorem 4.3 *Given $r \in ROLES$, $\bar{r}$ is the equivalent role with the highest degree:*

$$\forall r' \in ROLES : r' \equiv r \ \wedge \ r' \neq \bar{r} \quad \Longrightarrow \quad degree(r') < degree(\bar{r}).$$

PROOF Seeking a contradiction, suppose that $r_{\max} \in ROLES : r_{\max} \neq \bar{r}$ is the highest degree role among all those equivalent to r. Since the same users possess both $\bar{r}$ and $r_{\max}$, then $ass\_perms(\bar{r}) \subseteq ass\_perms(r_{\max})$. If this was not the case, then there would exist another role within *ROLES* made up of the union of permissions assigned to $\bar{r}$ and $r_{\max}$ having a larger degree than both of these. This other role would also be equivalent to $\bar{r}$ and $r_{\max}$, since it is possessed by the same users. However, this contradicts the fact that $r_{\max}$ is of the highest degree.

Let $\Delta = ass\_perms(r_{\max}) \setminus ass\_perms(\bar{r})$. If $\Delta \neq \emptyset$, then it is possible to identify "intermediate" roles $\varrho \in [r, r_{\max}]$ such that $\exists p \in \Delta : ass\_perms(\varrho) = ass\_perms(r) \cup \{p\}$. For Lemma 4.2, $\varrho \gg r$, while for Theorem 4.1, $\varrho \equiv r$. Since $\bar{r}$ is obtained by the union of all permissions assigned to all equivalent immediate seniors, it contains all the permissions of Δ. Consequently, it must be that $\Delta = \emptyset$ and so $\bar{r} = r_{\max}$. ■

Theorem 4.4 *Given $r, s \in ROLES : s \succeq r$, the interval $[r, s]$ is a sublattice of ROLES.*

PROOF As long as $[r, s]$ is a lattice, it must be true that $\forall r_1, r_2 \in [r, s] : r_1 \curlyvee r_2 \in [r, s] \ \wedge \ r_1 \curlywedge r_2 \in [r, s]$. Given $r_1, r_2 \in [r, s]$, let r_{ub} be an upper-bound role such that $ass\_perms(r_{\text{ub}}) = ass\_perms(r_2) \cup ass\_perms(r_2)$. Since $s \succeq r_1, r_2$ then the permissions of s include the union of the permissions of r_1, r_2, so $s \succeq r_{\text{ub}}$. Thus, $r_{\text{ub}} \in [r, s]$. Similarly, it can be demonstrated that $[r, s]$ contains a lower-bound role r_{lb} such that $ass\_perms(r_{\text{lb}}) = ass\_perms(r_2) \cap ass\_perms(r_2)$. ■

Definition 4.2 Given a role $r \in ROLES$, we define the *equivalent sublattice* of r, indicated by $\varepsilon(r)$, the interval $[r, \bar{r}]$, that is $\varepsilon(r) = [r, \bar{r}]$.

Note 4.2 The set $\varepsilon(r)$ *does not* represent all the equivalent roles of r, rather, only a subset. In fact, we could have $r' \in ROLES$ such that $r \equiv r'$ even though $r \parallel r'$. However, for Theorem 4.3, from the union of permissions assigned to immediate equivalent seniors of r or r', the same maximum equivalent role is obtained, that is $\bar{r} \equiv \bar{r}'$. In fact, in Figure 4.1, roles $\{3,4\}$ and $\{5,6\}$ are antichain but, being equivalent to each other, they share the same maximum equivalent role $\{2,3,4,5,6\}$.

Note 4.3 If a role has equivalent seniors, then no user possesses only its permissions, namely

$$\exists r' \in (\uparrow r) \setminus r : r \equiv r' \implies ass\_users(r) \setminus \bigcup_{\varrho \in (\uparrow r) \setminus r} ass\_users(\varrho) = \emptyset.$$

The converse is not true. Particularly, if there is no user possessing a given permission combination, it is unknown whether the role made up of such permissions has immediate equivalent seniors. This is verified in Table 4.1. Permissions 3 and 4 are always found together with 2, 5 and 6. Thus, no user is assigned to role $\{3,4\}$ unless also assigned to one of its seniors. Yet, the contrary is not true: even though $\{2,3\}$ has no immediate equivalent seniors, it is not assigned with any user.

Theorem 4.5 *Given a role $r \in ROLES$, let $E = \{r' \in ROLES \mid r' \succcurlyeq r \ \wedge \ r' \equiv r\}$ be the set of immediate equivalent seniors of r. Then $|\varepsilon(r)| = 2^{|E|}$.*

PROOF For Lemma 4.2, $\forall r' \in E : degree(r') = degree(r) + 1$. Thus, permissions assigned to the maximum equivalent role of r include those of r plus a number of other permissions equal to $|E|$, that is $degree(\bar{r}) = degree(r) + |E|$. Further, $\varepsilon(r)$ contains all roles whose permission combinations are between $ass\_perms(r)$ and $ass\_perms(\bar{r})$, all of which have support greater than 0. Hence, the number of permission combinations between $ass\_perms(r)$ and $ass\_perms(\bar{r})$ is $2^{|E|}$. ■

Theorem 4.6 *Let there be $r, s \in ROLES$ such that s is an immediate equivalent senior of r. If there is $s' \in ROLES$, an immediate non-equivalent senior or r, then certainly there is a role $s'' \in ROLES$, an immediate equivalent senior of s' and immediate senior of s, represented by the union of permissions of s, s':*

$$\forall r, s, s' \in ROLES \ : \ s \succcurlyeq r \ \wedge \ s' \succcurlyeq r \ \wedge \ s \equiv r \ \wedge \ s' \not\equiv r \quad \implies \quad \exists s'' \in ROLES \ :$$
$$s'' \succcurlyeq s \ \wedge \ s'' \succcurlyeq s' \ \wedge \ s' \equiv s'' \ \wedge \ ass\_perms(s'') = ass\_perms(s) \cup ass\_perms(s').$$

PROOF The role s'' is a senior of both s, s' since $ass\_perms(s'') \supseteq ass\_perms(s)$ and $ass\_perms(s'') \supseteq ass\_perms(s')$. But we also have that $r \equiv s$, consequently $ass\_users(s'') = ass\_users(s) \cap ass\_users(s') = ass\_users(r) \cap ass\_users(s')$. Furthermore, $ass\_users(s') \subseteq ass\_users(r)$ because of $s' \succcurlyeq r$, then $ass\_users(s'') = ass\_users(s')$. Finally, for Lemma 4.2 roles s and s' have an additional permission to that of r. If $s \neq s'$ then $degree(s'') = degree(r) + 2$. Hence, s'' is an immediate senior to both s, s'. ■

Note 4.4 The previous theorem can be observed in Figure 4.1. The role $\{3,4\}$ has three immediate equivalent senior roles, while $\{1,3,4\}$ represents an immediate non-equivalent senior. For Theorem 4.6, this means that $\{1,3,4\}$ has *at least* three immediate equivalent seniors, identifiable by adding the permission 1 to equivalent seniors of $\{3,4\}$; according to Theorem 4.6, further immediate equivalent seniors of $\{1,3,4\}$ are allowed.

Theorem 4.7 *Let there be $r, s \in ROLES$ such that $s \succcurlyeq r$ and $s \not\equiv r$. Let also $p = ass\_perms(s) \setminus ass\_perms(r)$. Then there is a replica of the sublattice $\varepsilon(r)$ obtained by adding permission p to those of $\varepsilon(r)$.*

PROOF For Theorem 4.6, role s has among its immediate equivalent seniors at least those obtainable by adding permission p to immediate equivalent seniors of r. Let then $s' \in ROLES$ be the senior of s represented by the union of such immediate equivalent seniors, meaning $ass\_perms(s') = ass\_perms(\bar{r}) \cup \{p\}$. According to Theorem 4.2, $s \equiv s'$, while for Theorem 4.4 the interval $[s, s']$ is a sublattice. Let σ be a role defined from role $\varrho \in \varepsilon(r)$ such that $ass\_perms(\sigma) = ass\_perms(\varrho) \cup \{p\}$. Then, $s' \succeq \sigma$ since $ass\_perms(\bar{r}) \cup \{p\} \supseteq ass\_perms(\varrho) \cup \{p\}$ and $\sigma \succeq s$ because $ass\_perms(\varrho) \cup \{p\} \supseteq ass\_perms(r) \cup \{p\}$. Hence, $\sigma \in [s, s']$. ■

A direct consequence of the preceding theorem can be seen in Figure 4.1. The equivalent sublattice $\varepsilon(\{1,3,4\})$ can be obtained from $\varepsilon(\{3,4\})$ by adding the permission 1 to all roles. In the Hasse diagram of *ROLES* it is therefore possible to identify a certain number of equivalent sublattice replicas determined by:

Theorem 4.8 *Given a role $r \in ROLES$ let S be the set of immediate non-equivalent seniors, $S = \{\varrho \in ROLES \mid \varrho \succcurlyeq r \ \wedge \ \varrho \not\equiv r\}$. Then ROLES has a number of $\varepsilon(r)$ replicas between $|S|$ and $2^{|S|} - 1$.*

PROOF For Theorem 4.7, for all roles $s \in S$ the sublattice $\varepsilon(r)$ is replicated by adding permission $ass\_perms(s) \setminus ass\_perms(r)$ to every role in $\varepsilon(r)$. So, there

are at least $|S|$ sublattice replicas. Starting from S, let $P = \bigcup_{s \in S} ass\_perms(s) \setminus ass\_perms(r)$ the set of permissions added to r from non-equivalent seniors of r. For Lemma 4.2, the difference of degree between r and $s \in S$ is equal to 1, thus $|P| = |S|$. Every role $s \in S$ has at most $|S| - 1$ immediate non-equivalent seniors, meaning those represented by $ass\_perms(s)$ to which are added one of the permissions of $P \setminus (ass\_perms(s) \setminus ass\_perms(r))$. If, by contradiction, there was a role s', an immediate non-equivalent senior of s, for which $p = ass\_perms(s') \setminus ass\_perms(s) \wedge p \notin P$, then a role r' such that $ass\_perms(r') = ass\_perms(r) \cup \{p\}$ would have a support greater than 0 and would belong to S. This means that, still for Theorem 4.7, the role s can produce, at most, another $|S| - 1$ replicas. Reiterating the same reasoning for all seniors of r, it can be deduced that at most $2^{|S|} - 1$ replicas can be constructed by roles of $\varepsilon(r)$ to which are added permission combinations of $2^P \setminus \{\emptyset\}$. ■

4.3 Leveraging Role Equivalence to Improve Role Mining

The previous section analyzed some properties of a role lattice based on the powerset of permissions excluding combinations of support equal to 0. It was shown that a certain number of equivalent sublattice replicas could exist within such lattice. Based on the premises of Section 4.1, *all these replicas can be eliminated from the set of candidate roles except for maximum equivalent roles*. In fact, a maximum equivalent role can be considered a "representative" of all sublattices to which it belongs. Removing equivalent sublattices prunes the candidate role set solution space. Given a role $r \in ROLES$, let $E = \{r' \in ROLES \mid r' \succcurlyeq r \ \wedge \ r' \equiv r\}$ be the set of immediate equivalent seniors and $S = \{r' \in ROLES \mid r' \succcurlyeq r \ \wedge \ r' \not\equiv r\}$ be the set of immediate *non*-equivalent seniors. For Theorem 4.5, the equivalent sublattice generated by r contains $|\varepsilon(r)| = 2^{|E|}$ roles, all of which can be eliminated from $ROLES$ except for $\bar{r}$. Based on the theorems of the preceding section, $\varepsilon(r)$ and $\bar{r}$ can be derived from r and E. Prospective algorithms calculating roles based on the permission-powerset lattice could benefit from eliminating equivalent sublattices if $2^{|E|} > |E| + 1$, namely when the cost of calculating $\varepsilon(r)$ is greater than the cost of calculating only the roles necessary for identifying $\bar{r}$. For simplicity, operating costs necessary for constructing role $\bar{r}$ from r and E are deemed negligible. The inequality $2^{|E|} > |E| + 1$ is always true when $|E| > 1$, namely when role r has more than one equivalent junior. For Theorem 4.8, every equivalent sublattice has at least $|S|$ number of replicas derivable from r, E, S.

4.1 Procedure Remove-Equivalent-Sublattices, used to obtain improved versions of Apriori and RBAM

Require: R_k, H_k, PA, UA, k
Ensure: R_k, H_k, PA, UA, M_i
1: $W \leftarrow \emptyset$ {Set of equivalent roles to be deleted}
2: $M_i \leftarrow \emptyset$ {Set of maximum equivalent roles}
3: **for all** $\varrho \in \{h.junior \mid h \in H_k : h.confidence = 1\}$ **do**
4: {Identify equivalences in R_k to be deleted and maximum equivalent role permissions}
5: $E \leftarrow \{h.senior \mid h \in H_k : h.junior = \varrho \ \wedge \ h.confidence = 1\}$ {Equivalent seniors}
6: $S \leftarrow \{h.senior \mid h \in H_k : h.junior = \varrho \ \wedge \ h.confidence < 1\}$ {Non-equivalent seniors}
7: $P \leftarrow (\bigcup_{r \in E} ass\_perms(r)) \setminus ass\_perms(\varrho)$ {Perms diff between maximum equiv role}
8: $W \leftarrow W \cup E$ {Mark equivalent immediate seniors for deletion}
9: {Transform ϱ into its maximum equivalent role. Enrich roles in S with permissions P.}
10: **for all** $\sigma \in S \cup \{\varrho\}$ **do**
11: $\sigma.degree \leftarrow \sigma.degree + |P|, \quad PA \leftarrow PA \cup (P \times \{\sigma\}), \quad M_i \leftarrow M_i \cup \{\sigma\}$
12: **end for**
13: **end for**
14: {Delete equivalent roles in R_k}
15: $R_k \leftarrow R_k \setminus W, \quad PA \leftarrow \{\langle p, r\rangle \in PA \mid r \notin W\}, \quad UA \leftarrow \{\langle u, r\rangle \in UA \mid r \notin W\}$
16: $H_k \leftarrow \{h \in H_k \mid h.senior \notin W\}$

It is thus advantageous to remove these when $(|S|+1)2^{|E|} > |E|+|S|+1$, that is true when $|E| > 1$, where $(|S|+1)2^{|E|}$ represent the amount pruned.

4.3.1 Equivalent Sublattice Pruning in Apriori

This section introduces the RB-Apriori (*Role-Based Apriori*) algorithm to identify roles based on permission-powerset lattices with no equivalent sublattices. Using the Apriori [17] algorithm makes it possible to generate a partial lattice by pruning permission combinations whose support is lower than a pre-established threshold $s_{\min}$ (see Chapter 3). RB-Apriori extends Apriori removing equivalent sublattices except for the maximum equivalent roles. The following are the main steps of Apriori summarized. The set $R_k \subseteq ROLES$ denotes all roles calculated at step k of the algorithm, while $H_k \subseteq RH$ gathers the immediate hierarchical relations among roles in R_i and R_{i-1}.

Step 1 An initial analysis of *UP* provides the set R_1 containing candidate roles of degree 1 with a support greater than the minimum.

Step k When $k \geq 2$, the set R_k is generated merging all possible role pairs in R_{k-1} (*join step*). In order not to generate roles with the same permission set, only role pairs differing in the greater permission are considered. Combinations not meeting minimum support constraints are rejected (*prune step*). Hierarchical associations (H_k) are also identified, relating roles in R_k whose assigned permissions are a superset of permissions of roles in R_{k-1}.

Stop The algorithm completes when $R_k = \emptyset$, returning *ROLES* as the union of all calculated R_i and *RH* as the union of all calculated H_i.

The algorithm RB-Apriori is obtained from Apriori by calling the procedure Remove-Equivalent-Sublattices procedure at the end of every step k. The procedure is described in Algorithm 4.1. Given $r \in ROLES$, $r.degree$ indicates the number of permissions assigned to it; given $h \in RH$, $h.junior$ and $h.senior$ indicate the pair of roles hierarchically related, while $h.confidence$ is the confidence value between them. Step 3 of Algorithm 4.1 identifies all roles calculated in step $k-1$ presenting immediate equivalent seniors in R_k. For each of these roles, the steps immediately following determine sets E, S and the permission set P to be added to the role in order to obtain the maximum equivalent role. Steps 10–12 make up the maximum equivalent role by adding permissions P to the current role. The immediate non-equivalent seniors are also enriched with the same permissions; if not, eliminating roles E (Steps 8, 15–16) could prevent identification of the combination of permissions assigned to those roles during step $k+1$. Based on the Note 4.4, enriching permissions assigned to immediate non-equivalent seniors with P it is not definite that the respective maximum equivalent roles will be generated. This means that RB-Apriori *prunes only one sublattice at a time*, without also simultaneously eliminating any replicas.

As described in Note 4.2, there could exist $r_1, r_2 \in ROLES : r_1 \equiv r_2 \wedge r_1 \parallel r_2$. In Figure 4.1, roles $\{3,4\}$ and $\{5,6\}$ are equivalent and share the same maximum equivalent role $\{2,3,4,5,6\}$. According to Algorithm 4.1, the role $\{2,3,4,5,6\}$ is built twice. This means that after the last step ("Stop") of RB-Apriori it is necessary to check for duplicate roles. Particularly, given the set $M = \bigcup M_i$ of identified maximum equivalent roles, for every $m \in M$ each $r \in ROLES \setminus \{m\} : ass\_perms(r) \subseteq ass\_perms(m) \wedge support(r) = support(m)$ needs to be discarded.

4.3.2 Testing on Real Data

To assess the efficiency of the RB-Apriori algorithm described in the previous section, many tests have been conducted using real data. In order to highlight

the properties of the algorithm, consider the results obtained from analyzing data of an application with a heterogeneous distribution of user permissions. In the analyzed data set, 954 users were possessing 1,108 different permissions. By applying the Apriori algorithm with $s_{\min} = 10\%$, a total of 299 roles were generated in about 119 seconds through the adopted Apriori implementation. These 299 roles were assigned with only 16 of the available 1,108 permissions resulting in 890 users possessing these permissions. Using the same minimum support, with RB-Apriori we obtained only 109 roles in 87 seconds, thus reducing the number of roles by 64% and the computation time by 27%. The difference in improvement between role number and computation time was due to time "wasted" in identifying equivalent sublattices. Actually, the algorithm identified 167 roles; although 58 of the 167 were subsequently eliminated as equivalents, time was saved avoiding computation of entire equivalent sublattices. Changing the minimum support to $s_{\min} = 5\%$, 8,979 roles were produced with Apriori in about 3,324 seconds, involving 31 permissions and 897 users. With RB-Apriori we obtained only 235 roles in 349 seconds, thus reducing the number of roles by 97% and computation time by 90%.

4.3.3 Comparison With RBAM

The RBAM algorithm (see Chapter 3) leverages the RBAC administration cost estimate to find the lowest cost candidate role-sets, implementing an extended version of Apriori to identify the optimal role set. Pruning operations are based on the variable minimum support concept. According to Chapter 3, a role $r \in ROLES$ can be removed when the percentage of users assigned to r but none of its seniors is below a threshold related to the administration cost of r. When r has equivalent seniors, this percentage is equal to 0 because of Note 4.3. Thus, RBAM always removes its equivalent sublattice. Since RBAM is an extended version of Apriori, it is easy to improve performances of the RBAM algorithm, basing it on RB-Apriori instead of Apriori. While producing the same candidate role sets, computation of the entire equivalent sublattices is avoided, thus improving the efficiency and obtaining performance comparable to RB-Apriori.

4.4 Finding the Minimum Number of Roles

In this section we provide a probabilistic method to estimate the number of roles needed to cover all the existing user-permission assignments. The method leverages a known reduction of the role number minimization problem to the chromatic number of a graph. We prove that the optimal role num-

ber is sharply concentrated around its expected value. We also show how this result can be used as a *stop condition* when striving to find an approximation of the optimum for any role mining algorithm. The corresponding rational is that if a result is close to the optimum, and the effort required to discover a better result is high, it might be appropriate to accept the current result.

4.4.1 Martingales and Azuma-Hoeffding Inequality

We shall now present some definitions and theorems that provide the mathematical basis we will further discuss later on in this chapter. In particular, we introduce: martingales, Doob martingales, and the Azuma-Hoeffding inequality. These are well known tools for the analysis of randomized algorithms [67, 97].

Definition 4.3 (Martingale) A sequence of random variables $Z_0, Z_1, \ldots, Z_n$ is a *martingale with respect to the sequence* $X_0, X_1, \ldots, X_n$ if for all $n \geq 0$, the following conditions hold:

- Z_n is function of $X_0, X_1, \ldots, X_n$,
- $\mathbb{E}[|Z_n|] \leq \infty$,
- $\mathbb{E}[Z_{n+1} \mid X_0, \ldots, X_n] = Z_n$,

where the operator $\mathbb{E}[\cdot]$ indicates the expected value of a random variable. A sequence of random variables $Z_0, Z_1, \ldots$ is called *martingale* when it is a martingale with respect to himself. That is $\mathbb{E}[|Z_n|] \leq \infty$ and $\mathbb{E}[Z_{n+1} \mid Z_0, \ldots, Z_n] = Z_n$.

Definition 4.4 (Doob Martingale) A *Doob martingale* refers to a martingale constructed using the following general approach. Let $X_0, X_1, \ldots, X_n$ be a sequence of random variables, and let Y be a random variable with $\mathbb{E}[|Y|] < \infty$. (Generally Y, will depend on $X_0, X_1, \ldots, X_n$.) Then

$$Z_i = \mathbb{E}[Y \mid X_0, \ldots, X_i], \;\; i = 0, 1, \ldots, n,$$

gives a martingale with respect to $X_0, X_1, \ldots, X_n$.

The previous construction assures that the resulting sequence $Z_0, Z_1, \ldots, Z_n$ is always a martingale.

A useful property of the martingales that we will use in this chapter is the Azuma-Hoeffding inequality [67]:

Theorem 4.9 (Azuma-Hoeffding inequality) *Let $X_0, \ldots, X_n$ be a martingale s.t.*

$$B_k \leq X_k - X_{k-1} \leq B_k + d_k,$$

for some constants d_k and for some random variables B_k that may be functions of $X_0, X_1, \ldots, X_{k-1}$. Then, for all $t \geq 0$ and any $\lambda > 0$,

$$\Pr(\left|X_t - X_0\right| \geq \lambda) \leq 2\exp\left(\frac{-2\lambda^2}{\sum_{k=1}^{t} d_k^2}\right). \tag{4.1}$$

The Azuma-Hoeffding inequality applied to the Doob martingale gives the so called *Method of Bounded Differences* (MOBD) [66].

4.4.2 Problem Modeling

The main goal related to mining roles is to find optimal candidate role-sets. We focus on optimizing a particular cost function. Let *cost* indicate the number of needed roles. The role mining objective then becomes to find a candidate role-set that has the minimum number of roles for a given system configuration. We will show that this problem is equivalent to that of finding the chromatic number of a given graph. Using this problem equivalence, we will identify a useful property on the concentration of the optimal candidate role-sets. This allows us to provide a stop condition for any iterative role mining algorithm that approximates the minimum number of roles.

Given the configuration $\varphi = \langle USERS, PERMS, UP\rangle$ we can build a bipartite graph $G = \langle V, E\rangle$, where the vertex set V is partitioned into the two disjoint subset *USERS* and *PERMS*, and where E is a set of pairs $\langle u, p\rangle$ such that $u \in USERS$ and $p \in PERMS$. Two vertices u and p are connected if and only if $\langle u, p\rangle \in UP$. A biclique coverage of the graph G identifies a unique candidate role-set for the configuration φ [34], that is $\psi = \langle ROLES, UA, PA\rangle$. Indeed, every biclique identifies a role, and the vertices of the biclique identify the users and the permission assigned to this role. Let the function *cost* (see Chapter 2) return the number of roles, that is:

$$cost(\varphi, \psi) = |ROLES| \tag{4.2}$$

In this case, minimizing the cost function is equivalent to finding a candidate role-set that minimizes the number of roles. Let $\mathcal{B}$ a biclique coverage of a graph G, we define the function $cost'$ as:

$$cost'(\mathcal{B}) = cost(\varphi, \psi)$$

where ψ is the state $\langle UA, PA, ROLES\rangle$ that can be deduced by the biclique coverage $\mathcal{B}$ of G, and G is the bipartite graph built from the configuration φ that is uniquely identified by $\langle USERS, PERMS, UP\rangle$. In this model, the problem of finding an optimal candidate role-set can be equivalently expressed as finding

a biclique coverage for a given bipartite graph G that minimizes the number of required bicliques. This is exactly the *minimum biclique coverage* (MBC) problem. In the following we first recall both the reduction of the MBC problem to the *minimum clique partition* (MCP) problem [34] and the reduction of MCP to the chromatic number problem.

From the graph G, it is possible to construct a new undirected unipartite graph G' where the edges of G become the vertices of G': two vertices in G' are connected by an edge if and only if the endpoints of the corresponding edges of G induce a biclique in G. Formally:

$$G' = \langle E,\ \{\langle e_1, e_2\rangle \mid e_1, e_2 \text{ induce a biclique in } G\}\rangle$$

The vertices of a (maximal) clique in G' correspond to a set of edges of G, where the endpoints induce a (maximal) biclique in G. The edges covered by a (maximal) biclique of G induce a (maximal) clique in G'. Thus, every biclique edge 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 to which it belongs to. Similarly, any clique partition of G' corresponds to a biclique cover of G. Thus, the size of a minimum biclique coverage of a bipartite graph G is equal to the size of a minimum clique partition of G'.

Finding a clique partition of a graph $G = \langle V, E\rangle$ is equivalent to finding a coloring of its complement $\overline{G} = \langle V, (V \times V) \setminus E\rangle$. This implies that the biclique cover number of a bipartite graph G corresponds to the chromatic number of $\overline{G'}$ [34].

4.5 A Concentration Result for Role Number Minimization

Using the model described in the previous section, we will prove that the cost of an optimal candidate role-set ψ for a given system configuration φ is tightly concentrated around its expected value. We will use the concept of martingales and the Azuma-Hoeffding inequality to obtain a concentration result for the chromatic number of a graph G [66,67]. Since finding the chromatic number is equivalent to both MCP and MBP, we can conclude that the minimum number of roles required to cover the user-permission relationships in a given configuration is tightly concentrated around its expected value.

Let G be an undirected unipartite graph, and $\chi(G)$ its chromatic number.

Theorem 4.10 *Given a graph G with n vertices, the following equation holds:*

$$\Pr(\left|\chi(G) - \mathbb{E}[\chi(G)]\right| \geq \lambda) \leq 2\exp\left(\frac{-2\lambda^2}{n}\right) \tag{4.3}$$

PROOF We fix an arbitrary numbering of the vertices from 1 to n. Let G_i be the subgraph of G induced by the set of vertices $1,\ldots,i$. Let $Z_0 = \mathbb{E}[\chi(G)]$ and $Z_i = \mathbb{E}[\chi(G) \mid G_1,\ldots,G_i]$. Since adding a new vertex to the graph requires no more than one new color, the gap between Z_i and Z_{i-1} is at most 1. This allows to apply the Azuma-Hoeffding inequality, that is Equation (4.1) where $d_k = 1$. ∎

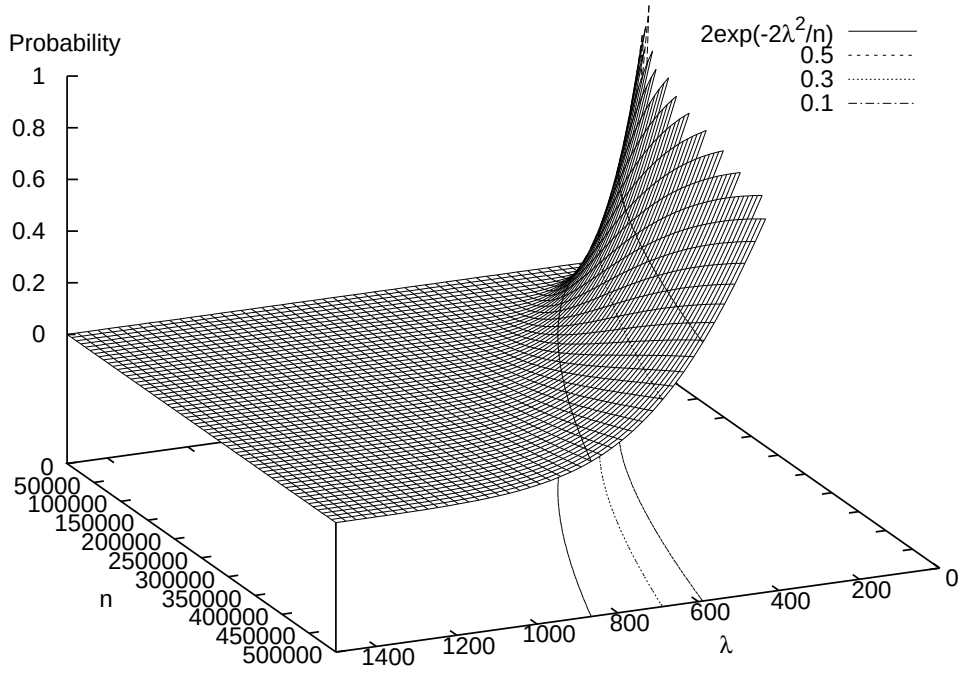
Note that this result holds even without knowing $\mathbb{E}[\chi(G)]$. Informally, Theorem 4.10 states that the chromatic number of a graph G is sharply concentrated around its expected value. Since finding the chromatic number of a graph is equivalent to MCP, and MCP is equivalent to MBC, this result holds also for MBC. Translating these concepts in terms of RBAC entities, this means that the cost of an optimal candidate role-set of any configuration φ with $|UP| = n$ is sharply concentrated around its expected value according to Equation (4.3), where $\chi(G)$ is equal to the minimum number of required roles. It is important to note that n represents the number of vertices in the coloring problem but, according to the proposed model, it is also the number of edges in MBP; that is, the user-permission assignments of the system configuration.

Figure 4.2(a) shows the plot of the Equation (4.3) for n varying between 1 and 500,000, and λ less than 1,500. It is possible to see that for $n = 500{,}000$ it is sufficient to choose $\lambda = 900$ to assure that $\Pr(\left|\chi(G) - \mathbb{E}[\chi(G)]\right| \geq \lambda) \leq 0.1$. In the same way, choosing $\lambda = 600$, then $\Pr(\left|\chi(G) - \mathbb{E}[\chi(G)]\right| \geq \lambda)$ is less than 0.5. Figure 4.2(b) shows the values for λ and n to have the left part of the inequality in Equation (4.3) to hold with probability less than 0.5, 0.3, and 0.1 respectively.

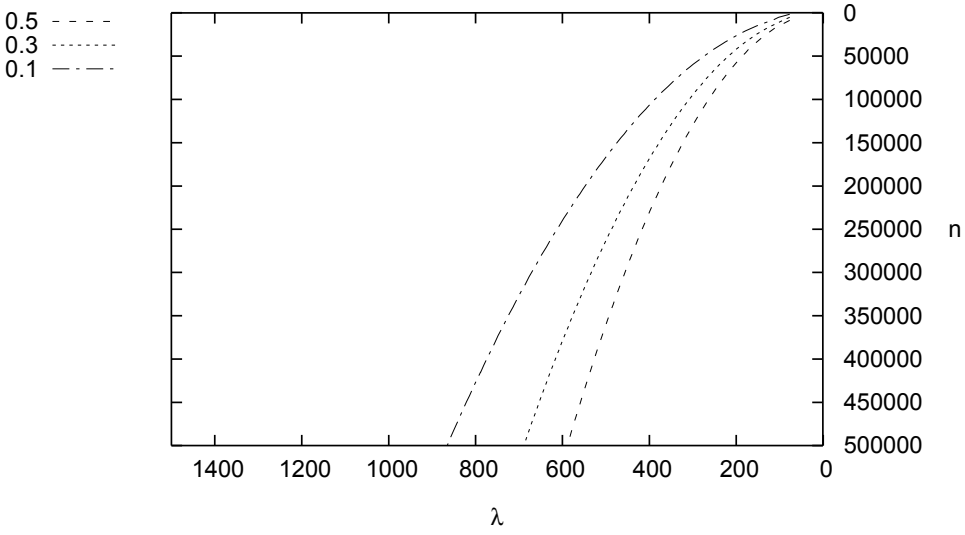
Setting $\lambda = \sqrt{n \log n}$, Equation (4.3) can be expressed as:

$$\Pr(\left|\chi(G) - \mathbb{E}[\chi(G)]\right| \geq \sqrt{n \log n}) \leq \frac{2}{n^2} \tag{4.4}$$

That is, the probability that our approach differ from the optimum more than $\sqrt{n \log n}$ is less than $2/n^2$. This probability becomes quickly negligible as n increases. To support the viability of the result, note that in a large organization there are usually thousands user-permission assignments.



(a) Plot of Equation 3



(b) Highlight of some λ values for Figure 1(a)

Figure 4.2 Relationship between λ, n, and resulting probability

4.5.1 Applications of the Bound

Assuming that we can estimate an approximation $\tilde{\mathbb{E}}[\chi(G)]$ for $\mathbb{E}[\chi(G)]$ such that $\left|\tilde{\mathbb{E}}[\chi(G) - \mathbb{E}[\chi(G)]\right| \leq \varepsilon$ for any $\varepsilon > 0$, Theorem 4.10 can be used as a *stop condition* when striving to find an approximation of the optimum for any role mining algorithm. Indeed, suppose that we have a probabilistic algorithm that provides an approximation of $\chi(G)$, and suppose that its output is $\tilde{\chi}(G)$. Since we know $\tilde{\mathbb{E}}[\chi(G)]$, we can use this value to evaluate whether the output is acceptable and therefore decide to stop the iterations procedure. Indeed, we have that:

$$\Pr(\left|\chi(G) - \tilde{\mathbb{E}}(\chi(G))\right| \geq \lambda + \varepsilon) \leq 2\exp\left(\frac{-2\lambda^2}{n}\right).$$

This is because

$$\Pr(\left|\chi(G) - \tilde{\mathbb{E}}(\chi(G))\right| \geq \lambda + \varepsilon) \leq \Pr(\left|\chi(G) - \mathbb{E}(\chi(G))\right| \geq \lambda)$$

and, because of Theorem 4.10, this probability is $\leq 2\exp\left(-2\lambda^2/n\right)$. Thus, if $\left|\tilde{\chi}(G) - \tilde{\mathbb{E}}[\chi(G)]\right| \leq \lambda + \varepsilon$ holds, then we can stop the iteration, otherwise we have to reiterate the algorithm until it outputs an acceptable value.

For a direct application of this result, we can consider a system configuration with $|UP| = x$. If $\lambda = y$, the probability that $\left|\chi(G) - \mathbb{E}[\chi(G)]\right| \leq y$ is greater than $2\exp\left(-2y^2/x\right)$. We do not know $\mathbb{E}[\chi(G)]$, but since we have that $\left|\tilde{\mathbb{E}}[\chi(G)] - \mathbb{E}[\chi(G)]\right| \leq \varepsilon$ we can conclude that $\left|\chi(G) - \tilde{\mathbb{E}}[\chi(G)]\right| < y + \varepsilon$ with probability at least $2\exp\left(-2y^2/x\right)$. For instance, we have considered the real case of a large size company, with 500,000 user-permissions assignments. With $\lambda = 1,200$ and $\varepsilon = 100$, the probability that $\left|\chi(G) - \tilde{\mathbb{E}}[\chi(G)]\right| < \lambda + \varepsilon$ is at least 99.36%. This means that, if $\tilde{\mathbb{E}}[\chi(G)] = 24,000$, with the above probability the optimum is between 22,700 and 25,300. If a probabilistic role mining algorithm outputs a value $\tilde{\chi}(G)$ that is estimated quite from this range, then it is appropriate to reiterate the process in order to find a better result. Conversely, let us assume that the algorithm outputs a value within the given range. We know that the identified solution differs, from the optimum, by at most $2(\lambda + \varepsilon)$, with probability at least 99.36%. Thus, one can assess whether it is appropriate to continue investing resources in the effort to find a better solution, or to simply accept the provided solution. This choice can depend on many factors, such as the computational cost of the algorithm, the economic cost due to a new analysis, and the error that we are prone to accept, to name a few.

There is also another possible application for this bound. Assume that a company is assessing whether to renew its RBAC state, just because it is

several years old [93]. By means of the proposed bound, the company can establish whether it is the case to invest money and resources in this process. Indeed, if the cost of the RBAC state in use is between $\tilde{\mathbb{E}}[\chi(G)] - \lambda - \varepsilon$ and $\tilde{\mathbb{E}}[\chi(G)] + \lambda + \varepsilon$, the best option would be not to renew it because the possible improvement is likely to be marginal. Moreover, changing the RBAC state requires a huge effort for the administrators, since they need to get used to the new configuration. In our proposal it is quite easy to assess if a renewal is needed. This indication can lead to important time and money saving.

Note that in our hypothesis, we assume that the value of $\tilde{\mathbb{E}}[\chi(G)]$ is known. Currently, not many researchers have addressed this specific issue in reference to a generic graph, whereas plenty of results have been provided for Random Graphs. In particular, it has been proven [25, 64] that for $G \in G_{n,p}$:

$$\mathbb{E}[\chi(G)] \sim \frac{n}{2\log_{\frac{1}{1-p}} n}$$

We are presently striving to apply a slight modification of the same probabilistic techniques used in this chapter, to derive a similar bound for the class of graphs used in our model.

4.6 Final Remarks

This chapter introduced a new formal framework based on a rigorous pattern analysis in access permissions data. In particular, it is possible to derive a *lattice* of candidate roles from the permission powerset. We have proved some interesting properties about the above-defined lattice useful for optimizing role mining algorithms. By leveraging our results, data redundancies associated with co-occurrence of permissions among users can be easily identified and eliminated, hence increasing output quality and reducing process time of data mining algorithms. To prove the effectiveness of our proposal, we have applied our results to two role mining algorithms: Apriori and RBAM. Applying these modified algorithms to a realistic data set, we drastically reduced the running time, while the output quality was either unaffected or even improved. Thus, we confirmed our analytical findings.

Further, we proved that the optimal administration cost for RBAC, when striving to minimize the number of roles, is sharply concentrated around its expected value. The result has been achieved by adopting a model reduction and advanced probabilistic tools. Further, we have shown how to apply this result to deal with practical issues in administering RBAC; that is, how it can be used as a stop condition in the quest for the optimum.

5 Devising Meaningful Roles

This chapter copes with the problem of assigning a *business meaning* to roles. To this aim, two different approaches are proposed. First, we introduce a new metric to assess how "good" are roles from a business perspective. Our key observation is that a role is likely to be meaningful from a business perspective when it involves activities within the same business process or organizational units within the same branch. To measure the *spreading* of a role among business processes or organization structure, we resort to centrality indices. Such indices are used in our cost-driven approach during the role mining process. Second, we propose a methodology where the dataset is *decomposed* into smaller subsets that are homogeneous from a business perspective. We introduce some indices that provide, for a given partition, the expected uncertainty in locating homogeneous set of users and permissions that are manageable with the same role. Therefore, by choosing the decomposition with the highest indices values, we most likely identify roles with a clear business meaning. This chapter summarizes the contribution previously published in [5, 10].

5.1 Modeling Business

In this section we shall provide some models to formally describe business information. These models will be later used to formalize the main contribution of this chapter.

5.1.1 Business Activities

Activities (or *tasks*) are a natural way to think about user actions and their contexts. Activities usually arise from the decomposition of the *business processes* of an organization. A business process is a collection of inter-related activities that accomplish a specific goal. From an access control point of view, an activity induces a set of permissions necessary to perform an elementary part of a more complex job. Activities are typically assigned to users on the basis of their job positions [72].

Since the activity and the role concepts are similar in that they both group permissions, one might think that there is no difference between them. However, they have completely different meanings and characteristics. A role focuses on "actors" and places emphasis on *how* the business should be organized, while an activity focuses on "actions" and emphasizes *what* should be done. For example, a typical workflow management system requires that each actor should complete certain tasks. In this case, each task may be performed by several actors with different business roles. It would be ineffective to assign an RBAC role to each task. Neither should an activity be considered a "sub-role": role-permission relationships are identified with different rationales from activity-permission relationship identification.

It is important to highlight that defining and maintaining the activity model up to date can increase the workload of business users within the company. However, a good understanding of the organization is a mandatory requirement when implementing an access control management framework. Therefore, the activity model is already available within an organization.

In the following we formally describe the activity concept.

Activity Tree Decomposing the business processes of an organization usually results in an activity tree structure. For the sake of simplicity, we do not make a formal distinction between business processes and activities. In particular:

- The set $ACTVT$ contains all activities.
- The set $ACTVT\text{-}H \subseteq ACTVT \times ACTVT \times \mathbb{R}$ defines a partial order on the hierarchy tree. The pair $\langle a_p, a_c, w\rangle \in ACTVT\text{-}H$ indicates that the activity a_p is the parent of the activity a_c, whereas w is the *weight* of the connection between a_p and a_c (see below). The existence of the tuple $\langle a_p, a_c, w\rangle$ in $ACTVT\text{-}H$ may alternatively be indicated as $a_c \xrightarrow{w} a_p$. The simplified notation $a_c \rightarrow a_p$ can also be used when w is always 1.
- $\forall a \in ACTVT$, the activity a has only one direct parent, namely $\forall a_p, a_c \in ACTVT : a_c \rightarrow a_p \implies \nexists a'_p \in ACTVT : a'_p \neq a_p, a_c \rightarrow a'_p$.

- The activity tree has only one root.

Given a pair $a_p, a_c \in ACTVT$, the ordering operator $a_c \succeq a_p$ indicates the existence of a hierarchical pathway of "→" from a_c to a_p. Note that, without loss of generality, it is always possible to identify a unique root for a set of activities; for example, a virtual activity that "collects" all the high level activities of the organization can always be defined.

Given an activity $a \in ACTVT$, the following sets can be defined out of convenience:

- $\uparrow a = \{a' \in ACTVT \mid a \succeq a'\}$ represents all possible parents of a. Since each activity has only one direct parent in the tree, $\uparrow a$ contains the path from a to the root of *ACTVT-H*. Note that $|\uparrow a|$ is the length of this path. Given a pair $a_1, a_2 \in ACTVT$, the value of $|\uparrow a_1 \cap \uparrow a_2|$ represents the length of the path from the root to the "nearest" common parent of both a_1, a_2.
- $\downarrow a = \{a' \in ACTVT \mid a' \succeq a\}$ represents all possible children of a.

Each activity is supported by sets of permissions which allow the activity to be performed. To execute a given activity, a user must have *all* the permissions associated to it. This concept can be formalized as follows:

- The set $ACTVT\text{-}A \subseteq ACTVT \times PERMS$ expresses the origin of a permission in a given activity.
- The function $actvt\_perms$: $ACTVT \to 2^{PERMS}$ provides the set of permissions associated to an activity. Given $a \in ACTVT$, it can be formalized as: $actvt\_perms(a) = \{p \in PERMS \mid \exists \langle a, p\rangle \in ACTVT\text{-}A\}$.
- The function $actvt\_perms^*$: $ACTVT \to 2^{PERMS}$ provides all the permissions assigned to a and its children, namely it takes into account the activity breakdown structure. Given $a \in ACTVT$, it can be formalized as: $actvt\_perms^*(a) = \{p \in PERMS \mid \exists a' \in \downarrow a : \langle a', p\rangle \in ACTVT\text{-}A\}$.

A permission can belong to multiple activities. Moreover, given the activity pair $a_1, a_2 \in ACTVT$ such that $a_1 \succeq a_2$ and $p \in PERMS$, if $\langle p, a_1\rangle \in ACTVT\text{-}A$ then we require that $\langle p, a_2\rangle \notin ACTVT\text{-}A$ since a_2 inherits p from its child.

Connection Weights Assigning a weight to connections between activities is a flexible way to model the business process break-down structure. In particular, weights indicate if there is a strong or a weak decomposition. For example, there is likely to be a large weight value between the root activity and activities such as "Human Resources Management" or "Inventory Management". Conversely, the weight between the parent activity "Customer Data

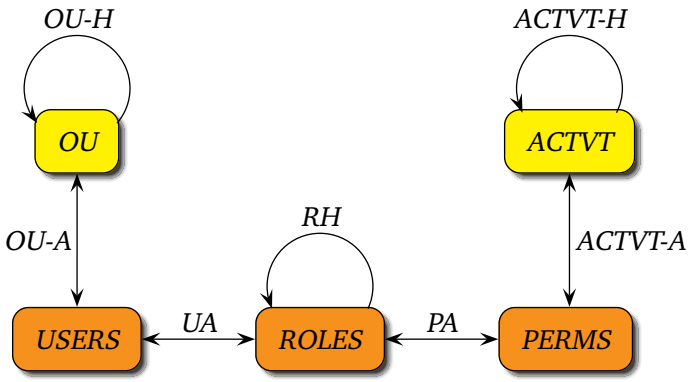


Figure 5.1 Relationships among activities, organization units, and RBAC entities.

Management" and its child "Customer Data Update" is likely to be weak. How weights are derived depends on the given organization, however this topic is not further analyzed in this chapter.

The given weight definition can easily be extended to the partial order as the sum of weights along the path between activities. Given $a_c, a_p \in ACTVT$: $a_c \succeq a_p$, the weight of the path between them is:

$$w_{\text{actvt}}(a_c \succeq a_p) = \sum_{\omega \in \Omega} \omega,\ \Omega = \{\omega \in \mathbb{R} \mid \forall a, a' \in ACTVT, \exists \langle a, a', \omega \rangle \in ACTVT\text{-}H : a_c \succeq a',\ a \succeq a_p\}. \quad (5.1)$$

Figure 5.1 gives a graphical representation of the aforementioned entities and the interaction with the RBAC entities. It also depicts other elements described in next section.

5.1.2 Organization Units

An *organization unit* (OU) is a group of employees which collaborate to perform certain business tasks. The organizational structure is designed by the top management and defines the lines of authority and the division of work. A typical organization is organized as a tree structure, represented by an organization chart. From an access control point of view, it is likely that users of the same OU have the same access permissions. For this reason, users usually have roles located within the organization units [74].

There are many examples of benefits related to the introduction of the organization structures into the access control model. For instance, OUs are used in various frameworks as a means to identify user pools [73,74]. In some

of these works, roles can be assigned directly with OUs instead of individually with users. In this chapter, we prefer to directly assign users to roles, thus allowing a complete delegation of organization unit maintenance to HR.

The following is a formal description of the organization unit structure, that reflects the notation used in Section 5.1.1.

Organization Unit Tree Organization units can usually be represented in a tree structure. In particular:

- The set OU contains all organization units.
- The set $OU\text{-}H \subseteq OU \times OU \times \mathbb{R}$ defines a partial order on the hierarchy tree. The pair $\langle o_p, o_c, w\rangle \in OU\text{-}H$ indicates that the organization unit o_p is the parent of the organization unit o_c, whereas w is the *weight* of the connection between o_p and o_c (see below). The existence of the tuple $\langle o_p, o_c, w\rangle$ in $OU\text{-}H$ may alternatively be indicated as $o_c \xrightarrow{w} o_p$. The simplified notation $o_c \to o_p$ can also be used when w is always 1.
- $\forall a \in OU$, the organization unit a has only one direct parent, namely $\forall o_p, o_c \in OU : o_c \to o_p \implies \nexists a'_p \in OU : a'_p \neq o_p, o_c \to a'_p$.
- The organization unit tree has only one root.

Given a pair $o_p, o_c \in OU$, the ordering operator $o_c \succeq o_p$ indicates the existence of a hierarchical pathway of "$\to$" from o_c to o_p. Note that, without loss of generality, it is always possible to identify a unique root for the organization units—namely, a unit representing the entire organization. Given an organization unit $o \in OU$, we define:

- $\uparrow o = \{o' \in OU \mid o \succeq o'\}$ represents all possible parents of o. It contains the path from o to the root of $OU\text{-}H$, thus $|\uparrow o|$ is the length of this path. Given $o_1, o_2 \in OU$, $|\uparrow o_1 \cap \uparrow o_2|$ is the length of the path from the root to the "nearest" common parent of both o_1, o_2.
- $\downarrow o = \{o' \in OU \mid o' \succeq o\}$ is the set of all children of o.

Each organization unit contains a sets of users. This concept can be formalized as follows:

- The set $OU\text{-}A \subseteq OU \times USERS$ expresses the origin of a user in a given organization unit.
- The function ou_users: $OU \to 2^{USERS}$ provides the set of users belonging to an organization unit. Given an organization unit $o \in OU$, it can be formalized as: $ou\_users(o) = \{u \in USERS \mid \exists \langle o, u\rangle \in OU\text{-}A\}$.

- The function $ou\_users^*: OU \rightarrow 2^{USERS}$ provides all the users belonging to o and its children, namely it takes into account the organization unit breakdown structure. Given $o \in OU$, it can be formalized as: $ou\_users^*(o) = \{u \in USERS \mid \exists o' \in \downarrow o : \langle o', u\rangle \in OU\text{-}A\}$.

Usually, each user belongs to only one organization unit. Hence, given $o_1, o_2 \in OU$ and $u \in USERS$, if $\langle u, o_1\rangle \in OU\text{-}A$, then $\langle u, o_2\rangle \notin OU\text{-}A$.

Connection Weights Weighting connections between OUs is a new concept when compared to the existing framework related to OUs. It is a more flexible way to model the organization break-down structure. For example, user sets can be divided into various *administrative domains* represented by organization unit branches [73]. It is assumed that each domain is independently administered. In such a case, weights may indicate whether domains are loosely or tightly coupled. Another case is when OUs are decomposed into a set of branches to model geographic areas; this often represents a weak partitioning since there are no big differences among users across different geographic areas. Decomposing a project in various working teams is another example of weak partitioning, since all users work for the same objectives. Conversely, domains represented by business units are more important as they usually identify users assigned with completely different jobs. Therefore, this is an example of strong OU partitioning.

The weight concept can be easily extended to the partial order as the sum of all weights between units along the shortest path between them. Given $o_c, o_p \in OU : o_c \succeq o_p$, the weight of the path between them is:

$$w_{\text{ou}}(o_c \succeq o_p) = \sum_{\omega \in \Omega} \omega,\ \Omega = \{\omega \in \mathbb{R} \mid \forall o, o' \in OU, \exists \langle o, o', \omega\rangle \in OU\text{-}H : o_c \succeq o',\ o \succeq o_p\}. \quad (5.2)$$

Figure 5.1 gives a representation of the aforementioned entities and the interaction with standard RBAC entities.

5.2 Measuring the Meaning of Roles

In this section we demonstrate how the business processes model (see Chapter 3) and the organization structure model (see Section 5.1.2) can be leveraged to evaluate the business meaning of roles elicited by role mining algorithms. We define a *metric* for evaluating the business meaning of candidate role-sets. This metric is used during the role mining algorithm execution to establish which roles should be included in the candidate role-set. The key insight is that a role is likely to be meaningful from a business perspective when:

it involves activities within the same business process; or, it involves users belonging to the same organization structure branch. To measure the "spreading" of a role among business processes or organization units we resort to *centrality*, a well-known concept in graph theory. In particular, we define two different centrality indices, namely the *activity-spread* and the *organization-unit-spread*. Leveraging our cost-driven approach (see Chapter 3) makes it feasible to take into account the proposed indices during the role mining process. Finally, we demonstrate the effectiveness of our proposal through a few examples and applications to real data. To measure the business meaning of roles, we introduce the following indices:

- *activity-spread* (detailed in Section 5.2.2) that measures the "dispersion" of business activities that are enabled by permissions assigned to a role;
- *organization-unit-spread* (detailed in Section 5.2.3) that measures the "dispersion" of organization units that users assigned to roles belong to.

The reason why we resort to business processes and organization structure is quite simple. Given $r \in ROLES$, it identifies both a set of permissions ($auth\_perms(r)$) and a set of users ($auth\_users(r)$). Hence, both these sets should be analyzed in order to evaluate the "quality" of the candidate role. Regarding the user set, the structure of the organization is probably the most significant business-related information that is always available in every medium to large sized organization. As a matter of fact, it is usually found within the HR-related IT systems. Similarly, business activities represent the main justification for the adoption of IT applications within a company. Indeed, each application is usually introduced to support business activities. Usually, the business activity tree can be provided by business staff.

Once activity-spread and organization-unit-spread indices are defined, we propose to adopt our cost-driven approach to take them into account during the role mining process. In particular, leveraging the cost function concept makes it possible to combine such indices with other "cost" elements in a single metric (function) to be used for the evaluation of elicited roles during the role mining process. Minimizing such a function means eliciting those roles that contextually minimize the overall administration effort and fit the needs of an organization from a business perspective. This approach is particularly suitable for role mining algorithms since it makes it possible to introduce business elements within the analyzed data, thus leading to a hybrid role engineering approach. Note though that identifying cost elements and then combining them in a cost function is something that can be done in several ways. This, however, is a completely separate subject of research and it is only marginally addressed here. Instead, we mainly focus on the formal description of business elements that can contribute in defining a suitable cost function.

5.2.1 Farness

We first introduce a tool that is particularly useful in topology and related areas in mathematics. This tool will be adapted to our analysis in order to consider business information during the role mining process. Given a graph, we usually refer to a sequence of vertices connected by edges as a *walk*. We also refer to a walk with no repeated vertices as a *path*, while the shortest path between two vertices is referred to as a *geodesic*. These concepts make it possible to introduce the *farness* index, namely a quantity related to the lengths of the geodesics from one vertex to every other vertex in the graph. Vertices that "tend" to have long geodesic distances to other vertices within the graph have higher farness. In the literature it is more common to find another index in place of farness, that is the *closeness* index. Closeness is the inverse of farness, so they can be considered perfectly equivalent. Both farness and closeness are examples of *centrality* measures [95].

The farness index that is used in this chapter can be defined in every metric space where a notion of distance between elements is defined. Given a vertex v_j, its farness f is

$$f = \frac{\sum_{i=1}^{n} d(i,j)}{n-1}, \tag{5.3}$$

where $d(i,j)$ is the distance between the vertices v_i and v_j.

Other centrality measures such as *betweenness*, *degree*, and *eigenvector* centrality [95] might be applicable to our analysis. Since farness is the most suitable and simple one for our analysis, we omit discussions of the others.

5.2.2 Activity-Spread

We now describe an index intended to evaluate business meaning. It is based on the analysis of business processes and their decomposition into activities. For this purpose, we observe that a typical organization is unlikely to have users that perform activities derived from different business processes. For example, given the processes "Human Resources Management" and "Inventory Management", it is very difficult that the organization needs a role that simultaneously allows activities within both these processes. This observation might also be applicable to the decomposition of a business process into simpler activities. It is difficult to have the same users involved in "Training and development" and "Recruitment" of employees, even though they are both activities of "Human Resources Management". However, this constraint becomes weaker as we compare simpler activities; for example, "Screening" and "Se-

lection" might be two possible activities of "Recruitment", but it now becomes possible for a single user to perform both of them.

In general, given a role and the activities involved with it, the basic idea is that a role is likely to have a business meaning when such activities are "close" to one another within the process break-down structure. According to this, "Screening" and "Selection" are close since they are both children activities of "Recruitment"; instead, "Screening" is far from any other activity below "Inventory Management".

To measure the spreading of a role within the activity tree we resort to the farness concept introduced in Section 5.2.1. Given a role, farness may be used to evaluate the distances among activities granted by the role. After having calculated the farness index for each involved activity, averaging such indices offers a metric to capture the "degree of spread" of the role among its activities. We refer to such an index as *activity-spread* of a role. The higher the activity-spread is, the less business meaning the role has.

Distance Function Before describing the activity-spread index in a formal way, we need to introduce the *distance* among two activities $a, a' \in ACTVT$ as:

$$d_{\text{actvt}}(a, a') = w_{\text{actvt}}(a \succeq \bar{a}) + w_{\text{actvt}}(a' \succeq \bar{a}),$$
$$\bar{a} = \{\alpha \in (\uparrow a \cap \uparrow a') \mid \forall \alpha' \in (\uparrow a \cap \uparrow a') : \alpha \succeq \alpha'\}, \quad (5.4)$$

whereas $\bar{a}$ is the nearest common parent of a, a'. If weights between activities are always equal to 1, the following alternative distance definition can be given:

$$d_{\text{actvt}}(a, a') = |\uparrow a| + |\uparrow a'| - 2\,|\uparrow a \cap \uparrow a'|. \quad (5.5)$$

Since activities are organized as a tree with a unique root, Equation (5.5) represents the number of edges between a and a'. Indeed, the distance between two vertices is the number of edges in the geodesic connecting them. Furthermore, it can be easily shown that the activity set is a metric space, since the provided function d_{actvt} is a metric; that is, given activities $a, a', a'' \in ACTVT$ the following properties holds:

- Distance is positive between two different activities, and precisely zero from an activity to itself, namely: $d_{\text{actvt}}(a, a') \geq 0$, and $d_{\text{actvt}}(a, a') = 0 \iff a = a'$.
- The distance between two activities is the same in either directions, namely: $d_{\text{actvt}}(a, a') = d_{\text{actvt}}(a', a)$.
- The distance between two activities is the shortest one along any path, namely: $d_{\text{actvt}}(a, a'') \leq d_{\text{actvt}}(a, a') + d_{\text{actvt}}(a', a'')$ (*triangle inequality*).

Activity-Spread Formalization Given a role $r \in ROLES$, we identify the set of activities allowed by the role r as

$$\mathcal{A}_r = \{a \in ACTVT \mid \nexists a' \in (\downarrow a) \setminus \{a\}, auth\_perms(r) \supseteq actvt\_perms^*(a), \\ auth\_perms(r) \supseteq actvt\_perms^*(a')\}. \quad (5.6)$$

Equation (5.6) requires that *all* the permissions needed to execute the activities of $\mathcal{A}_r$ must be assigned to the role r. Thus, $\mathcal{A}_r$ contains only those activities that are allowed by assigning a user just to the role r. Further, given $a \in \mathcal{A}_r$ none of the parents of a are contained in $\mathcal{A}_r$, that is $\mathcal{A}_r$ contains only activities that are farther from the root.

We therefore define the *activity farness* for $a \in \mathcal{A}_r$ as:

$$d_{\mathcal{A}_r}(a) = \frac{1}{|\mathcal{A}_r| - 1} \sum_{a' \in \mathcal{A}_r} d_{\text{actvt}}(a, a'). \quad (5.7)$$

Equation (5.7) directly derives from Equation (5.3) (see Section 5.2.1) by adopting the distance function d_{actvt}. The greater $d_{\mathcal{A}_r}(a)$ is, the farther a is from all other activities allowed by permissions assigned to r.

Now we define a metric that takes into consideration the farness generated among *all* activities in $\mathcal{A}_r$. To this aim, the *variance* concept is likely to be the most intuitive and suitable to use. Given the arithmetic mean of all the farness indices calculated over $\mathcal{A}_r$, namely

$$\bar{d}_{\mathcal{A}_r} = \frac{1}{|\mathcal{A}_r|} \sum_{a \in \mathcal{A}_r} d_{\mathcal{A}_r}(a) = \frac{1}{|\mathcal{A}_r|(|\mathcal{A}_r| - 1)} \sum_{a, a' \in \mathcal{A}_r} d_{\text{actvt}}(a, a'), \quad (5.8)$$

we define the *activity-spread* of a role $r \in ROLES$ as the farness variance among all the activities in $\mathcal{A}_r$, that is:

$$actvt\_spread(r) = \left(\frac{1}{|\mathcal{A}_r|} \sum_{a \in \mathcal{A}_r} d^2_{\mathcal{A}_r}(a) \right) - \bar{d}^2_{\mathcal{A}_r}. \quad (5.9)$$

Note that the value of $actvt\_spread(r)$ should be adjusted in order to assign a higher value to those roles associated with permissions that do not allow the execution of any activity. This way, it is possible to "dissuade" a role engineering process from eliciting roles where the corresponding permissions are not associated to any activities. Indeed, when a role contains permissions that are not related to any activities, it becomes harder to identify a business meaning for it. The number of permissions not related to activities are:

$$\left| auth\_perms(r) \setminus \bigcup_{a \in \mathcal{A}_r} actvt\_perms(a) \right|. \quad (5.10)$$

If Equation (5.10) is equal to 0, there is no need to adjust $actvt\_spread(r)$. Otherwise, $actvt\_spread(r)$ may be multiplied by a coefficient that is proportional to Equation (5.10).

5.2.3 Organization-Unit-Spread

We now describe an index to evaluate business meaning of roles by analyzing the organization unit structure. With this aim in mind, note that users located into different OUs are likely to perform different tasks. Moreover, a role used across multiple organization units may require the co-ordination of various administrators; thus, requiring a higher administration effort than one for roles used exclusively within a single organizational unit. Hence, the idea is that the more distant the involved organization units are from each other, the less business meaning the role has. The ideal situation is when a role is almost exclusively assigned with users belonging to the "most central" OUs. Similar to the previous section, we calculate a farness index for each OU involved by a role—namely those OUs which contain the users assigned to this role. After having calculated the farness index for each involved OU, averaging such indices offers a metric to capture the "degree of spread" of the role among OUs. We refer to such an index as *organization-unit-spread* of a role. The higher the organization-unit-spread is, the less business meaning the role has.

Distance Function Before formally describing the organization-unit-spread index, we need to introduce the *distance* among two organization units $o, o' \in OU$ as:

$$d_{\text{ou}}(o,o') = w_{\text{ou}}(o \succeq \bar{o}) + w_{\text{ou}}(o' \succeq \bar{o}),\ \bar{o} = \{\omega \in (\uparrow o \cap \uparrow o') \mid \forall \omega' \in (\uparrow o \cap \uparrow o') : \omega \succeq \omega'\}, \quad (5.11)$$

whereas $\bar{o}$ is the nearest common parent of o, o'. If weights between organization units are always equal to 1, the following alternative distance definition can be given:

$$d_{\text{ou}}(o,o') = |\uparrow o| + |\uparrow o'| - 2|\uparrow o \cap \uparrow o'|. \quad (5.12)$$

Since organizational units are organized as a tree with a unique root, Equation (5.12) represents the number of edges between o and o'. Furthermore, given $o, o', o'' \in OU$, for both distance definitions it can be demonstrated that:

- The distance is positive between two different organization units, and precisely zero from an organization unit to itself: $d_{\text{ou}}(o,o') \geq 0$, and $d_{\text{ou}}(o,o') = 0 \iff o = o'$.
- The distance between two organization units is the same in either direction: $d_{\text{ou}}(o,o') = d_{\text{ou}}(o',o)$.
- The distance between two organization units is the shortest one along any path: $d_{\text{ou}}(o,o'') \leq d_{\text{ou}}(o,o') + d_{\text{ou}}(o',o'')$ (*triangle inequality*).

Organization-Unit-Spread Formalization Note that the number of users supporting each organization unit can influence the administration cost of roles. Indeed, bigger OUs require more effort from a single administrator or perhaps require multiple administrators. For this reason, the spreading index must be influenced both by the organization structure and by the percentage of users assigned to a given role and contextually belonging to the same organization unit.

Given a role $r \in ROLES$, we define the following sets:

- The set $\mathcal{O}_r = \{o \in OU \mid \nexists o' \in (\downarrow o) \setminus \{o\}, auth\_users(r) \supseteq ou\_users^*(o), auth\_users(r) \supseteq ou\_users^*(o')\}$, namely the set of organization units being involved with the role r.
- The set $\mathcal{U}_r = auth\_users(r)$, namely the set of users assigned to r.
- Given $o \in \mathcal{O}_r$, then $\mathcal{U}_o = ou\_users^*(o)$ is the set of users assigned to an organization unit o.

Given $o \in \mathcal{O}_r$ none of the parents of o is contained in $\mathcal{O}_r$, that is $\mathcal{O}_r$ contains only OUs that are farther from the root. We therefore define the *OU farness* index for $o \in \mathcal{O}_r$ as:

$$d_{\mathcal{O}_r}(o) = \frac{1}{|\mathcal{U}_r|} \sum_{o' \in \mathcal{O}_r} |\mathcal{U}_{o'} \cap \mathcal{U}_r| \, d_{\text{ou}}(o, o'). \tag{5.13}$$

We assume that $(1/|\mathcal{U}_r|) \sum_{o \in \mathcal{O}_r} |\mathcal{U}_o \cap \mathcal{U}_r| = 1$, namely organization units in $\mathcal{O}_r$ contain all the users assigned to r and no user simultaneously belongs to more than one OU. The greater $d_{\mathcal{O}_r}(o)$ is, the farther o is from OUs containing the majority of users assigned to r. For example, if there is one particular OU containing most of the users of the role, then $d_{\mathcal{O}_r}(o)$ is close to the distance between o and such an OU.

Now we define the *variance* of all farness indices related to organization units in $\mathcal{O}_r$. Given the weighted arithmetic mean of all the farness indices calculated upon $\mathcal{O}_r$, namely

$$\bar{d}_{\mathcal{O}_r} = \frac{1}{|\mathcal{U}_r|} \sum_{o \in \mathcal{O}_r} |\mathcal{U}_o \cap \mathcal{U}_r| \, d_{\mathcal{O}_r}(o) = \frac{1}{|\mathcal{U}_r|^2} \sum_{o, o' \in \mathcal{O}_r} |\mathcal{U}_o \cap \mathcal{U}_r| |\mathcal{U}_{o'} \cap \mathcal{U}_r| \, d_{\text{ou}}(o, o'), \tag{5.14}$$

the *organization-unit-spread* is defined as

$$ou\_spread(r) = \left(\frac{1}{|\mathcal{U}_r|} \sum_{o \in \mathcal{O}_r} |\mathcal{U}_o \cap \mathcal{U}_r| \, d^2_{\mathcal{O}_r}(o) \right) - \bar{d}^2_{\mathcal{O}_r}. \tag{5.15}$$

Note that the value of $ou\_spread(r)$ grows when: OUs contain many users—identified through $|\mathcal{U}_o \cap \mathcal{U}_r|$; or, OUs are far from each other—according to $d^2_{\mathcal{O}_r}(o)$.

5.2.4 Revising the Cost Function

In this section we briefly explain how cost elements can be combined in a global cost function, leveraging our cost-driven approach. In general, finding the optimal candidate role-set can be seen as a *multi-objective optimization problem* [33]. An optimization problem is multi-objective when there are a number of objective functions that are to be minimized or maximized. Multi-objective optimization often means to trade-off conflicting goals. In a role engineering context, possible objectives are:

- Minimize the number of roles;
- Minimize the number of role possessed by each user;
- Maximize the business meaning of roles, that corresponds to minimizing the activity-spread and/or the organization-unit-spread indices of all elicited roles.

The previous statements can be formalized as:

- $\min\{|ROLES|\}$;
- $\min\left\{\sum_{r\in ROLES} ass\_users(r)\right\} = \min\{|UA|\}$;
- $\min\left\{\sum_{r\in ROLES} actvt\_spread(r)\right\}$;
- $\min\left\{\sum_{r\in ROLES} ou\_spread(r)\right\}$.

The constraint of this optimization problem is that elicited roles must "cover" all possible combinations of permissions, namely they represent a candidate role-set.

The dependencies among these objectives are quite complex. For example, if on one side $|ROLES|$ decreases, there is a strong chance that more roles will be required to cover all the permissions possessed by users, causing $|UA|$ to increase. Since each role has a high number of assigned users, it is likely that the number of involved organization units is high, thus increasing the value of the *ou_spread* function. On the other hand, if we want to reduce the average number of roles per user, we will need more *ad-personam* roles, then $|ROLES|$ will likely increase.

Hence, there is little use in the quest for a global maximum or a global minimum. Since trade-off for conflicting criteria can be defined in many ways, there exist multiple approaches to define what an optimum is. The simplest approach is that of computing a *weighted sum* [33]: multiple objectives are transformed into an aggregated scalar objective function by multiplying each objective by a weighted factor and summing up all contributors. As for role

engineering, we can combine all the proposed objectives as follows:

$$\min \left\{ w_r |ROLES| + w_u |UA| + w_a \sum_{r \in ROLES} actvt\_spread(r) + w_o \sum_{r \in ROLES} ou\_spread(r) \right\}. \tag{5.16}$$

where the w_i indicate the importance of the i^{th} objective in the overall optimization problem. Section 5.3.2 will show a real application of Equation (5.16) to a role mining algorithm.

5.3 Spread Index in Action

In this section we will analyze the distinguishing features of the proposed framework through some practical examples. We first show a use case of the activity-spread index via a simple example. Then, we demonstrate the effectiveness of the organization-unit-spread index through the RBAM role mining algorithm (see Chapter 3) applied to an existing company. Due to space limitation, we do not provide examples of the combined usage of both indices.

5.3.1 Example of Activity-Spread

Figure 5.2 depicts a possible candidate role set and an activity tree. At the top there are representations of RBAC entities, namely roles and permissions. Example of roles are $ass\_perms(r_1) = \{p_1, p_2\}$, $ass\_perms(r_6) = \{p_3\}$, while we have that $auth\_perms(r_6) = \{p_1, p_2, p_3, p_4\}$ since $r_1 \succeq r_6$ and $r_2 \succeq r_6$. At the bottom of the figure, there are activities organized in a tree structure. For instance, a_{11} is the root of the tree, while $a_9 \rightarrow a_{11}$ and $a_7 \succeq a_{11}$ holds. To ease exposition, weights of connections between activities are not represented in Figure 5.2, and they are all assumed to be equal to 1. Further, $\downarrow a_8 = \{a_3, a_4, a_8\}$ and $\uparrow a_1 = \{a_1, a_7, a_9, a_{11}\}$. Activity a_7 is performed when a user possesses the permissions p_1, p_2 (that allows for the child activity a_1) and p_2, p_3, p_4 (that are related to the child activity a_2).

As for the business meaning estimation, let us calculate the activity-spread of role r_6. By looking at Figure 5.2 it is easy to verify that role r_6 allows for the execution of the activities $\mathcal{A}_{r_6} = \{a_1, a_2\}$. In line with Equation (5.7), the farness index for each activity in $\mathcal{A}_{r_6}$ is:

$$d_{\mathcal{A}_{r_6}}(a_1) = \frac{d_{\text{actvt}}(a_1,a_1) + d_{\text{actvt}}(a_1,a_2)}{2-1} = \frac{0+2}{2-1} = 2,$$
$$d_{\mathcal{A}_{r_6}}(a_2) = \frac{d_{\text{actvt}}(a_2,a_1) + d_{\text{actvt}}(a_2,a_2)}{2-1} = \frac{2+0}{2-1} = 2.$$

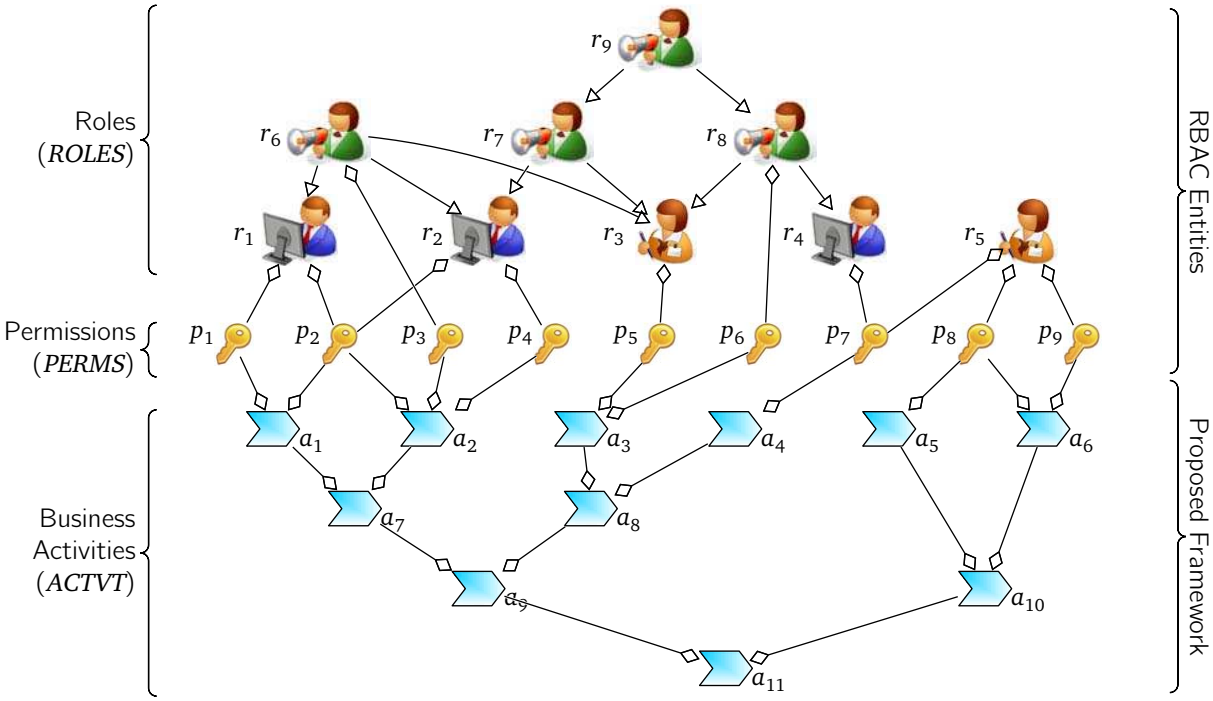


Figure 5.2 An example of activity model

For such indices we have $\bar{d}_{\mathcal{A}_{r_6}} = \frac{2+2}{2} = 2$ and thus:

$$actvt\_spread(r_6) = \frac{2^2+2^2}{2} - 2^2 = 0.$$

The activity spread is 0 since the activities associated to r_6 are somewhat "balanced" within the tree. Thus, r_6 is probably a good role; indeed, it allows the execution of activities a_1, a_2, and consequently the execution of a_7.

A different result is obtained by analyzing role r_5. In this case, $\mathcal{A}_{r_5} = \{a_4, a_5, a_6\}$. Then:

$$d_{\mathcal{A}_{r_5}}(a_4) = \frac{d_{\text{actvt}}(a_4,a_4)+d_{\text{actvt}}(a_4,a_5)+d_{\text{actvt}}(a_4,a_6)}{4-1} = \frac{0+5+5}{3-1} = 5,$$
$$d_{\mathcal{A}_{r_5}}(a_5) = \frac{d_{\text{actvt}}(a_5,a_4)+d_{\text{actvt}}(a_5,a_5)+d_{\text{actvt}}(a_5,a_6)}{4-1} = \frac{5+0+2}{3-1} = \frac{7}{2},$$
$$d_{\mathcal{A}_{r_5}}(a_6) = \frac{d_{\text{actvt}}(a_6,a_4)+d_{\text{actvt}}(a_6,a_5)+d_{\text{actvt}}(a_6,a_6)}{4-1} = \frac{5+2+0}{3-1} = \frac{7}{2}.$$

Notice that a_4 has the largest farness, hence it is far from other activities. These observations can also be graphically justified by observing Figure 5.1. The arithmetic mean of such indices is $\bar{d}_{\mathcal{A}_{r_5}} = \frac{5+7/2+7/2}{3} = 4$ and thus:

$$actvt\_spread(r_5) = \frac{(5)^2+(7/2)^2+(7/2)^2}{3} - 4^2 = 0.5.$$

This means that r_5 has less business meaning than r_6, since r_5 is more spread out among its activities.

Finally, it is worth noticing that both the farness index and the activity-spread make up useful information that could also be visualized in a role engineering tool for each activity that is involved within a given role. Having access to this information could help users validate roles from a business perspective.

5.3.2 Organization-Unit-Spread on Real Data

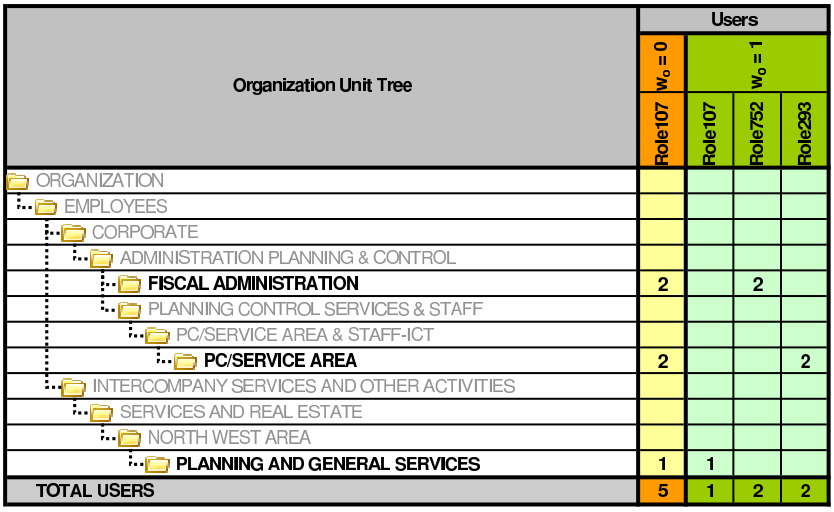
To highlight the framework viability in real applications, we examined a large private company. In particular, we analyzed permissions related to an ERP application. The system configuration was made up of 1,139 permissions that were used by 1,034 users within 231 organization units, resulting in 10,975 user-permission assignments.

To test the effectiveness of our approach, we used an improved version of the RBAM algorithm (see Chapter 4 for further details), seeking to elicit roles which minimize the function

$$c = w_r|ROLES| + w_u|UA| + w_o \sum_{r \in ROLES} ou\_spread(r). \tag{5.17}$$

The algorithm was set to discard permission combinations possessed by less than 4 users. Moreover, we assigned a weight 1 to all direct hierarchical relationships between organization units. Then, we compared the algorithm output obtained in two distinct settings: in the first one (from now on indicated as "Experiment 1"), we did not considered the contribution of the organization-unit-spread index, by using $w_o = 0$ and $w_r = 10$, $w_u = 1$; in the second one (from now on indicated as "Experiment 2"), we considered all the cost elements provided by Equation (5.17), using $w_o = 1$ and the same values for the other weights, namely $w_r = 10$, $w_u = 1$.

The first thing that we can observe from Experiment 2 is that the number of elicited roles is greater than the number of elicited roles in Experiment 1. In particular, we have 157 roles in the first case (that cover 7,857 user-permission assignments with 2,191 role-user relationships) and 171 roles in the second (that cover 7,857 user-permission assignments with 2,196 role-user relationships). The justification for this behavior is that when using $w_o \neq 0$ (namely, taking into account the organization-unit-spread) in some cases it is necessary to reduce the number of users assigned to a role by introducing additional roles to be assigned with a subset of such users. Indeed, the greater the number of users assigned to a role are, the more organization units are likely to



Organization Unit Tree	Users			
	$w_o = 0$	$w_o = 1$		
	Role107	Role107	Role752	Role293
ORGANIZATION				
EMPLOYEES				
CORPORATE				
ADMINISTRATION PLANNING & CONTROL				
FISCAL ADMINISTRATION	2		2	
PLANNING CONTROL SERVICES & STAFF				
PC/SERVICE AREA & STAFF-ICT				
PC/SERVICE AREA	2			2
INTERCOMPANY SERVICES AND OTHER ACTIVITIES				
SERVICES AND REAL ESTATE				
NORTH WEST AREA				
PLANNING AND GENERAL SERVICES	1	1		
TOTAL USERS	5	1	2	2

Figure 5.3 An example of elicited roles with different values for w_o

be involved. Hence, according to Equation (5.15), the value of the function *ou_spread* is likely to be higher.

Figure 5.3 shows an interesting example—to protect company privacy, the names of the organization units are slightly different from the original ones. In Experiment 1 ($w_o = 0$), we noticed that 2 out of 1,139 permissions were possessed by 22 users, and these users were granted these permissions by assigning them to only one of the two roles which the algorithm automatically called `role107` and `role594`. Moreover, $\texttt{role}_{594} \succeq \texttt{role}_{107}$, while $|auth\_perms(\texttt{role}_{107})| = 2$ and $|auth\_perms(\texttt{role}_{594})| = 15$. Further, we have that $|ass\_users(\texttt{role}_{107})| = 5$ and users were belonging to 3 different organization units, namely 'FISCAL ADMINISTRATION', 'PC/SERVICE AREA' and 'PLANNING AND GENERAL SERVICES'. Because of the spreading of users among these OUs, $ou\_spread(\texttt{role}_{107}) = 12.243$. Figure 5.3 also offers a partial view of the entire organization structure, showing the already cited organization units with a bold font. Conversely, in Experiment 2 (where $w_o = 1$) the same 2 permissions were completely covered by 4 roles that the algorithm automatically called `role107`, `role752`, `role293`, `role594`. Roles with the same name between the two experiments contain the same permission set, but are assigned with different user sets. Indeed, by observing Figure 5.3 it is possible to verify that the same 5 users from the other experiment now spread among 3 roles, having $|ass\_users(\texttt{role}_{107})| = 1$, $|ass\_users(\texttt{role}_{752})| = 2$, and $|ass\_users(\texttt{role}_{293})| = 2$. In this case,

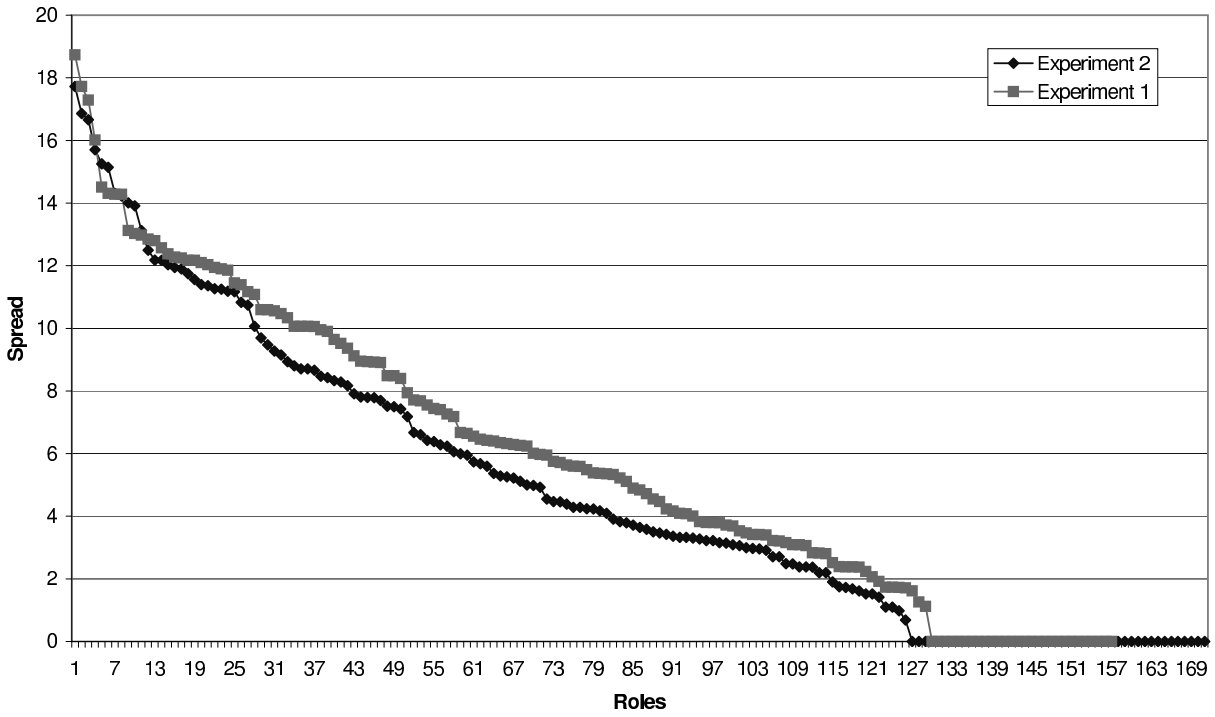


Figure 5.4 Roles obtained via different values of w_o and sorted by spread

each of these roles have assigned users who belong to only one organization unit. Moreover, role752 $\succeq$ role107 and role293 $\succeq$ role107. One role has $|auth\_perms(\texttt{role752})| = 13$ and the other one has $|auth\_perms(\texttt{role293})| = 8$. Their organization-unit-spread is equal to 0, thus demonstrating that taking the organization-unit-spread into account may help identify roles with higher business meaning.

Another interesting behavior of the RBAM algorithm, if used with the cost function described in Equation (5.17), is that increasing the value of w_o causes the decrease of the average organization-unit-spread. Indeed, when $w_o = 0$ we have an average value of $\left(\sum_{r \in ROLES} ou\_spread(r)\right)/|ROLES| = 5.96$. Instead, when $w_o = 1$ the average spread drops down to 4.86, and $w_o = 10$ causes an average spread of 2.87. This means that a high value of w_o allows to elicit roles with a more relevant business meaning on average. In such a case, the price to pay to obtain more meaningful roles is in terms of the cardinality of *ROLES*. By analyzing Figure 5.4, it could also be noted that even though Experiment 2 has more roles, the percentage of roles with spread equal to 0 is increased in comparison with Experiment 1.

5.4 Using Business Model Before Mining Roles

A possible viable solution to value business requirements in role mining may be *to restrict the 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. Suppose we know, from a partial or coarse-grained top-down analysis, that a certain set of users perform the same tasks, but the analysis lacks information about which permissions are required to execute these tasks. In this scenario, restricting role mining techniques to these users only—instead of analyzing the organization as a whole—, will ensure that elicited roles are only related to such tasks. Consequently, it will be easier for an analyst to assign a business meaning to the roles suggested by the bottom-up approach. Moreover, elicitation of roles with no business meaning can be avoided by grouping users that perform similar tasks together first, and then analyzing each group separately. Indeed, investigating analogies among groups of users that perform completely different tasks is far from being a good role mining strategy. 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 [34, 94]. To apply this divide-and-conquer strategy, a lot of enterprise information can be used. Business processes, workflow tasks, and organization unit trees are just a few examples of business elements that can be leveraged. Notice that very often such information is already available in most companies before starting the role engineering task—for instance within HR systems. When dealing with information from several sources, the main problem is thus ascertaining which information induces the partition that improves the role engineering task the most.

To address all the aforementioned issues, this section proposes a methodology that helps role engineers to leverage business information during the role mining process. In particular, we propose to divide the access data into smaller subsets that are homogeneous according to a business perspective, instead of performing a single bottom-up analysis on the entire organization. This eases the attribution of business meaning to roles elicited by any existing role mining algorithm and reduces the problem complexity. To select the business information that induces the most suitable partition, an index referred to as "ENTRUSTABILITY" (***ent****ropy-based* ***r****ole* ***us****efulness predic****tability***) is identified. Rooted on information theory, it measures the expected uncertainty in locating a homogeneous set of users and permissions that can be managed as a whole by a single role. The decomposition with the highest ENTRUSTABILITY

value is the one that most likely leads to roles with a clear business meaning. Several examples illustrate the practical implications of the proposed methodology and related tools, which have also been applied on real enterprise data. Results support the quality and viability of the proposal.

5.4.1 A Divide-And-Conquer Approach

In this section we describe how to condition existing role mining algorithms to craft roles with business meaning and to downsize the problem complexity. By leveraging the observations of the previous section, it is possible to exploit available business information, or top-down analysis results, in order to drive a bottom-up approach. In particular, a business attribute (e.g., organizational units, job titles, applications, tasks, etc.) naturally induces a *partition* of the user-permission assignment set *UP* to analyze, where each subset is made up of all the assignments that share the same attribute values. Decomposing the role mining problem into smaller sub-problems is a best practice, especially when dealing with large datasets. Typical steps of a generic role mining process are [58]:

1. *Choice of Information Sources*. From the available data, a subset has to be selected which is most promising to yield suitable information for role creation.
2. *Data preparation*. The data is collected from various locations, cleaned up from obvious or known as incorrect information, and transformed into a format in which it can then be processed by the data mining software.
3. *Exploration*. This phase is crucial for the whole role mining process. It will provide a "feeling" for the data contents and the expected role scheme. The results of this phase are suitable attribute sets for the unique representation of organizational and functional roles and suitable parameters for the role mining algorithms.
4. *Mining*. The role mining algorithm is performed.
5. *Role creation*. The outcome of the data mining run are used to derive candidate organizational and functional roles.
6. *Check, approval and implementation of resulting roles*. The resulting roles have to be checked for plausibility and correctness.
7. *User Assignment*. The elicited roles are finally assigned to users.

In this scenario, leveraging ENTRUSTABILITY allows for the identification of the business information that "best fits" with the access control data, namely the information that induces a decomposition which most simplifies the identification of a business meaning for roles elicited in the role mining phase. The

index can be calculated in the exploration step and used to guide the subsequent role mining steps.

When several business attributes are at our disposal, the difficulty arises in the selection of the one that induces a partition for *UP* that simplifies the subsequent mining steps. To this end, for each business information we calculate an index referred to as ENTRUSTABILITY, which measures the uncertainty in identifying homogeneous sets of users and permissions that can be managed through a single role. The decomposition with the highest ENTRUSTABILITY value is the one that most likely leads to roles with a clear business meaning. Before formally defining the ENTRUSTABILITY index, in the following, we first 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 describe ENTRUSTABILITY as a measure of how much a partition reduces the uncertainty in locating such sets of users and permissions in each subset of the partition.

5.4.2 Pseudo-Roles

The following definition introduces an important concept of the proposed methodology:

Definition 5.1 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 $perms(u)$.

Pseudo-roles are also described in Chapter 6, with the alternative name of "pseudo-biclusters". In such a chapter we will also discuss pseudo-roles from a graph theory perspective. We will propose a mapping between binary matrices and undirected graphs where a pseudo-role represent all the neighbors of a given node. We will provide efficient algorithms for viable computation of pseudo-roles. In this chapter, pseudo-roles will be employed to identify those user-permission assignments that can be managed together with a given assignment through a single role. Notice that all users $users(p)$ should not necessarily be granted all permissions $perms(u)$—this is the reason for the "pseudo" prefix. 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 5.2 The percentage of user-permission assignments of *UP* that gen-

erates a pseudo-roles $\hat{r}$ is referred to as its *frequency*, defined as:

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

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

Theorem 5.1 *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 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 5.1, $u' \in ass\_users(\hat{r}) = users(p)$ 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 5.1 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.

5.4.3 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 [27] 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 $|\mathcal{R}_* \cap \mathcal{R}_a| \geq 0$, 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 $\Pr(a) = |UP_a| / \sum_{a \in \mathcal{A}} |UP_a|$ 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:

$$\textsc{entrustability}(A) = \frac{I(R;A)}{H(R)} = 1 - \frac{H(R \mid A)}{H(R)}.$$

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

Table 5.1 ENTRUSTABILITY values of the analyzed business information

Attribute	User	Perm	entrustability
Job Title	✓		1.00
Unit	✓		0.93
Cost Center	✓		0.85
Organizational Unit	✓		0.82
Building	✓		0.58
Application		✓	0.49
Division	✓		0.46
Surname	✓		0.02

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.

5.5 Entrustability in Real Cases

We now show an application to a real case. Our case study has been carried out on a large private organization. Due to space limitation, we only report on a representative organization branch that contained 50 users with 31 granted permissions, resulting in a total of 512 user-permission assignments. We adopted several user and permission attributes at our disposal. In order to protect organization privacy, some names reported in this chapter for business attributes are different from the original ones.

According to the proposed approach, we computed the ENTRUSTABILITY index for each available business information. To further demonstrate the reliability of the methodology, we introduced a *control* test. That is, we try to categorize users according to the first character of their surname. Since this categorization does not reflect any access control logic, our methodology reveals that—as expected—partitioning by surname does not help the mining phase. Table 5.1 reports on the outcome of the analysis—it also specifies whether the attributes were used to partition user-permission assignments by users or by permissions. According to the reported values, the "Job Title" information induces the most suitable partition for the attribution of business meaning to roles. As a matter of fact, when ENTRUSTABILITY equals 1, each subset can be managed by just one role. Unsurprisingly, the categorization by surname leads to an ENTRUSTABILITY index that is very close to 0, indicat-

ing that the role engineering task does not substantially change its complexity after decomposition.

To better understand the meaning of the ENTRUSTABILITY values obtained from our analysis, Figure 5.5 depicts user-permission relationships involved with subsets for each partition. In particular, we report on the attribute values that identify each subset, the entropy value $H(R)$ computed for each subset, and a matrix representation of user-permission assignments, where each black cell indicates a user (row) that has a certain permission (column) granted. Figure 5.5(a) visually demonstrates why the Job Title information leads to a value for ENTRUSTABILITY that equals 1. Indeed, in this case all users sharing the same job title always share the same permission set. Therefore, by creating one role for each subset, we provide roles that can straightforwardly be associated with users whenever they join the organization (and get their job title for the first time) or change their business functions (and thus likely change their job title). Another piece of information that induces a good partition is Unit. As is noted from Figure 5.5(b), almost all users within each group share the same permission sets. For example, within the unit "Personal Communication Unit" there is one user (the first one) that has an additional permission granted compared to other users of the same unit. For this reason, the identification of roles needed to manage these users requires a little more investigation—hence, leading to a non-zero entropy value, that is, $H(R) = 0.98$. This example also raises another important point: even though the ENTRUSTABILITY value for Job Title is higher than for Unit, the Unit information induces fewer and larger subsets, hence allowing to cover all user-permission relationships with fewer roles. In general, the smaller the subsets, the more likely it is that the ENTRUSTABILITY index is high. However, small subsets reduce the benefits introduced by RBAC in terms of administration effort, due to the limited number of user-permission relationships that can be managed via a single role. Hence, a trade-off should be reached between ENTRUSTABILITY value and subset dimensions. An alternative approach could be to further partition those subsets that have high entropy values by introducing other pieces of business information. In the previous case, the subset identified by the unit named "CS Operations Unit II" (see Figure 5.5(b)) involves users with two job titles: if we recursively apply our methodology and divide the unit "CS Operations Unit II" according to the Job Unit information, we will obtain an ENTRUSTABILITY value that equals 1. Hence, obtaining larger roles when compared to the partition by Job Title only.

Figure 5.5(g) also demonstrates that not every bit of business information improves the role finding task. Although analyzing the data as a whole is obviously more difficult than analyzing smaller subsets, in this case there are still uncertainties regarding the identification of roles. For instance, it is not triv-

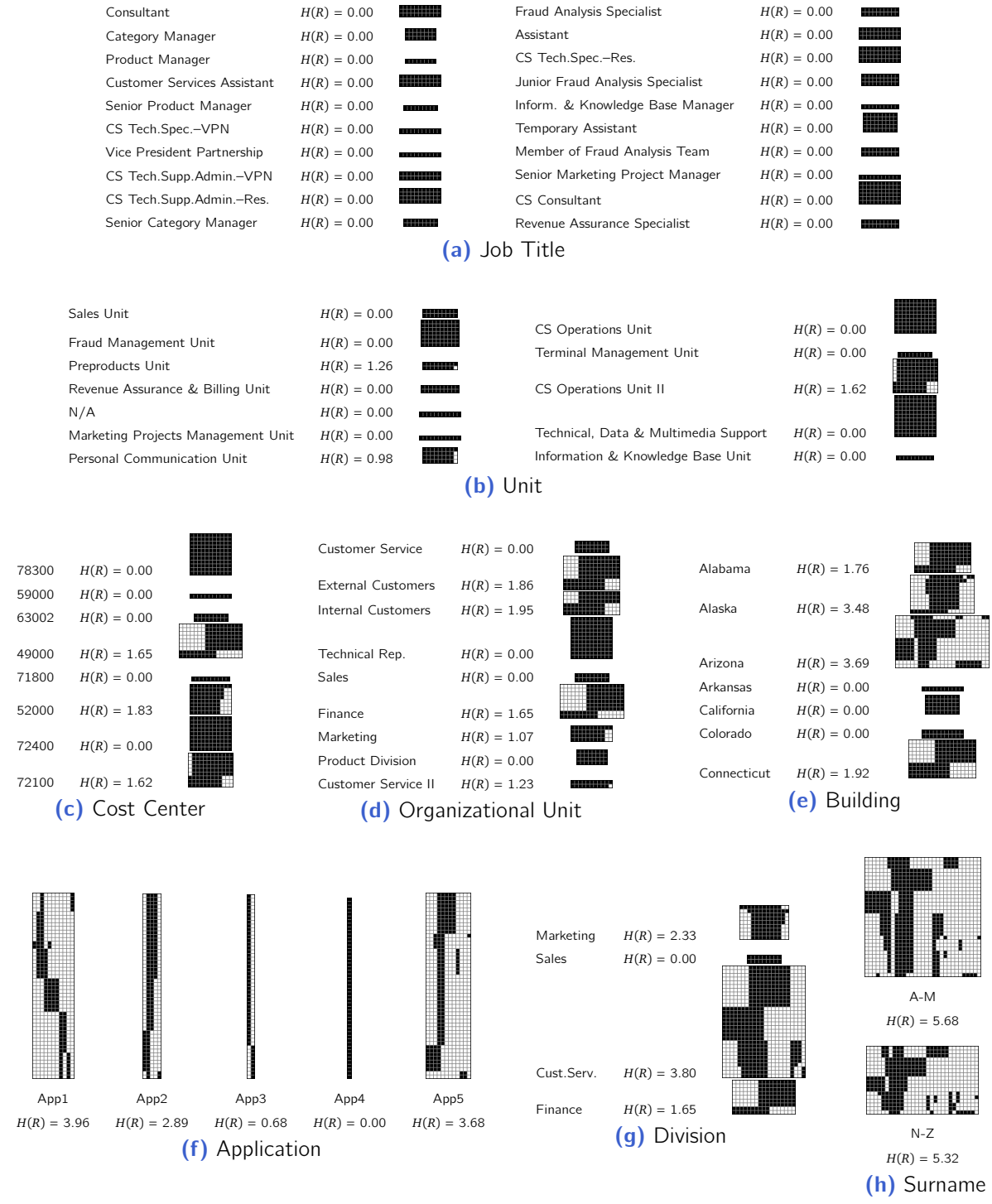
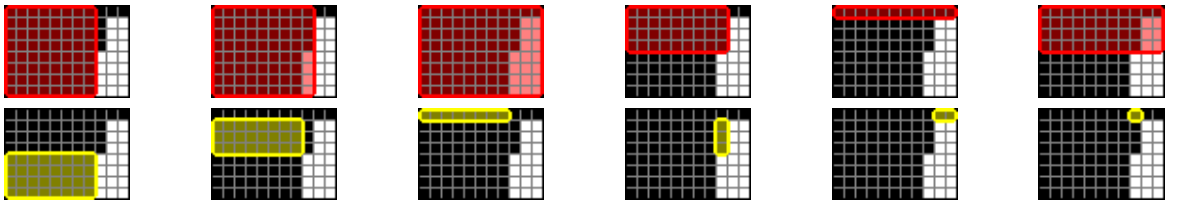


Figure 5.5 Graphical representation of each partition and corresponding entropy values



(a) $\varphi = 32/70$ **(b)** $\varphi = 24/70$ **(c)** $\varphi = 8/70$ **(d)** $\varphi = 3/70$ **(e)** $\varphi = 2/70$ **(f)** $\varphi = 1/70$

Figure 5.6 Pseudo-roles (top figures, highlighted in red) and corresponding generators (bottom figures, highlighted in yellow)

ial to assign a meaning to possible roles—without any further information—within the division "Cust.Serv.", namely the division with the highest entropy value. Finally, Figure 5.5(h) clearly shows that surname information is completely useless. In fact, if we compute the entropy of the entire user-permission assignment, we obtain the value $H(R) = 5.69$. In this case, the entropy values for users "A-M" and "N-Z" are almost the same as before the decomposition.

To conclude, Figure 5.6 depicts all the pseudo-roles that can be identified in a simple case represented by the cost center named "52000" (from Figure 5.5(c)), which numbers 8 users, 11 permissions, and 70 user-permission assignments. Each figure from Figure 5.6(a) to Figure 5.6(f) shows a different pseudo-role. At the top of each figure, a binary matrix shows all the user-permission assignments covered by the pseudo-role (dark red cells are existing assignments covered by the pseudo-role, light red are non-existing assignments). At the bottom, another matrix shows the assignments that generate the pseudo-role (highlighted in yellow). Notice that when the pseudo-role frequency is high (e.g., Figure 5.6(a) and Figure 5.6(b)), it likely contains a role for managing most of the assignments. Conversely, unfrequent pseudo-roles (e.g., Figure 5.6(e) and Figure 5.6(f)) identify assignment sets that are not worth investigating due to the reduced number of assignments that can be managed by a single role.

5.6 Final Remarks

This chapter provides a new formal framework for the role engineering problem. It allows to implement a concrete hybrid approach through the com-

bination of existing role mining algorithms (bottom-up) and business analysis (top-down). Once the business processes of an organization have been correctly modeled, we then analyze roles in order to evaluate their business meaning. In particular, a role is likely to be meaningful when it involves activities within the same business process. Thus, we measure the spreading of a role among business processes by introducing the spread index. Leveraging the cost-driven approach makes it possible to take into account the spread during the role engineering process. The proposed framework has been implemented on a real case, and the results support the its viability.

We also described a methodology that helps role engineers to leverage business information during the role mining process. In particular, we demonstrate that by dividing data into smaller, more homogeneous subsets, it practically leads to the discovery of more meaningful roles from a business perspective, decreasing the risk factor of making errors in managing them. To drive this process, the ENTRUSTABILITY index has been introduced to measure the expected uncertainty in locating homogeneous set of users and permissions that can be managed by a single role. Leveraging this index allows to identify the decomposition that increases business meaning in elicited roles in subsequent role mining steps, thus simplifying the analysis. The quality of the index is also guaranteed by analysis. Several examples, developed on real data, illustrate how to apply the tools that implement the proposed methodology, as well as its practical implications. Those results support both the quality and the practicality of the proposal.

6 Taming Role Mining Complexity

In the following we address the problem of reducing the role mining *complexity* in RBAC systems. The main idea is to identify those *exceptional* assignments (i.e., permissions exceptionally or accidentally granted or denied) which badly bias any role mining analysis. To this aim, we propose a three steps methodology: first, we associate a weight to roles; second, we identify user-permission assignments that cannot belong to roles with a weight exceeding a given threshold; and third, we restrict the role-finding problem to user-permission assignments identified in the second step. We formally show that this methodology allows role engineers for the elicitation of stable candidate roles, by contextually simplifying the role selection task. Furthermore, we also consider the possibility that analyzed data present some *missing* user-permission assignment. Thus, we propose a novel algorithm that is able to impute such missing values in a very efficient and effective way. This chapter summarizes the contribution previously published in [6, 8, 12].

6.1 Stability Concept

To address all the above mentioned issues, this chapter proposes a methodology that helps role engineers: to identify roles that are stable; and, to minimize the effort required to select the most meaningful roles for the organization. The proposed approach allows to *prune* user-permission assignments which lead to unstable roles and that increase the complexity of the role mining task. In this way, we are able to build a core set of roles that have the above men-

tioned features. These results have been formally proven through sound graph theory. In particular, we leverage the mapping between role mining and some well-known graph problems (i.e., biclique cover, clique partition, vertex coloring, and maximal clique enumeration). A further contribution of this chapter is the adoption of the clustering coefficient as a metric to evaluate the role mining complexity. To Furthermore, efficient deterministic and randomized algorithms that implement the proposed pruning approach are also described. A thorough analysis on the quality of the results provided by these algorithms is reported. Finally, applications of the methodology to real-world data are shown.

6.1.1 Assignment and Role Stability

The cost function is thoroughly described in Chapter 3 and Chapter 5. A similar concept is provided by [68], where the authors utilizes user attributes to provide a measurement of the RBAC state complexity. In general, finding the optimal candidate role-set can be seen as a *multi-objective* optimization problem. An optimization problem is multi-objective when there are a number of objective functions that are to be minimized or maximized. For the purposes of this chapter, we thus introduce the following metric for roles:

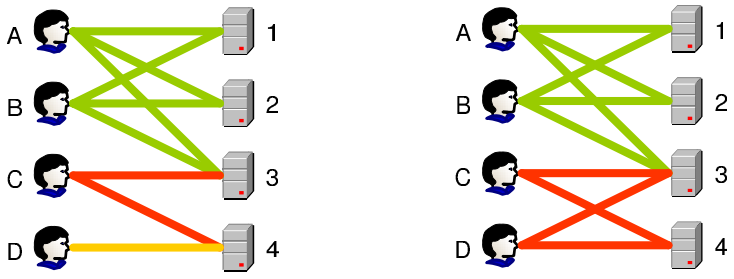
Definition 6.1 (Role Weight) Given a role $r \in ROLES$, let P_r and U_r be the sets of permissions and users associated to r, that is $P_r = \{p \in PERMS \mid \langle p, r\rangle \in PA\}$ and $U_r = \{u \in USERS \mid \langle u, r\rangle \in UA\}$. We indicate with $w : ROLES \to \mathbb{R}$ the *weight* function of roles, defined as

$$w(r) = c_u \left|U_r\right| \oplus c_p \left|P_r\right|, \tag{6.1}$$

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

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

[1] 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, as shown in Section 6.5.3.



(a) Before adding the assignment

(b) After adding the assignment

Figure 6.1 Behavior of stable and unstable assignments when new assignments are added. Candidate roles are highlighted with different colors.

Definition 6.2 (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 6.3 (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 contains 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 $\forall r \in R_{\langle u,p \rangle} : w(r) \leq t$.

If a role is composed by few user-permission relationships, its weight will be limited, and subsequently it will be unstable. Indeed, when a change of the access control configuration happens, there is the need to recalculate the optimal candidate role-set. In this case, the introduction of a new user-permission assignment could drastically change the configuration of an unstable role, according to the specific cost function considered. To better understand this concept, Figure 6.1 shows an example of assignment addition in a context where assignments that belong to roles with different weights are present. In particular, Figure 6.1(a) shows a possible system configuration. On the left side there are users $\{A, B, C, D\}$, while on the right side there are permissions $\{1, 2, 3, 4\}$. A "link" between a user and a permission indicates that the given user is granted the given permission. The picture also highlights a candidate role-set, represented by the roles:

- Role r_1: users $U_{r_1} = \{A, B\}$, permissions $P_{r_1} = \{1, 2, 3\}$;
- Role r_2: users $U_{r_2} = \{C\}$, permissions $P_{r_2} = \{3, 4\}$;
- Role r_3: users $U_{r_3} = \{D\}$, permissions $P_{r_3} = \{4\}$.

Suppose that we want user D to be granted permission 3. Figure 6.1(b) shows the resulting new configuration and proposes a new candidate role-set, represented by the following roles:

- Role r_1: users $\{A, B\}$, permissions $\{1, 2, 3\}$;
- Role r_4: users $\{C, D\}$, permissions $\{3, 4\}$.

Intuitively, by replacing roles r_2 and r_3 with the new role r_4 we get more advantages than creating a new role to manage the newly introduced assignment. Instead, role r_1 exists in both the solutions, due to the high number of users and permissions involved. Thus, it is not advantageous to modify the definition of r_1 in order to manage the new assignment. As a consequence of the previous observation, the administration of unstable assignments through roles requires more effort. Hence, the direct assignment of permissions to users could be more profitable.

In general, once an optimal set of roles has been found, the introduction of a new user or a new permission may change the system equilibrium whenever roles with limited weight exist. This translates in higher administration cost, which is something that RBAC administrators tend to avoid. Therefore, roles with a consistent weight are preferable, since they are more stable and less affected by the modifications of the existing user-permission assignments. The main idea is thus to identify and "discard" the user-permission relationships that only belong to roles with a limited weight—that is, unstable assignments. Put another way, we do not manage unstable assignments with any roles. Equivalently, we can create as many single-permission roles as the permissions involved with unstable assignments. Thus, restricting the role mining problem to the remaining user-permission assignments only. In this way, the elicited roles are *representative* and *stable*. Representative since they are used by several users or they cover several permissions. Stable because they are not greatly affected by the introduction of new users or new permissions.

6.2 Pruning Unstable Assignments

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 [34]. Then we introduce our three-step methodology.

Moreover, we prove the relation between the degree of a graph nodes and their instability. Finally, we explain how to identify unstable assignments and show some possible applications to role mining.

6.2.1 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

$$G = \langle V_1 \cup V_2,\ E \rangle = \langle USERS \cup PERMS,\ UP \rangle, \tag{6.2}$$

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 [34]. Thus, finding the optimal role-set is equivalent to identifying the biclique cover such that the corresponding roles are optimal.

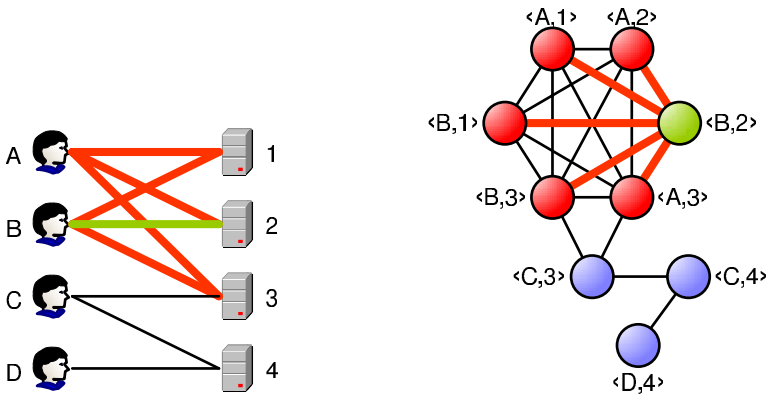
By starting from the bipartite graph G, it is possible to construct an undirected unipartite graph G' in the following way: each edge in G (i.e., an assignment of UP) becomes a vertex in G', and two vertices in G' are connected by an edge if and only if the endpoints of the corresponding edges of G induce 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:

$$B(\langle u, p \rangle) = \{ \langle u', p' \rangle \in UP \mid \langle u, p' \rangle, \langle u', p \rangle \in UP \ \wedge\ \langle u, p \rangle \neq \langle u', p' \rangle \}. \tag{6.3}$$

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 $\langle u_1, p_2 \rangle, \langle u_2, p_1 \rangle \in UP$ 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:

$$G' = \langle V', E' \rangle = \langle UP,\ \{ \langle \omega_1, \omega_2 \rangle \in UP \times UP \mid \omega_1 \in B(\omega_2) \} \rangle \tag{6.4}$$

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.



(a) Bipartite view **(b)** Unipartite view

Figure 6.2 An assignment (green) and those that induce a biclique with it (red), in both graph G and G'.

To clarify this concept, Figure 6.2 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 6.2(a)) and a vertex in the unipartite graph (Figure 6.2(b)). The figures show in red and thicker lines all the assignments that induce a biclique with $\langle B, 2\rangle$, according to Equation (6.3); 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.

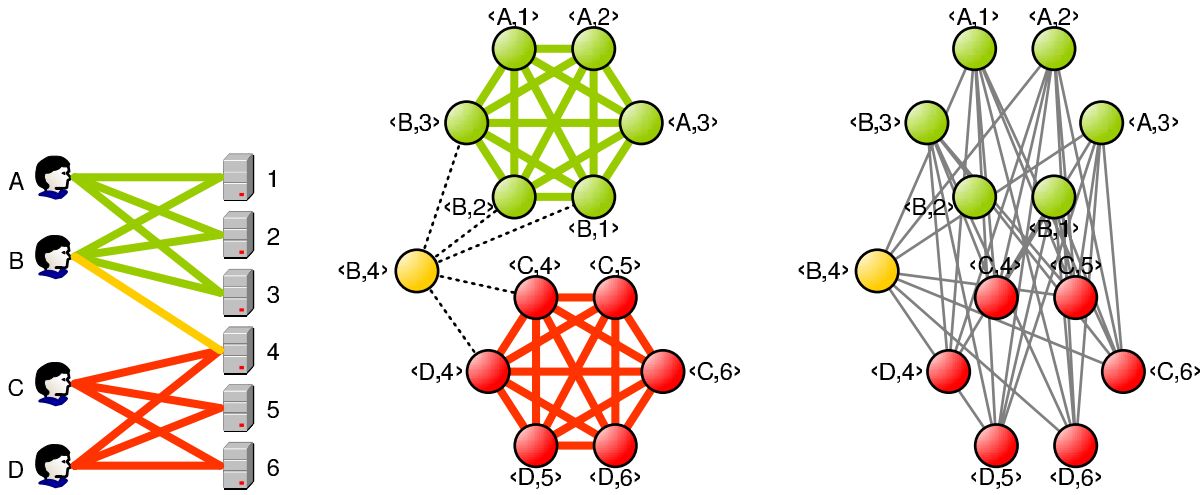
It is known that finding a clique partition of a graph is equivalent to finding a coloring of its complement [34] (see also Chapter 4). To this aim, let the graph $\overline{G}'$ made up of the same vertices of G', but edges of $\overline{G}'$ are the complement of edges of G'. Given an assignment $\omega \in UP$, we indicate with $\overline{B}(\omega)$ the assignments that do *not* induce a biclique together with ω, namely

$$\overline{B}(\omega) = (UP \setminus \{\omega\}) \setminus B(\omega). \tag{6.5}$$

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

$$\overline{G}' = \langle \overline{V}', \overline{E}' \rangle = \left\langle UP,\ \{\langle \omega_1, \omega_2 \rangle \in UP \times UP \mid \omega_1 \in \overline{B}(\omega_2)\} \right\rangle \tag{6.6}$$

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.



(a) Biclique cover in G **(b)** Clique partition in G' **(c)** Vertex coloring in $\overline{G}'$

Figure 6.3 Relationship among biclique cover, clique partition, and vertex coloring.

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 aforementioned properties are graphically depicted in Figure 6.3. In particular, Figure 6.3(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 6.3(b) represents the same information in the unipartite view in terms of clique partition. Figure 6.3(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.

6.2.2 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 6.1 *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 6.1 *If G is disconnected, the union of the biclique covers of each component of G is a biclique cover of G.*

PROOF From Lemma 6.1, 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.

6.2.3 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, 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 (6.6) the degree of $\overline{G}'$ can be expressed as:

$$\Delta(\overline{G}') = \max_{\omega \in UP} |\overline{B}(\omega)| \,. \tag{6.7}$$

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 relationship cannot be in the same role together with a high number of other user-permission relationships, it will belong to a role with few user-permission relationships, and we can estimate the maximal weight of such a

role. Hence, we can prune those user-permission relationships 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 6.2 *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 $\forall \omega \in UP,\ |B(\omega)| > d$, according to Equation (6.5) 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 6.2 *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 6.1, 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, for Theorem 6.1, we can split the problem by considering the different components distinctly, and then join the results of each component.

6.2.4 Applications to Role Mining

Having a bound for $\Delta(\overline{G}')$ makes it possible to use many known algorithms to color a graph in addition to the classical role mining algorithms mentioned in Section 6.1.1. Indeed, finding a coloring for $\overline{G}'$ corresponds to finding a candidate role-set for the given access control system configuration. The choice of which algorithm to use depends on what we are interested in. For example, a company could be interested in obtaining no more than a given number of roles, and to manage the remaining user-permission assignments through single-permission roles or directly. The following are two possible approaches to this problem.

Naïve Approach It is known that any graph with maximum degree Δ can be colored with $\Delta+1$ colors by choosing an arbitrary ordering of the vertices, and then coloring them one at time by labeling each vertex with a color not already used by any of its neighbors. In other words, we can find $\Delta(\overline{G}')+1$ roles which cover all the user-permission assignments that survived the pruning. With the pruning operation, we disregard some user-permission assignments; this is the cost to pay in order to have a limited degree for $\overline{G}'$. Due to Theorem 6.2, the neglected user-permission assignments will belong to roles with a limited weight. Thus, it is better to directly manage them or to create single-permission roles for them. Note that the value $\Delta(\overline{G}')+1$ does not represent the minimal number of roles, but it is only an upper bound for the chromatic number of $\overline{G}'$.

Randomized Approach By applying the randomized approach described in [46] to our case, it is possible to generate $\Delta(\overline{G}')/k$ roles that cover all the user-assignments which survive after the pruning, where $k = O(\log \Delta(\overline{G}'))$. This is a good result when minimizing the number of roles. The hypothesis for this result to hold are basically two: the graph must be a Δ-regular graph, with $\Delta \gg \log n$ (where n is the number of vertices that, in our case, corresponds to $|UP|$), and it must have a girth (i.e., the length of the shortest cycle contained in the graph) of at least 4. The former can be easily verified by adding some null nodes and the corresponding edges to the pruned graph $\overline{G}'$, thus obtaining a Δ-regular graph. The latter is more complex and we next discuss how to deal with this hypothesis. A triangle (i.e., a cycle of length 3) in $\overline{G}'$ means that there are three edges of G such that they do not pairwise induce a biclique. Given two edges of G, let A be the event "these two edges induce a biclique" and let $\Pr(A) = p$. If $\overline{A}$ is the complement of A, then $\Pr(\overline{A}) = 1 - p$. Given three edges, if B is the event "these three edges do not induce a biclique", then

$\Pr(B) = (\Pr(\overline{A}))^3$. Indeed, every unordered pair of the chosen triplet of edges must induce a biclique. Thus, $\Pr(B) = (1-p)^3$. This is also the probability that, having chosen three vertices in $\overline{G}'$, they compose a triangle. In other words, the probability to have a triangle in $\overline{G}'$, depends on the number of edges of the graph G. Therefore, it depends on how many user-permission assignments exist in the given access control configuration. Indeed, if the number of edges of G is close to the maximal number of edges, the probability p will be very high, and $\Pr(B)$ will be close to 0. However, suppose that $\overline{G}'$ is not completely free of triangles, but there are only a few of them. We can still use the randomized approach by removing an appropriate edge for such triangles, hence breaking them. Note that removing an edge from $\overline{G}'$ corresponds to forcing two edges of G to induce a biclique. This means that we are adding some user-permission assignments not present in the given access control configuration. The roles obtained can then be sanitized by removing those users that do not have all the permissions of the role, and managing these users in other ways, i.e. by creating single-permission roles.

6.3 Measuring Role Engineering Complexity

In this section we will discuss about the application of the *clustering coefficient* in RBAC. 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 6.2.3 not only does identify the user-permission assignments that are unstable, but it is also able to simplify the identification and selection of stable roles among all the candidate roles. 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.*

6.3.1 Clustering Coefficient

Another mathematical tool used is this chapter is the *clustering coefficient*. It was first introduced by Watts and Strogatz [96] 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:

$$\delta(v) = \left| \{ \langle u, w \rangle \in E \mid \langle v, u \rangle \in E \ \wedge \ \langle v, w \rangle \in E \} \right|. \tag{6.8}$$

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:

$$\tau(v) = \left| \{ \langle u, w \rangle \in V \times V \mid \langle v, u \rangle \in E \ \wedge \ \langle v, w \rangle \in E \} \right|. \tag{6.9}$$

The *clustering coefficient* of a graph G is defined as:

$$C(G) = \frac{1}{|V|} \sum_{v \in V} c(v), \tag{6.10}$$

where

$$c(v) = \begin{cases} \dfrac{\delta(v)}{\tau(v)}, & \tau(v) \neq 0; \\ 1, & \text{otherwise} \end{cases} \tag{6.11}$$

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—in Section 6.3 we will further explain what does "well" means in an access control scenario. 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.

6.3.2 Clustering Coefficient in G'

Let G' be the unipartite graph derived from user-permission assignments UP according to Equation (6.4). Consequently, Equation (6.9) becomes:

$$\tau(\omega) = \left| \{ \langle \omega_1, \omega_2 \rangle \in UP \times UP \mid \omega_1, \omega_2 \in B(\omega), \ \omega_1 \neq \omega_2 \} \right|, \tag{6.12}$$

namely $\tau(\omega)$ is the set of all possible pairs of elements in UP that both induce a biclique with ω. Further, Equation (6.8) becomes:

$$\delta(\omega) = \left| \{ \langle \omega_1, \omega_2 \rangle \in UP \times UP \mid \omega_1, \omega_2 \in B(\omega), \ \omega_1 \in B(\omega_2) \} \right|, \tag{6.13}$$

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 (Equation (6.10)) of the graph G' derived from an access control system configuration is thus defined as:

$$C(G') = \frac{1}{|UP|} \sum_{\omega \in UP} c(\omega), \tag{6.14}$$

where $c(\omega)$ is the local clustering coefficient of ω (see Equation (6.11)) defined as:

$$c(\omega) = \begin{cases} \dfrac{\delta(\omega)}{\tau(\omega)}, & \tau(\omega) \neq 0; \\ 1, & \text{otherwise.} \end{cases} \tag{6.15}$$

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—see below what "easy" means. Notice that, according to Equation (6.15), 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, equations (6.14) and (6.15) 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 (6.2). 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

$$\nexists U \subseteq USERS, \nexists P \subseteq PERMS \,:\, U \times P \subseteq UP,\ U_{\overline{r}} \times P_{\overline{r}} \subset U \times P, \tag{6.16}$$

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 properties of roles represented by maximal biclique are further detailed in Chapter 4, which also proposes an efficient algorithm to identify all possible roles associated to maximal biclique within G.

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

Theorem 6.3 *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 (6.3), $\forall \langle u_1, p_1\rangle, \langle u_2, p_2\rangle \in B(\omega) \cup \{\omega\} \implies \exists \langle u_1, p_2\rangle, \langle u_2, p_1\rangle \in UP$, namely both users u_1, u_2 have permissions p_1, p_2 granted. According to Equation (6.15), $c(\omega) = 1 \implies \tau(\omega) = \delta(\omega)$, 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 (6.3), $\langle u', p' \rangle \in B(\omega)$, 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 $U \subseteq \mathit{USERS}$ and $P \subseteq \mathit{PERMS}$ would exist such that $\{u, u_i\} \times \{p, p_i\} \subset U \times P \subseteq \mathit{UP}$. 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 (6.15), 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, $\forall i \in 1 \ldots n : \omega_i \in B(\omega)$, 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, $\forall i, j \in 1 \ldots |B(\omega)| : i \neq j$ we have that $\omega_i \notin B(\omega_j)$. Thus, $\delta(\omega) = 0$ and, according to Equation (6.15), $c(\omega) = 0$.

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 $\omega_1, \omega_2 \in \mathit{UP}$ 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.

6.3.3 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 6.4 *Let $\omega \in UP$ be a user-permission assignment such that $|B(\omega)| > 1$. Then, the following holds:*

$$c(\omega) \leq \frac{\underset{\omega' \in B(\omega)}{\text{avg}} \; |B(\omega')| - 1}{|B(\omega)| - 1}. \tag{6.17}$$

PROOF According to its definition, the local clustering coefficient of a vertex in G' is the ratio between its triangles (Equation (6.13)) and its triples (Equation (6.12)). 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:

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

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 (6.17) 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 (6.17) 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 [61]. The problem is even worse for roles which cover many user-permission assignments, since they are more vulnerable to noise [65].

In Section 6.5 we will show through experiments on real data that the clustering coefficient increases when pruning unstable assignments.

6.4 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 in Section 6.2. 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.

6.4.1 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

6.1 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
```

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 holds:

Lemma 6.3 *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 (6.3), $\langle u', p' \rangle \in B(\omega)$ 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 6.3 is used to define the procedure NEIGHBORS in Algorithm 6.1. 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 6.5 shows a practical application of this methodology in a real case.

6.4.2 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 6.1, which compute in $O(k|UP|)$ an *approximated* neighbor-counter

6.2 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
```

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 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 6.2 describes RandomizedCountNeighbors as an alternative approach for the procedure CountNeighbors of Algorithm 6.1. 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 6.5 *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 6.2 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 [51] 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

$$\Pr\left(\left|\sum_{i=1}^{k} X_i - \mathbb{E}\left[\sum_{i=1}^{k} X_i\right]\right| \geq t\right) \leq 2\exp\left(-\frac{2t^2}{k}\right), \tag{6.18}$$

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 (6.18) can be rewritten as

$$\Pr\left(\left|\frac{1}{k}\sum_{i=1}^{k} X_i - \mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} X_i\right]\right| \geq \varepsilon\right) \leq 2\exp\left(-2k\varepsilon^2\right), \tag{6.19}$$

where $\varepsilon = t/k$. Notice that the value $\frac{1}{k}\sum_{i=1}^{k} X_i$ is exactly the output of Algorithm 6.2. 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:

$$\mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} X_i\right] = \frac{1}{k}\sum_{i=1}^{k}\mathbb{E}[X_i]. \tag{6.20}$$

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 6.2, 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 6.5:

$$k > -\frac{1}{2\varepsilon^2}\ln\left(\frac{1-p}{2}\right). \tag{6.21}$$

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

6.4.3 Threshold Tuning

Notice that in Definition 6.3 we introduced a "threshold" to define the stability concept. However, both algorithms 6.1 and 6.2 do not require such a threshold to be defined beforehand. Put another way, changing the threshold t does not require a complete re-computation of the number of neighbors for each assignment. 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. Moreover, notice that automatically discarding all the assignments not covered by stable roles might not always be appropriate. Rather, it would be advisable to submit these results to a checker. In our case, we should not forget that security still remains the main objective. Hence, the system administrator should carefully check resulting stable roles.

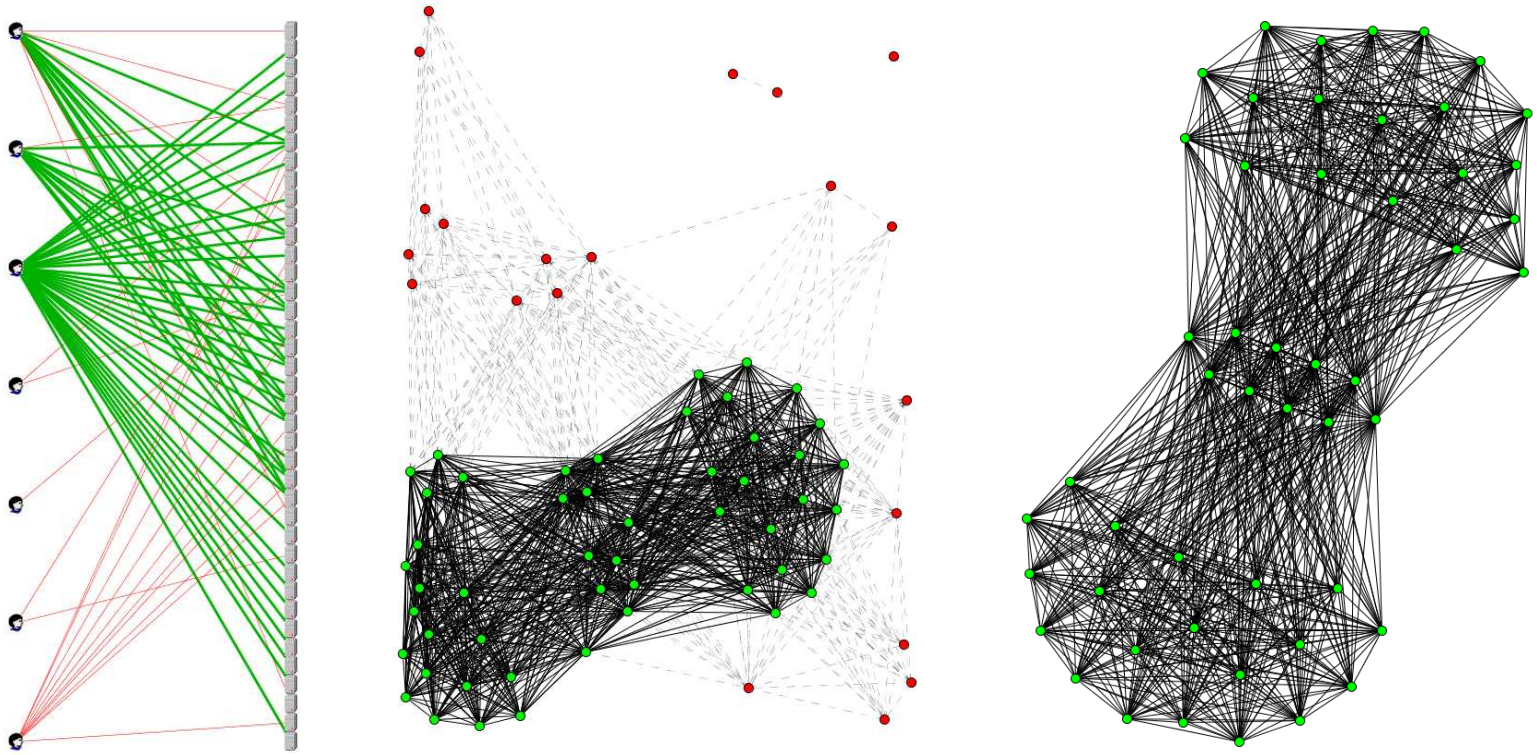
Finally, notice that checking all possible stable roles could be an unfeasible task. By having a list of roles sorted by their stability values, a system administrator can only focus on the most relevant ones, evaluating them in the reverse order. In this case, a reasonable metric for the objective achievement could be the number of clusters generated by a role mining algorithm. This approach is similar to that of Section 6.7.1, where similar considerations are made for missing values.

6.5 Pruning Examples

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 6.4.

6.5.1 A Real Case

Figure 6.4 shows an example of our strategy when applied to a real dataset. Figure 6.4(a) represents the bipartite graph G built from the access control



(a) Bipartite representation of the analyzed OU

(b) Relative unipartite graph G'

(c) Pruned unipartite graph

Figure 6.4 Our model applied to a real Organizational Unit

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 6.4(b) depicts the unipartite graph G', built according to Equation (6.4). 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 6.4(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.

6.5.2 Effects of the Pruning on the Mining Complexity

Theorem 6.4 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 coefficient grows.

Figure 6.5 graphically shows this behavior. The dataset analyzed is the same that has been used in Section 6.5.1. 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 6.5, 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

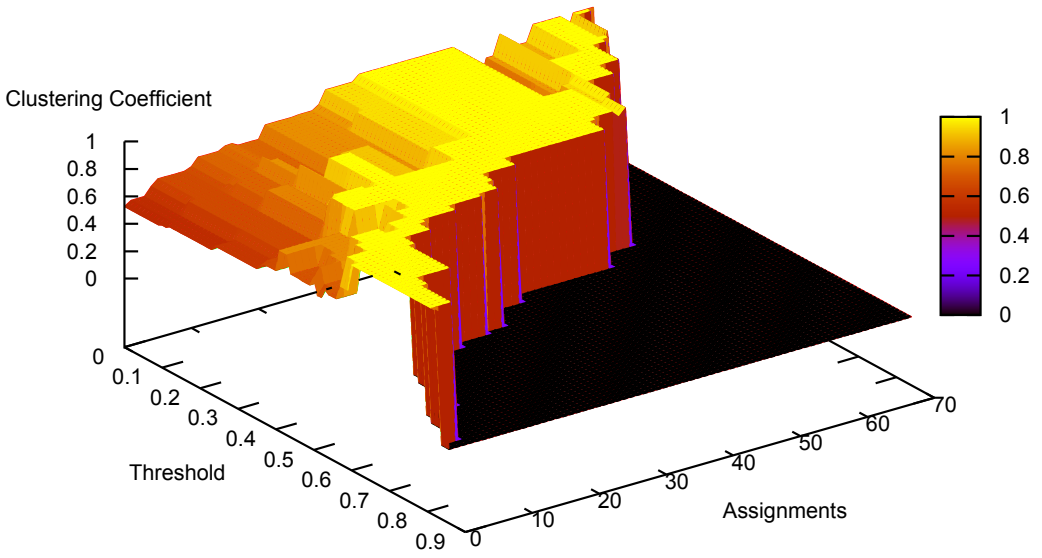


Figure 6.5 Pruning effect on local clustering coefficient

10 and 50 have a clustering coefficient equal to 1. Note that, according to Theorem 6.3 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.

6.5.3 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 [33]. One possible way to do this is combining different functions in a *weighted sum*, with the following general formulation:

$$\sum_{f_i \in F} \alpha_i f_i. \tag{6.22}$$

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:

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

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.

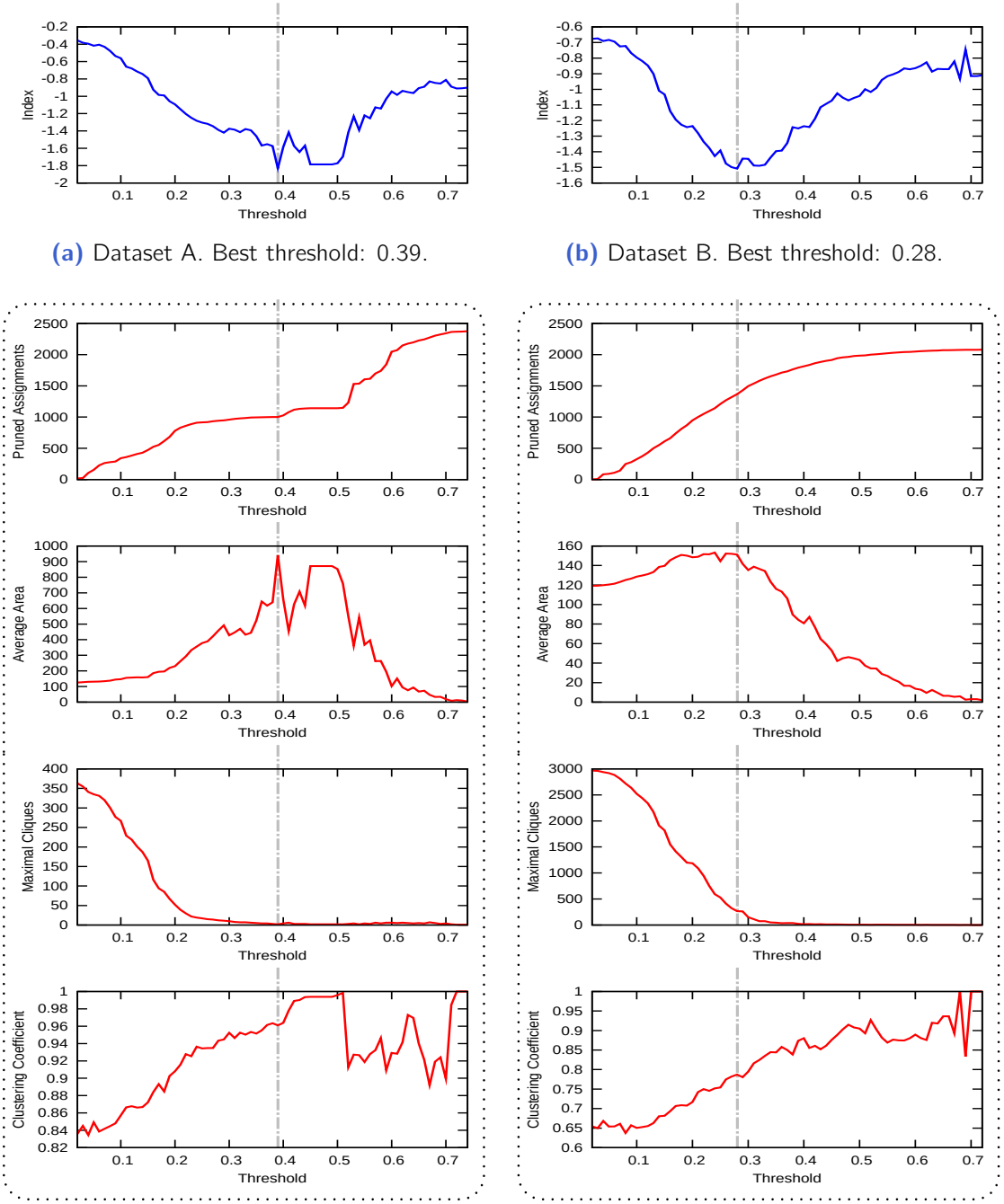
In Figure 6.6, we report two examples of the threshold tuning applied to two real datasets at our disposal. The two analyzed cases concern two organization units of a large company. They are comparable with respect to their size: the first one counts 54 users and 285 permissions, with a total of 2,379 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 6.6(a) and 6.6(b) represent the aggregated functions for these two organization unit. Figures 6.6(c) and 6.6(d) 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 6.6(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 6.6(b)).



(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 6.6 Finding the best threshold

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.

6.6 Missing Values

When dealing with binary matrices (see Chapter 2), a fundamental problem that analysts need to deal with is incomplete data, namely portions of data that are unavailable or unobserved. We refer to this data as *missing values*. Missing values arise in many practical situations. For example, in almost all medical studies important data may be missing for some subjects for a variety of reasons: from malfunctioning hardware components, to indecision due to the rounding of some measurement values, from mistakes made by the medical personnel, to refusal or inability of the patients to provide information. But, when a complete dataset is required, as is the case for most data mining and clustering tools, data analysts typically have three options before performing the analysis: to discard the rows that contain missing data, to replace missing data values with some constant, or to estimate values of missing data entries [91]. The inappropriate handling of missing values can pose serious problems. It may introduce bias on the final results, and sometimes it can also affect the generalization of the research results. For instance, when missing values are present, classical data mining algorithms discover multiple small fragments of the real cluster, but miss the cluster itself [62]. The problem is even worse for clusters which cover many rows and columns, since they are more vulnerable to this kind of "noise" [65].

In general, missing values in a binary matrix can be classified in two categories: *flagged* and *non-flagged*. In the first case, missing values are explicitly "highlighted" with a special value (e.g., with a '$*$'). In the second case, missing values are not explicitly flagged, but they are embedded in the 0's data—

namely each 0 is potentially a missing value. An example of flagged missing values comes from biology. In DNA micro-array experiments, we would like to translate hybridization intensity values into binary values where 1 indicates hybridization and 0 indicates the opposite. Unfortunately, given the intensity values provided by the scanned array, it is not always easy to determine which clones hybridized and which did not, so in the final data set we will not only have 1's and 0's but also some '$*$'. As an example of non-flagged missing values, consider the presence/absence data from paleontology or ecology. Rows represent sites, columns represent species, and 1 indicates that the corresponding species have been found in that site. In this case, we often have the situation where the 1's are reasonably certain, but the 0's can be missing values.

One solution to the missing value 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 missing value could either require a high cost or even be impossible. In turn, it is impossible to avoid missing values when they are generated by the uncertainty associated to the collected data—this happens, for example, in DNA fingerprints [38]. Yet, replacing the missing items with "plausible" values always gives better results than ignoring missing data points. One possibility is to *impute* them by analyzing uncovered structures that reveal the nature of the relationship among the rows and the columns of the binary dataset. This approach is rooted on the consideration that in a 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 physical evidences have not been found. Therefore, looking at the data makes it possible to uncover their embedded relationships and leverage them to impute missing values. However, caution should be taken: a missing value imputed in this way is *not* real data. It is only an estimation, and it could not reflect the real value. A conservative approach consists in suggesting which missing values should be analyzed first in order to improve the subsequent mining analysis. Indeed, checking all possible missing values could be an unfeasible task. By having a sorted list of the missing row-column relations based on some "relevance" index, we can focus on the most relevant missing values, thus repeating the experiment only for those values if necessary.

An approach that is often used to impute missing values is the *k-nearest neighbors* (KNN) [90]: 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

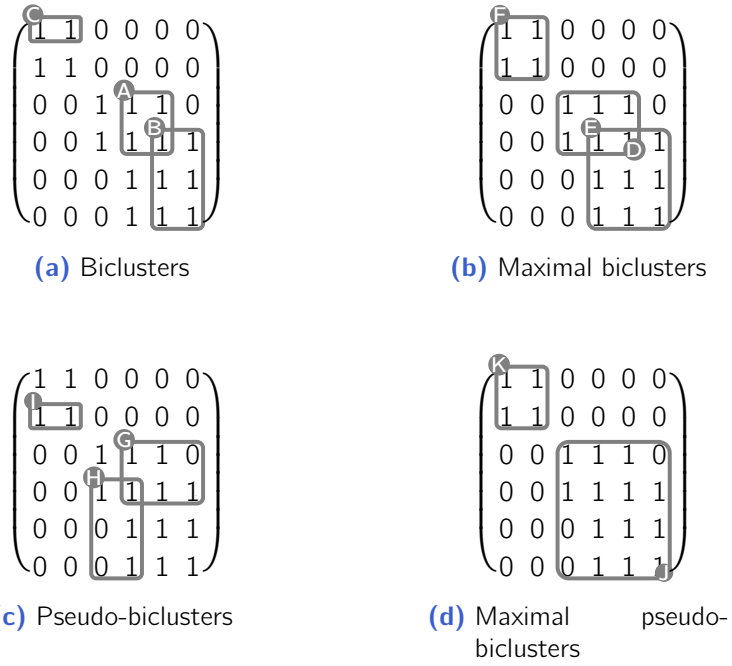
some similarity metric. In turn, the missing value is replaced with the average value for the cells on the column i within the k-nearest rows. One of the critical issues using the KNN is the choice of the parameter k. On one hand, if parameter k has a high value, rows that are significantly different 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* [55, 71, 90].

In this chapter we address the challenge posed by the imputation of flagged and non-flagged missing values. 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.

A thorough analysis of the proposed algorithm is performed. We tested our algorithm on both simulated and real data, and compared its performance to KNN. Results shows that our proposal overcome the KNN approach, especially in the difficult cases when the rate of missing values is high, and binary matrices are sparse.

6.6.1 Formal Description

Let M be a $n \times m$ binary matrix. We denote with $[n] = \{i \in \mathbb{N} \mid 1 \leq i \leq n\}$ the indices of the rows of M, and with $[m] = \{j \in \mathbb{N} \mid 1 \leq j \leq m\}$ the column indices. Moreover, we denote the i^{th} row of M with m_{i*}, the j^{th} column of



(a) Biclusters **(b)** Maximal biclusters

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

Figure 6.7 Some examples of data patterns

M with m_{*j}, and the element that is intersection of the i^{th} row with the j^{th} column with m_{ij}, and we write $m_{ij} \in M$. The *cardinality* of M is the number of elements equal to 1, and it is indicated by $|M| = \left|\{m_{ij} \in M \mid m_{ij} = 1\}\right|$. A binary matrix with flagged missing values is a matrix where some elements are set to '$*$'. Our target is to impute missing data by leveraging identifiable patterns within available data.

Definition 6.4 (Bicluster) Given a 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 6.5 (Maximal Bicluster) Let $B = \langle R, C\rangle$ be a bicluster in the 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 subsets of rows (or columns) that are filled by 1's for a given subset of columns (or

rows). 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* [101]. Applications include the discovery of association rules, strong rules, correlations, sequential rules, episodes, multidimensional patterns, and many other important discovery tasks [50]. Thus, a maximal bicluster definitely represents an interesting pattern to identify among the available data.

Figure 6.7(a) shows an example of binary matrix and also highlight 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 6.7(b) represent a maximal bicluster; indeed, '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 6.4 and 6.5:

Definition 6.6 (Pseudo-Bicluster) Given a 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 6.7 (Maximal Pseudo-Bicluster) Let $B = \langle R, C\rangle$ be a pseudo-bicluster in the 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 6.7(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 6.7(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 6.4 *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 6.6. ■

Definition 6.8 (Maximal Pseudo-Bicluster Generator) An element $m_{ij} \in M$ is referred to as a *generator* of the maximal pseudo-bicluster $B = \langle R, C \rangle$ if $R = \{\ell \in [n] \mid m_{\ell j} = 1\}$ and $C = \{k \in [m] \mid m_{ik} = 1\}$. If $m_{ij} \in M$ is a generator of B, we will also say that B is the maximal pseudo-bicluster *generated by* m_{ij}, and we indicate it as $B_{m_{ij}}$.

The key idea behind the generator concept is that, given a cell $m_{ij} \in M$, we can easily identify the bicluster generated from it by just selecting all the 1's in the same row and the same column. Further, the generators of a maximal pseudo-bicluster B all belong to rows/columns that have 1's in the same columns/rows, as is shown in the following theorem:

Theorem 6.6 *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. ■

Note that each cell that intersects a row and a column both filled by 1's is definitely a generator. Further, every pseudo cluster has at least one generator, since a pseudo-bicluster has at least one row and one column filled by 1's by definition. For instance, in Figure 6.7(d) the maximal pseudo-bicluster denoted by 'J' has two columns (the fourth and the fifth) and one row (the fourth) completely filled by 1's. Thus, cells m_{44} and m_{45} are generators of 'J'. Note that an element m_{ij} *generates exactly one* maximal pseudo-bicluster, but a maximal pseudo-bicluster can be generated by several elements $m_{ij} \in M$. In particular:

Lemma 6.5 *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 introduced the maximal pseudo-bicluster concept is that it can be used to impute missing data, for both flagged and non-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 non-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. Next section details our approach.

6.6.2 Tools

We now introduce a metric to measure the *relevance* of a maximal pseudo-bicluster as a pattern for imputing missing values. Our analysis is mainly based upon two considerations. First, the more a maximal pseudo-bicluster B is *close to being a maximal bicluster*—namely, it is "almost" filled by 1's—, the more the pattern represented by B correctly describes the available data. The second consideration is that the more a maximal pseudo-bicluster B has a *large area*—that is, a large number of involved cells—, the more it should

be considered a significant pattern, and subsequently it should have a higher relevance. In more details:

1. We prefer maximal pseudo-biclusters that are close to being maximal biclusters because, in this way, the rows and the columns of the matrix M that are involved in the maximal pseudo-biclusters are likely to be more similar.

2. We prefer maximal pseudo-biclusters with a large area because they identify patterns that involve many rows and many columns. Indeed, having a large number of rows and columns usually makes it easier to detect missing values when compared to the case when the missing values are present in a smaller pattern.

Both considerations represent a possible way to select good "candidate" patterns to be used to infer missing values. The following index captures both points described above:

Definition 6.9 (Relevance of a Maximal Pseudo-Bicluster) Let *MPBS* be the set of all maximal pseudo-bicluster existing in M, and $B \in MPBS$. The *relevance* of B is defined as:

$$\varrho(B) = \Big|\{m_{ij} \in M \mid m_{ij} \text{ is a generator of } B\}\Big| = \sum_{m_{ij} \in B} \gamma(m_{ij}, B),$$

where

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

The index $\varrho(B)$ counts the number of generators of the maximal pseudo-bicluster B. Informally, this index implicitly considers both the number of involved cells and the closeness of B to be a maximal bicluster. As for the area, the generators m_{ij} of $B = \langle R, C\rangle$ are definitely elements of B. Thus, if the area $|R| \times |C|$ is small, the value of $\varrho(B)$ cannot be high. As for closeness, notice that generators are the intersection of rows and columns filled by 1's—see Theorem 6.6 in the previous section. Hence, the more rows and columns belonging to B are similar, the higher $\varrho(B)$ is. In particular:

Theorem 6.7 *If a maximal pseudo-biclusters $B = \langle R, C\rangle$ is a maximal bicluster, then $\varrho(B) = |R| \times |C|$*

PROOF Since B is a bicluster, the submatrix of M induced by $R \subseteq [n]$ and $C \subseteq [m]$ is a 1's matrix. Moreover, since B is a maximal bicluster, then $\nexists$ a bicluster $B' = \langle R', C'\rangle : R \times C \subset R' \times C'$. Thus, for Theorem 6.6 each element $m_{ij} \in B$ is a generator of B. Since there are $|R| \times |C|$ elements in B, then $\varrho(B) = |R| \times |C|$. ∎

As stated before, the value of $\varrho(B)$ is related to the area of the maximal pseudo-biclusters $B = \langle R, C \rangle$ and its closeness to being a maximal bicluster. According to Theorem 6.7, when B is a maximal bicluster, $\varrho(B)$ is exactly equal to its area. Otherwise, $\varrho(B) < |R| \times |C|$. Indeed, some elements of B are set to 0 or '$*$'—otherwise, B would be a maximal bicluster. This means that there are rows and columns belonging to B that are not equal to each other, and thus, because of Theorem 6.6, not all the elements of B will be its generators. Subsequently, $\varrho(B) < |R| \times |C|$.

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 6.10 (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 indices $\varrho(B)$ 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 (non-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 non-flagged missing

values, we will evaluate the relevance of all the elements $m_{ij} \in M$ such that $m_{ij} = 0$.

6.7 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 6.9 and 6.10 suggest a way to practically compute the relevance values. By simply combining the two indices, the following holds:

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

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:

$$\sigma(m_{ij}) = \sum_{m_{\ell k} \in M : m_{\ell k} = 1} \delta(m_{ij}, B_{m_{\ell k}}) \tag{6.23}$$

where

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

We used Equation (6.23) to define an algorithm referred to as ABBA (*Adaptive Bicluster-Based Approach*), that is described in Algorithm 6.3. 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.

6.3 The ABƎA algorithm

```
1: procedure EvaluateMissing(M)
2:     for all m_ℓk ∈ M s.t. m_ℓk = 1 do
3:         for all m_ij ∈ B_mℓk 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
```

Algorithm 1 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 non-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 6.3 is guaranteed by Equation (6.23). 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 non-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 $O(n^2m)$ [90].

6.7.1 Threshold Tuning

As stated in Section 2.4, the *multiple imputation* method [80] 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 6.3. 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 (see Chapter 3). 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* (see Chapter 5) 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.

6.8 Missing Values in Real Data

To test the quality of the results produced by ABϬA, we now introduce a measure based on the Jaccard's coefficient [36] that has already been used in [38] 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"—namely, the number of 1's in both matrices M and R in the same cell positions—, 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 $n_{11}/(n_{11}+n_{10}+n_{01})$. It represents the proportion of the correctly identified mates to the sum of the correctly identified mates plus the total number of disagreements. Hence, the Jaccard's coefficient should score one when all the missing values are correctly identified. Conversely, the closer this index is to zero, the less the two matrices can be considered similar.

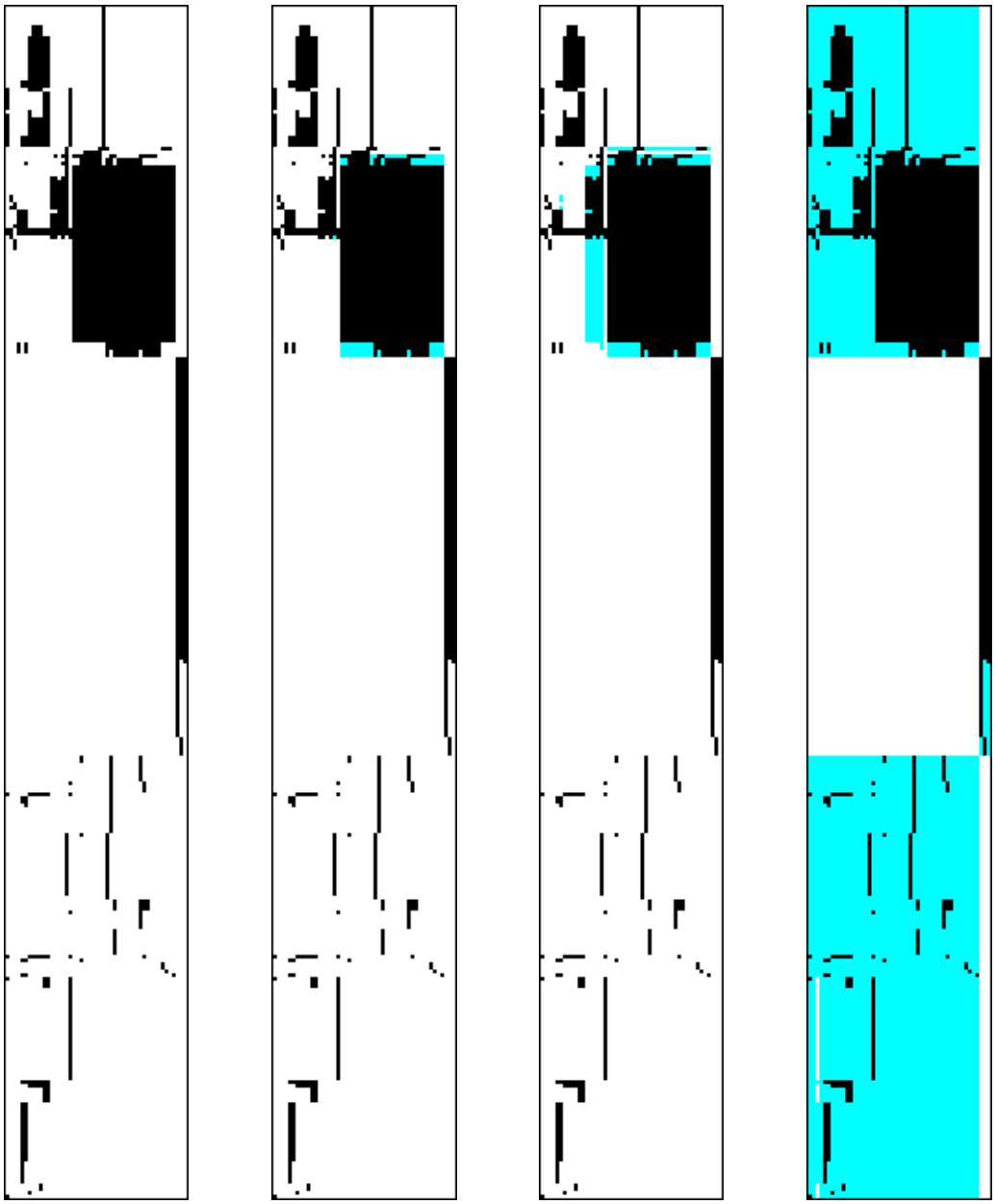
6.8.1 Testing on Synthetic Data

In the following simulations we first generate a matrix M then, uniformly at random, we introduce in M a fraction of missing values, generating a 0-1-'$*$' matrix M'. In turn, M' is given as an 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).

To implement our experiments, we generated sample matrices composed by 600 rows and 100 columns. Each matrix has been generated with the next described procedure. 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 $\texttt{MaxColumns} \times x^y$, where x is a random number uniformly chosen between 0 and 1, while y, `MaxRows` and `MaxColumns` 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.

Figure 6.9 reports the value of the Jaccard's coefficient, considering the threshold t and the fraction of missing values introduced in M. To plot each point of the surface, 20 matrices have been generated according to the described method, with $y = 2$, $\texttt{MaxRows} = 60$ and $\texttt{MaxColumns} = 10$. Then, the average value of the Jaccard's coefficients has been calculated. It can be seen that the threshold plays a fundamental role. Indeed, using a threshold $t = 0$ means switching all the missing values that are involved in some patterns to 1, though it may be an unimportant one. Conversely, when the threshold is close to 1, a missing value is switched to 1 only if it is involved in a relevant pattern. As exposed in Section 6.7.1, depending on the target function, it is possible to select the best threshold t to use. As for the Jaccard's coefficient, by analyzing Figure 6.9 it is possible to note that the best results are obtained using a threshold between 0.1 and 0.2. Additionally, Figure 6.9 shows that our algorithm is able to reach a Jaccard's coefficient equal to 0.8 even if the missing values rate is equal to 0.4.

In order to show the advantages provided by our approach, we compared our algorithm with KNN. Table 6.1 summarizes the results for some sample matrices. To be fair, we reported several results for KNN obtained by using different k. Note that selecting the best k corresponds to an additional operation, and that it is not needed in ABϐA. Each row of the table corresponds to a sample matrix generated as above using the indicated parameters, and $y = 5$.



(a) Original **(b)** $t = 0.73$ **(c)** $t = 0.36$ **(d)** $t = 0$

Figure 6.8 An application of ABƎA to non-flagged missing values

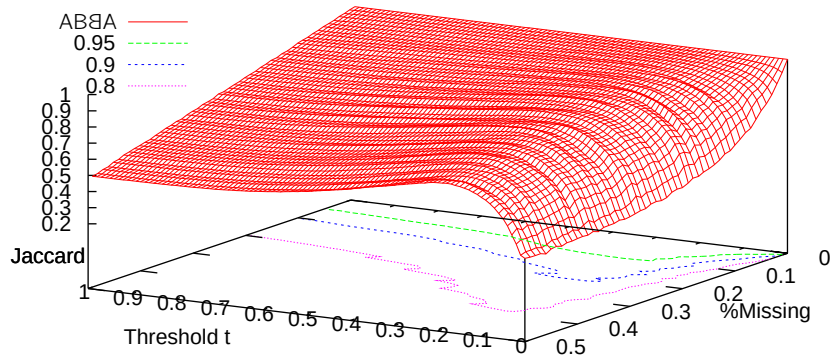


Figure 6.9 Comparing the rebuilt matrix with the original one

To have more reliable results, the results correspond to the average of 20 simulations, where the threshold t that assures 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 `MaxColumns`), mainly when the number of missing values is high.

6.8.2 Testing on Real Data

As an application of our algorithm to discover non-flagged missing data, we have chosen to analyze a real dataset coming from an access control application. In particular, the dataset is composed by a binary matrix where rows represent users, and columns are permissions. A user-permission pair is set to 1 if the user must have the permission granted, 0 otherwise. We evaluated the dataset using our algorithm with the target to discover data that are set to 0, but that could be set to 1. This operation, if carefully devised, can drastically reduce the number of biclusters (referred to as "roles" in the access control terminology) that are needed to manage an access control system. The analyzed data involves a set of 323 users and 49 permissions related to a particular organization unit of a private company, counting a total of 2342 permissions granted—that correspond to almost 15% of cells set to 1. Figure 6.8 graphically shows the achieved results. Figure 6.8(a) is the original matrix, where black pixels corresponds to 1's (i.e., a permission granted to a user), while white pixels to 0's. Figure 6.8(b) shows the imputed matrix, where some of the initial 0-values have been switched to 1 according to a threshold $t = 0.73$ and highlighted in light blue. Figures 6.8(c) and 6.8(d) differs from Figure 6.8(b)

Table 6.1 A comparison between KNN and ABBA

MaxRows	Max-Columns	Missing rate	KNN(4)	KNN(16)	KNN(32)	KNN(64)	ABBA
30	6	0.1	0.885	0.895	0.896	0.896	0.914
		0.2	0.786	0.791	0.794	0.791	0.836
		0.3	0.702	0.715	0.715	0.716	0.811
		0.4	0.611	0.618	0.618	0.618	0.724
		0.5	0.500	0.507	0.508	0.507	0.679
60	8	0.1	0.911	0.921	0.925	0.925	0.949
		0.2	0.821	0.836	0.841	0.842	0.885
		0.3	0.703	0.717	0.720	0.727	0.854
		0.4	0.604	0.616	0.616	0.616	0.750
		0.5	0.497	0.505	0.509	0.513	0.719
90	12	0.1	0.942	0.944	0.944	0.944	0.949
		0.2	0.848	0.862	0.867	0.865	0.903
		0.3	0.747	0.767	0.771	0.777	0.861
		0.4	0.620	0.638	0.644	0.653	0.811
		0.5	0.500	0.514	0.514	0.523	0.708
120	16	0.1	0.938	0.944	0.945	0.944	0.949
		0.2	0.827	0.833	0.834	0.831	0.887
		0.3	0.772	0.793	0.802	0.803	0.850
		0.4	0.614	0.635	0.637	0.639	0.756
		0.5	0.549	0.577	0.584	0.586	0.772
150	20	0.1	0.957	0.958	0.959	0.959	0.959
		0.2	0.914	0.925	0.929	0.930	0.930
		0.3	0.817	0.838	0.842	0.844	0.883
		0.4	0.671	0.698	0.706	0.709	0.783
		0.5	0.587	0.622	0.633	0.637	0.751
180	24	0.1	0.958	0.960	0.960	0.959	0.952
		0.2	0.913	0.922	0.923	0.923	0.920
		0.3	0.795	0.808	0.807	0.808	0.858
		0.4	0.711	0.732	0.738	0.738	0.812
		0.5	0.611	0.644	0.657	0.659	0.771

only for the used threshold, that is equal to 0.36 in the first case and 0 in the other one.

It is possible to seen that, the more the threshold t is decreased, the larger the number of user-permission assignments switched to 1 are. Once we submitted these results to the system administrators, they recognized that the user-permissions assignments found using the threshold $t = 0.2$ are actually permissions that could be granted to the corresponding users. This means

that only 96 out of 13,485 possible cells that were 0's have been switched to 1. The new access control scheme, that corresponds to the matrix rebuilt according to this threshold, contains 102 maximal biclusters, with respect to the 127 maximal biclusters that can be identified within the original access control scheme. This means that, for each user-permission assignment, the ambiguity of selecting the role (i.e., the bicluster) needed to manage them is lower.

6.9 Final Remarks

In this chapter we proposed a three steps methodology, rooted on sound graph theory, to reduce the role mining complexity in RBAC systems. The methodology is implemented by two different algorithms: a deterministic one and a probabilistic one. The latter trades off computation time with a (slight, yet tunable) decrease in the quality of the provided results. To show the viability of the proposal, the methodology is applied to a concrete case. Extensive experiments on real data set do confirm its viability, as well as the quality of the results achieved by the related algorithms.

In this chapter we also proposed a solution to the missing values imputation problem in binary matrices. In particular, we proposed a novel algorithm ABBA (Adaptive Bicluster-Based Approach) that leverages the underlying implicit relationships among data to address the issue. A thorough formal framework has been provided to justify the rationales behind our algorithm. Further, ABBA enjoys some relevant features when compared to its direct competitor (KNN): its tuning is much easier and the computational complexity is reduced for a wide range of parameters. Moreover, experimental results on simulated data confirmed that the output of ABBA is always better than KNN, mainly in the difficult case when the rate of missing values is high. Finally, we applied our algorithm to a real problem in order to impute non-flagged missing values, also observing that our approach can improve the results achieved by a subsequent data analysis task.

7 The Risk of Unmanageable Roles

The last contribution of this thesis is represented by the analysis of the *risk* derived from granting access to resources, and how RBAC allows to effectively manage such a risk. In particular, we point out that existing role mining techniques are not able to elicit roles with an associated clear business meaning. Hence, it is difficult to mitigate risk, to simplify business governance, and to ensure compliance throughout the enterprise. To elicit meaningful roles, we recall the methodology described in Chapter 5 where data to analyze are decomposed into smaller subsets according to the provided business information. We show how this leads to a decrease in the likelihood of making errors in role management, and consequently reduces the risk of role misuse. Additionally, we extend the concepts proposed in Chapter 6 by introducing an approach to highlight users and permissions that markedly deviate from others, and that might consequently be prone to error when roles are operating. By focusing on such users and permissions during the role definition process, it is possible to mitigate the risk of unauthorized accesses and role misuse. This chapter summarizes the contribution previously published in [9, 11].

7.1 Reducing the Risk of Unmanageable Roles

To address all the aforementioned issues, this section proposes a methodology that helps role engineers leverage business information during the role mining process. In particular, we propose to divide the access data to analyze

into smaller subsets that are homogeneous according to some business data, instead of performing a single bottom-up analysis on the entire organization. This eases the attribution of business meaning to automatically elicited roles and reduces the problem complexity, thus allowing for better enforcement of security policies and reducing the risk related to illegal accesses. In order to select the best business information that improves the subsequent role mining process, as well as to establish how deeply the data must be partitioned, two indices, referred to as *minability* and *similarity*, are identified. Minability and similarity are both rooted on sound mathematical theory. These indices are used to measure the expected complexity of analyzing the outcome of the bottom-up approaches. Leveraging these indices allows for the identification of business information that best fits with the access control data, namely the information that induces a decomposition which increases the business meaning of the roles elicited in the role mining phase and, at the same time, simplifies the analysis. This leads to a decrease in the likelihood of making errors in role management, and consequently reduces the risk of role misuse. The chapter also introduces two fast probabilistic algorithms to efficiently compute such indices, making them suitable also for big organization with hundreds of thousands of users and permissions. The quality of the indices is also formally assured. Several examples illustrate the practical implications of the proposed methodology and related tools, which have also been applied on real enterprise data. Results support the quality and viability of the proposal.

7.2 Preliminaries

7.2.1 Jaccard and Clustering Coefficients

In this chapter we extensively use two mathematical tools. The first one is represented by the *Jaccard coefficient* [52] that is a measure of the similarity between two sets. Given two sets S_1, S_2, the coefficient is defined as the size of the intersection divided by the size of their union:

$$J_{S_1 S_2} = |S_1 \cap S_2| \,/\, |S_1 \cup S_2| \,. \tag{7.1}$$

The Jaccard coefficient is widely used in statistic. However, to our knowledge, the only application to RBAC is given by Vaidya et al. [93], that offers a method to consider previously defined roles during the role mining process in order to minimize the "perturbation" introduced by new candidate roles. In Section 7.3.1 we will use the Jaccard index to measure the similarities among users of an access control configuration.

The other mathematical tool is the *clustering coefficient*, described in Section 6.3.1. In this section, however, we use an alternative form: in particular, please note that Equation (6.10) at page 103 can also be written as:

$$C(G) = \frac{1}{|V|} \sum_{v \in V} \sum_{\pi \in \Pi_v} \frac{X(\pi)}{\tau(v)} + \frac{|\{v \in V \mid \tau(v) = 0\}|}{|V|},$$

where $X : \Pi_v \to \{0, 1\}$ is such that $X(\pi)$ is equal to 1 if there is an edge between the outer nodes of the triple π, and 0 otherwise. The first addendum is the local clustering coefficient of the subset of vertices that have $\tau(v) \neq 0$, while the second one corresponds to the local clustering coefficient of the set of vertices such that $\tau(v) = 0$. Leveraging this definition of the clustering coefficient, $C(G)$ can be computed by considering each triple of the graph G, and then checking if it is a triangle or not. Further details follows in Section 7.3.2.

7.2.2 Risk Model

To better clarify the benefits introduced by the proposed approach, we first recall some risk management concepts. In particular, a typical risk management approach is made up of two key components: *risk analysis* (or *assessment*) and *risk control*. During risk analysis we identify potential risks and assess probabilities of negative events together with their consequences. With risk control we establish the tolerable level of risk for the organization, hence providing controls for failure prevention as well as actions to reduce the likelihood of a negative event—such an activity is usually referred to as *risk mitigation*.

Plugging the previous concepts in a RBAC environment, three essential components should be considered: users, roles, and permissions. Among them, particular attention must be taken on risk incurred by users. Indeed, the main threat in an access control scenario is to allow a user to execute an illegitimate operation over an object or a resource. A system which is only supposed to be used by authorized users must attempt to detect and exclude unauthorized ones. Accesses are therefore usually controlled by insisting on an authentication procedure to determine with some established degree of confidence the identity of the user, hence granting permissions authorized to that identity. RBAC mitigate the risk of unauthorized accesses by restricting user's permission to predefined role definitions. In a usual RBAC setting, users are assigned to roles which are then granted permissions to perform predefined tasks [42].

In this scenario, and assuming that it is not possible to by-pass the access control mechanism, the risk of illegitimate credentials is prevented by adopting a RBAC system. But, there is an important aspect that has not been

considered so far: the role lifecycle. Roles are not static, but they follow the evolution of the organization: new users may join, existing users may leave or may change their job position, applications may be replaced with new ones, etc.. Hence, an important aspect to consider when evaluating the risks related to RBAC systems is the risk introduced by roles that are difficult to manage, mainly due to an unclear understanding of their meaning. Indeed, the more a role is intelligible and well designed, the less error prone it will be. A comprehensive risk management approach should consider these aspects starting from the creation of roles, that is the role mining phase. More specifically, we focus on the following risk-related aspects of a generic RBAC system:

- *Vulnerabilities*. They correspond to roles that are not meaningful enough from the administrator's perspective, namely roles that are difficult to manage and to maintain.
- *Threats*. They are errors and wrong administration actions, unintentionally committed while managing roles during their lifecycle.
- *Risks*. They correspond to allowing users to execute operations that are not permitted, or hampering their jobs by not granting required permissions. In both cases, the consequences could raise financial loss.

To evaluate such risks, in this chapter we propose a general risk formula that involves multiple factors with different probabilities, namely:

$$Risk = \sum_{i=1}^{n} P_i \times C_i, \tag{7.2}$$

where P_i denotes the probability of each risk factor i, and C_i quantifies the consequences of these risk factors. In our model, risk factors are represented by homogeneous groups of users. Indeed, every user does not have the same degree of importance. For example, there could be users in charge of activities that are critical for the main business of the organization, while other users could be assigned to roles that have a marginal importance for the business. In general, we need to assign various degree of importance to each risk factor by taking the consequence of its execution into consideration. This process requires a thorough analysis of the organization. We assume that the impact evaluation is provided by experts. As for the probability of occurrence, we are able to propose two metrics that are suitable to evaluate the likelihood that an administration error is made when managing roles. In such a way, we evaluate the risk of an error in role management, and subsequently we are able to drive the definition of roles that mitigate this risk.

7.3 New Metrics for Role Mining

In this section we describe two indices that can help role engineers to condition role mining in order to craft roles with business meaning and to downsize the problem complexity, hence reducing risk issues as well as required role mining effort. The first index is referred to as *similarity*, and its value is proportional to the number of permissions a given set of users share. The second one is *minability*, and it measures the complexity of selecting candidate roles given a set of user-permission assignments. Both indices provide a measure of how easy it is to analyze a given set of user-permission assignments through a bottom-up approach, but using different perspectives. Indeed, roles may be classified in two categories, as described in [29, Ch. 5] and [70]:

- *Organizational or Structural Roles*, which depend on employee's position within a homogeneous group of users—for instance, an organization unit or all users that have the same job title, e.g., doctor, nurse, bank teller. Common permissions are usually assigned to these kinds of roles, and each user typically has only one organizational role.

- *Functional Roles*, which depend on the task that need to be performed in a particular position. Detailed permissions are usually assigned to this kind of roles. Functional roles are supposed to provide further access rights in addition to those being granted by organizational roles—i.e., for a given organizational role "Specialist", we can provide specialized roles such as "Cardiologist". or "Oncologist" Any number of functional roles can be assigned to a user.

Since the similarity index helps identify situations where all users share the majority of their permissions, its usage is most suitable when evaluating how easy identifying organizational roles is. Instead, the minability index indicates the level of complexity involved in identifying subsets of users that share the same permissions; thus, it is suitable to evaluate whether finding roles, both functional and organizational, is a simple task or not.

By leveraging the previous observations, it possible to use information resulting from a top-down approach in order to drive a bottom-up analysis. The key idea is to partition the data-set to analyze into smaller subsets according to some business information. For instance, a top-down analysis might identify groups of users that are homogeneous from an enterprise perspective. Then, user-permission assignments can be partitioned such that each subset contains only assignments related to users of the same group. For each subset, we calculate the minability and similarity indices, thus getting a prediction about how

complex a subsequent role mining task on the subset will be. This decomposition process can be performed for each available business information, thus generating different data partitions; by selecting the one with the highest index values, we choose the partition that most simplifies the subsequent role mining analysis. Furthermore, each subset might be iteratively partitioned in even smaller subsets until we reach a given threshold for minability and similarity. The application of role mining algorithms on each subset will produce roles with more business meaning when compared to the outcome of the same algorithms applied on the whole data-set: since subsets are identified according to some business criteria, elicited roles will likely have a business meaning and the probability that administrators select a wrong role to assign to users will decrease, hence reducing the related risk.

In the following, we formally describe the minability and similarity indices, then in Section 7.4 we will introduce a methodology to apply them in practice within the proposed risk model.

7.3.1 Similarity

In order to introduce the similarity index, we leverage the Jaccard coefficient defined in Section 7.2.1. In particular, referring to Equation (7.1), we provide the following definition:

Definition 7.1 (Similarity Between Two Users) Given two users $u_1, u_2 \in$ *USERS*, the *similarity index* between them is formally defined as:

$$s(u_1, u_2) = \frac{\left|perms(u_1) \cap perms(u_2)\right|}{\left|perms(u_1) \cup perms(u_2)\right|}. \tag{7.3}$$

The following observations about permission overlapping and inclusion are useful for the sequel:

- u_1 "contains" u_2 if $perms(u_1) \supset perms(u_2)$, namely permissions of u_1 are also possessed by u_2, but are not equal ($perms(u_1) \neq perms(u_2)$). In such a case, $s(u_1, u_2) \in (0, 1)$.
- u_1 is "equivalent" to u_2 if $perms(u_1) = perms(u_2)$, namely u_1, u_2 share the same permission set. Hence, we have that $s(u_1, u_2) = 1$.
- u_1 "overlaps" u_2 when $perms(u_1) \cap perms(u_2) \neq \emptyset$ but $perms(u_1) \not\supseteq perms(u_2)$ and $perms(u_2) \not\supseteq perms(u_1)$, namely u_1, u_2 share some permission but neither does u_1 contain u_2 nor does u_2 contain u_1. This means that $s(u_1, u_2) \in (0, 1)$.
- u_1 is "not related" to u_2 if $perms(u_1) \cap perms(u_2) = \emptyset$, namely u_1, u_2 do not share any common permissions. As a result, $s(u_1, u_2) = 0$.

Definition 7.2 (Similarity Among a Set of Users) For a given set of users *USERS*, the similarity index is the average similarity between all possible (unordered) user pairs. Formally,

$$\mathcal{S}(USERS) = \begin{cases} \dfrac{1}{\binom{|USERS|}{2}} \displaystyle\sum_{\substack{u_1,u_2 \in USERS: \\ u_1 \neq u_2}} s(u_1,u_2), & |USERS| > 1; \\ 1, & \text{otherwise.} \end{cases} \tag{7.4}$$

Notice that Equation (7.3) and Equation (7.4) can be extended to also consider other enterprise information. For instance, similarities can be evaluated over shared activities, involved organization units, etc.. We can define a similarity index for each kind of business data. In general, the most suitable similarity definition depends on specific organization needs and role engineering requirements. To ease exposition, in this chapter the term "similarity" indicates only the percentage of permissions shared among users, according to the previous definitions.

7.3.2 Minability

An access control system configuration can be represented by a bipartite graph $B = \langle USERS \cup PERMS,\ UP \rangle$, 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 this graph B univocally identifies a *candidate role-set* (see Chapter 3 and Chapter 5), namely a set of roles, and for each role a set of users and permissions, such that all the user-permission assignments belonging to *UP* can be covered by at least one role. Indeed, every biclique identifies a role, and the vertices of the biclique identify the users and the permissions assigned to this role [34] (see also Chapter 4). Starting from the bipartite graph B set up via the user-permission relations in *UP*, it is possible to construct an undirected unipartite graph G in the following way: each edge in B (i.e., a user-permission relationship 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 B induce a biclique. To ease exposition, we define the function $B\colon UP \to 2^{UP}$ that indicates all edges in *UP* which induces a biclique together with the given edge, namely:

$$B(\langle u,p \rangle) = \{ \langle u',p' \rangle \in UP \mid \langle u,p' \rangle, \langle u',p \rangle \in UP \wedge \langle u,p \rangle \neq \langle u',p' \rangle \}. \tag{7.5}$$

Note that a pair of 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 another pair

$\langle u_1, p_2\rangle, \langle u_2, p_1\rangle \in UP$ exists. 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 UP can be defined as:

$$G = \big\langle UP,\ \{\langle \omega_1, \omega_2\rangle \in UP \times UP \mid \omega_1 \in B(\omega_2)\}\big\rangle. \tag{7.6}$$

Any clique partition of G corresponds to a biclique cover of B as well as to a possible solution for the role mining problem represented by the sets *USERS*, *UA*, and *PA* (see Chapter 4).

Given the previous graph model, the minability index measures how complex it is to identify and select the roles required to manage existing user-permission assignments. To do this, we will redefine the clustering coefficient (see Section 6.3.1) to be used with bipartite graphs that represent the user-permission assignments of an organization. In particular, given a user-permission assignment $\omega \in UP$, we define the function $triples\colon UP \to 2^{UP\times UP}$ as

$$triples(\omega) = \big\{\langle \omega_1, \omega_2\rangle \in UP \times UP \mid \omega_1, \omega_2 \in B(\omega) \ \wedge\ \omega_1 \neq \omega_2\big\}, \tag{7.7}$$

namely the set of all possible pairs of elements in UP that both induce a biclique with ω. We also define the function $triangles\colon UP \to 2^{UP\times UP}$ as

$$triangles(\omega) = \big\{\langle \omega_1, \omega_2\rangle \in triples(\omega) \mid \omega_1 \in B(\omega_2)\big\}, \tag{7.8}$$

namely 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.

Definition 7.3 (Minability) The *minability index* of an access control system configuration represented by the set UP is defined as

$$\mathcal{M}(UP) = \frac{1}{|UP|}\sum_{\omega\in UP} m(\omega), \tag{7.9}$$

where

$$m(\omega) = \begin{cases} \dfrac{|triangles(\omega)|}{|triples(\omega)|}, & triples(\omega) \neq \emptyset; \\ 1, & \text{otherwise.} \end{cases} \tag{7.10}$$

The value of $m(\omega)$ is also referred to as the *local* minability index of ω, and it quantifies how close ω, together with all the edges which induce a biclique with it, are to being a biclique. Equation (7.9) can be alternatively written as

$$\mathcal{M}(UP) = \frac{1}{|UP|}\sum_{\omega\in UP}\sum_{\langle \omega_1,\omega_2\rangle\in triples(\omega)} \frac{Y(\omega,\omega_1,\omega_2)}{|triples(\omega)|} + \frac{|\{\omega \in UP \mid triples(\omega) = \emptyset\}|}{|UP|}, \tag{7.11}$$

where $Y : UP \times UP \times UP \rightarrow \{0,1\}$ returns 1 if their parameters induce a biclique, and 0 otherwise. Formally:

$$Y(\omega, \omega_1, \omega_2) = \begin{cases} 1, & \langle \omega_1, \omega_2 \rangle \in \mathit{triangles}(\omega); \\ 0, & \text{otherwise.} \end{cases}$$

With the above formulation, the minability index $\mathcal{M}(UP)$ can be computed by considering each tuple $\langle \omega, \omega_1, \omega_2 \rangle \in UP \times UP \times UP$ such that $\omega_1, \omega_2 \in B(\omega)$, and checking whether the condition $\omega_1 \in B(\omega_2)$ holds true.

The following lemma defines a mapping between the clustering coefficient, as defined in Equation (6.10), and the minability index:

Lemma 7.1 *If G is the unipartite graph constructed from UP according to Equation (7.6), then $\mathcal{M}(UP) = C(G)$, that is the minability of UP is equal to the clustering coefficient of G.*

PROOF It follows by construction of the graph G in Equation (7.6), and definitions of $\mathcal{M}(UP)$ in Equation (7.9) and $C(G)$ in Equation (6.10). ■

Lemma 7.1 will be used in Section 7.3.3 and Section 7.4.3 to offer a graph representation for the given examples. A relevant observation relates similarity with minability, and it is represented by the following lemma:

Lemma 7.2 *Given a set of users $USERS$ that have been granted permissions in $PERMS$ through the corresponding assignments UP, then $\mathcal{S}(USERS) = 1 \implies \mathcal{M}(UP) = 1$.*

PROOF When $\mathcal{S}(USERS) = 1$ and $|USERS| = 1$, there is only one user that possesses all the permissions in $PERMS$. Instead, when $\mathcal{S}(USERS) = 1$ and $|USERS| > 1$, all users share the same permission set, that is $\forall u_1, u_2 \in USERS : \mathit{perms}(u_1) = \mathit{perms}(u_2)$. In both cases, $UP = USERS \times PERMS$. According to Equation (7.7) and Equation (7.8), $\forall \omega \in UP : \mathit{triples}(\omega) = \mathit{triangles}(\omega)$. The proof immediately follows from minability definition Equation (7.9). ■

The previous lemma states that the similarity index is tighter than the minability index. Indeed, a similarity index equal to 1 requires that all users share the same set of permissions, whereas minability can be equal to 1 even if the users do not share all the same permissions. Further, notice that the inverse of Lemma 7.2 does not hold. The example depicted in Figure 7.1(a) is one possible case of $\mathcal{M}(UP) = 1$ and $\mathcal{S}(USERS) < 1$. In the following section we will offer more details on this example.

In the remainder of this section we introduce definitions and prove theorems that help to better understand the relationship between the minability index and the complexity of the role mining problem. In particular, the following definition is required to formalize all the subsequent considerations:

Definition 7.4 (Maximal Equivalent Roles (MERs)) Given a role $\overline{r} \in ROLES$, it is a *Maximal Equivalent Role* (MER) if:

$$\nexists U \subseteq USERS, \nexists P \subseteq PERMS : U \times P \subseteq UP \wedge ass\_users(\overline{r}) \times ass\_perms(\overline{r}) \subset U \times P. \quad (7.12)$$

Informally, a MER is a role that is "representative" of all possible subsets of permissions shared by a given set of users (see Chapter 4). The key observation which is made regarding a MER is that two permissions which always occur together among users should simultaneously belong to the same candidate roles. Without further business semantics of access control data, a bottom-up approach to role engineering cannot differentiate between a role made up of two permissions and two roles that contain individual permissions. 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. MERs properties are further detailed in Chapter 4, which also proposes a variant of the *Apriori* algorithm to efficiently identify all possible MERs within *UP*.

The following theorem relates the minability index to the complexity of the role mining problem in terms of number of MERs:

Theorem 7.1 *Let ROLES be the set of all possible MERs that can be derived from UP. Given a user-permission assignment $\omega \in UP$, let $MERS_\omega$ be the set of all MERs that "cover" the given user-permission assignment, that is $MERS_\omega = \{r \in ROLES \mid \omega \in ass\_users(\overline{r}) \times ass\_perms(\overline{r})\}$. Then, the followings holds:*

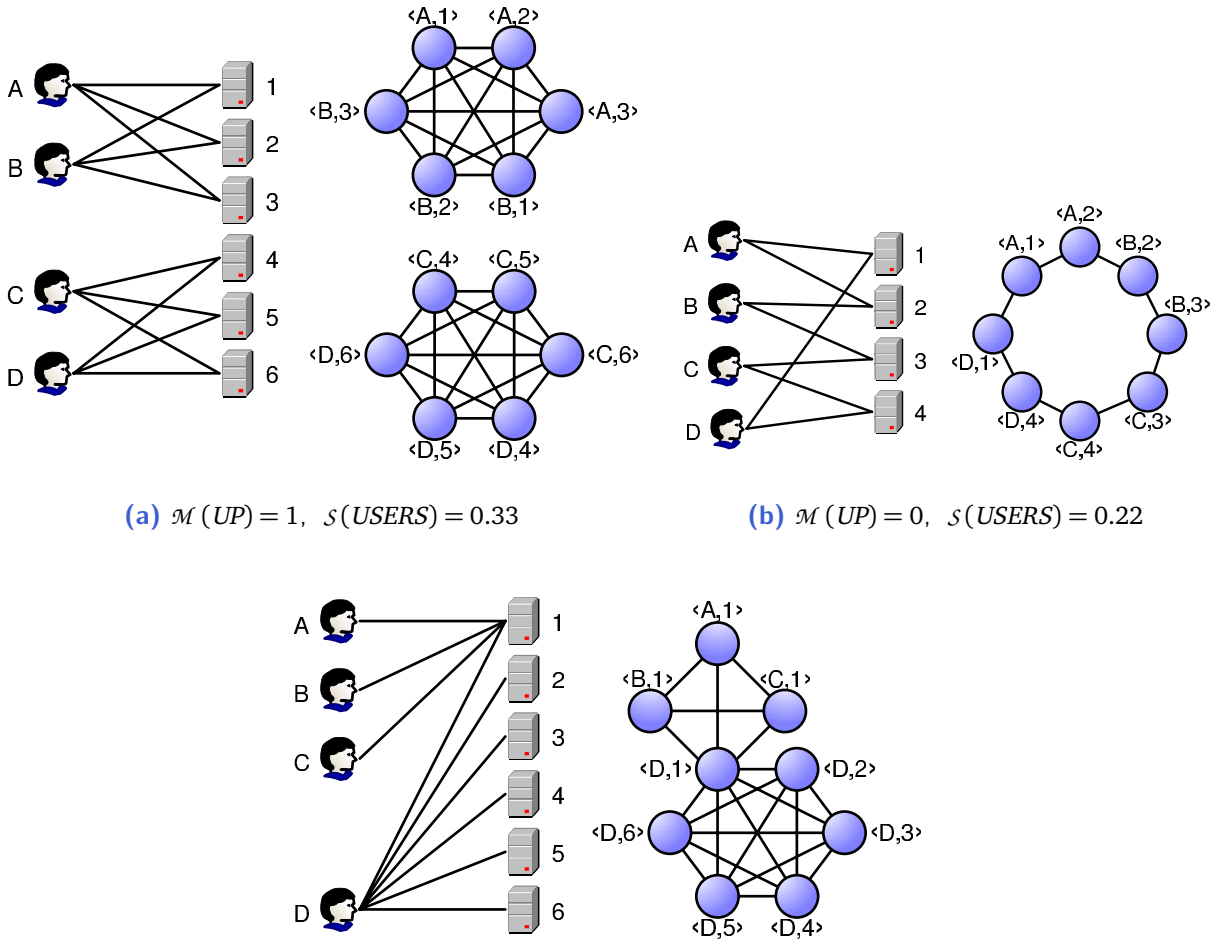
- $m(\omega) = 1 \iff |MERS_\omega| = 1$;
- $m(\omega) = 0 \iff |MERS_\omega| = |B(\omega)|$;
- $m(\omega) \in (0,1) \iff 1 < |MERS_\omega| < |B(\omega)|$.

PROOF The proof immediately follows from Theorem 6.3 by using MERs in lieu of maximal bicliques. ■

The previous theorem allows us to make some consideration on the complexity of the role mining problem. Given a user-permission assignment ω, the higher its local minability is, the less the number of possible MERs to analyze is. The following subsection and Section **??** offer practical examples about this property.

7.3.3 Examples

We now show some examples to demonstrate how minability and similarity indices can actually provide role mining engineers with the expected complexity



(a) $\mathcal{M}(UP) = 1, \quad \mathcal{S}(USERS) = 0.33$

(b) $\mathcal{M}(UP) = 0, \quad \mathcal{S}(USERS) = 0.22$

(c) $\mathcal{M}(UP) = 0.94 \quad \mathcal{S}(USERS) = 0.58$

Figure 7.1 Access control configurations as bipartite graphs and corresponding unipartite graphs

to find functional and/or organizational roles. In Figure 7.1, three different and simple access control configurations are depicted. In each one, we have 4 users and 6 permissions, but with different user-permission assignments. To better illustrate these indices, we also report the corresponding unipartite graphs constructed according to Equation (7.6).

In Figure 7.1(a) the minability index is 1. In this case, it is straightforward to verify that a possible clique cover of the unipartite graph is represented by cliques $C_1 = \{\langle A,1\rangle, \langle A,2\rangle, \langle A,3\rangle, \langle B,1\rangle, \langle B,2\rangle, \langle B,3\rangle\}$ and $C_2 = \{\langle C,4\rangle,$

$\langle C,5\rangle$, $\langle C,6\rangle$, $\langle D,4\rangle$, $\langle D,5\rangle$, $\langle D,6\rangle\}$. In RBAC terms, C_1 and C_2 correspond to two maximal equivalent roles: the first one made up of permissions $\{1,2,3\}$ and it is assigned with users $\{A,B\}$, the second one made up of permissions $\{4,5,6\}$ and assigned with users $\{C,D\}$. As for the similarity index, $\mathcal{S}(\{A,B,C,D\}) = 1/3$.

Figure 7.1(b) shows another access control configuration such that the minability index is equal to 0. According to Th. 7.1, it represents the most ambiguous case. Indeed, in this example we have two possible MERs to manage each user-permission assignment (one is composed by one user and two permissions, the other one is made up of one permission and two users). Yet, without further business semantics of access control data, it is not clear which is the best choice. Another observation is that in Figure 7.1(b) the similarity index is smaller than in Figure 7.1(a). Indeed, since each permission is used by 2 users, it is impossible to define a role that has to be assigned to the majority of users.

Figure 7.1(c) shows a slightly more complicated configuration, where the minability index is between 0 and 1, while the similarity is higher than in all previous cases. It is quite clear that the unipartite graph can be covered with two cliques (i.e., two MERs), and this suggests that the minability must be very close to 1. Indeed, we have an ambiguity only for the user-permission assignment $\langle D,1\rangle$: it can belong to both the cliques $C_1 = \{\langle A,1\rangle, \langle B,1\rangle, \langle C,1\rangle, \langle D,1\rangle\}$ and $C_2 = \{\langle D,1\rangle, \langle D,2\rangle, \langle D,3\rangle, \langle D,4\rangle, \langle D,5\rangle, \langle D,6\rangle\}$. $\mathcal{M}(UP) = 0.94$ is in line with the previous observation.

7.4 Applications of Minability and Similarity

As shown in the previous section, minability and similarity are estimates of the expected complexity to select roles within the role mining results. As a consequence, they are also metrics for the likelihood of making administration errors when managing roles throughout their lifecycle. For instance, given a group of users, when the minability index equals 1 for user-permission assignments involved in the group, according to Th. 7.1 there will be just one possible maximal equivalent role for managing those assignments. This means, for example, that new permissions introduced within the system will likely be assigned to all users of the group (via the single role) or to none of them, and new users that join such a group will likely be granted the same permissions of other users (namely, all the permissions contained within the single role). Moreover, the business meaning of the role is strictly related to business aspects that users within the group have in common. Therefore, the probability of making wrong access control decisions is low.

One possible application of the minability and similarity indices is to help data analysts guide a *divide-and-conquer* approach to role mining described in Section 5.4. The reason why minability and similarity change after decomposing the problem can be analyzed in terms of the graph model. As a matter of fact, partitioning the set *UP* is equivalent to partitioning the unipartite graph *G* constructed according to Equation (7.6) in subgraphs, since each element in *UP* corresponds to a node in *G*. Hence, partitioning *UP* means discarding all the relationships between user-permission assignments that are in two different subsets of the partition—they will no longer induce a biclique—, namely removing edges in *G* that connect distinct subgraphs. In other words, the partition "breaks" roles that spread across multiple subsets into more parts; that is, roles without a clear meaning according to the business information that induced the partition. However, an important problem arises: how to be sure that the roles selected to be broken down are less relevant from a business perspective. The following section explains how minability and similarity indices can practically be used in conjunction with a divide-and-conquer approach to role mining to elicit business roles and to mitigate enterprise risk.

7.4.1 Choosing the Best Decomposition

When we have several business information at our disposal (e.g., organization units, job titles, applications, etc.), we have to select the one that induces a partition for *UP*, which minimizes the risk for each subset and that simplifies the subsequent mining steps. The best partition can change depending on the organization needs. To guide the decomposition process, it is useful to have a metric that allows data analysts:

- To decide what business information most reduces the risk of a poor role definition and simplifies the subsequent mining steps.
- To predict whether splitting the problem into more sub-problems actually reduces the risk of having ill-defined roles. In particular, we can decide to iteratively decompose the data before executing the mining step, by applying a different decomposition at each iteration until given minability and similarity thresholds are reached for each subset.
- To verify that partitioning does not actually reduce the role mining complexity. If this is the case, access control information should thus be reviewed in order to improve their manageability.

In all previous cases, similarity and minability are a means to estimate the risk related to the data being analyzed. In fact, since both indices express the likelihood of making bad administration decisions due to the unclear meaning

of roles, they can be used in conjunction with the risk formula Equation (7.2) proposed in Section 7.2.2. Depending on the kind of roles that role engineers are looking for (organizational or functional), the similarity or the minability value can be combined with the importance of each subset to evaluate the risk of incurring a poor role design. In particular, the following indicators can support the partition selection problem:

Definition 7.5 (Similarity-Based Risk) Let *USERS* be a set of users to analyze, and *PERMS*, *UP* be the corresponding permissions and assignments. The *similarity-based risk* of *USERS* is defined as:

$$Risk_{\mathcal{S}}(USERS) = \big(1 - \mathcal{S}(USERS)\big) \times C, \tag{7.13}$$

where C is the importance of the user group *USERS*, while $\mathcal{S}(USERS)$ is the similarity value computed over the users belonging to *USERS*.

Definition 7.6 (Minability-Based Risk) Let *UP* be a set of user-permission assignments between users in *USERS* and permissions in *PERMS*. The *minability-based risk* of *UP* is defined as:

$$Risk_{\mathcal{M}}(UP) = \big(1 - \mathcal{M}(UP)\big) \times C, \tag{7.14}$$

where C is the importance of the data represented by the assignment set *UP*, while $\mathcal{M}(UP)$ is the similarity value computed over the user-permission assignments belonging to *UP*.

The previous definitions offer an estimate for the risk related to each subset. However, if the objective is to identify the best partition, we will compare the values obtained for similarity-based and/or minability-based risks on all subsets and choose the partition with the lowest "average" risk. Similarly, when we want to check if further decomposing the problem actually reduces the role mining complexity, we have to compare the "average" risk that we have with and without the decomposition. The following section shows a possible approach to summarize the risk related to a partition.

7.4.2 Conditioned Indices

Instead of analyzing the risk values calculated on each subset of a given partition, in most case it is more advantageous to have a risk value that "summarizes" the simplification introduced by the partition. To this aim, we need to review all the aforementioned indices to *condition* them by the given partition. The conditioning concept will apply on both similarity and minability indices (we therefore speak of *conditioned similarity and minability*) and risk evaluation metrics (we therefore speak of *conditioned similarity- or minability-based risk*).

Conditioned Similarity and Similarity-Based Risk

Let $\Omega = \{\Omega_1, \ldots, \Omega_k\}$ be a k-partition of UP such that $\Omega_i \subseteq UP$ and $UP = \bigcup_{i=1}^{k} \Omega_i$. Each subset Ω_i induces a set of users Υ_i, such that $\Upsilon_i = \{u \in USERS \mid \exists p \in PERMS, \langle u, p\rangle \in \Omega_i\}$. According to Equation (7.4), we can define the similarity of Υ_i in the following way:

$$\mathcal{S}(\Upsilon_i) = \begin{cases} \dfrac{1}{\binom{|\Upsilon_i|}{2}} \displaystyle\sum_{\substack{u_1,u_2 \in \Upsilon_i: \\ u_1 \neq u_2}} s_{\Omega_i}(u_1, u_2), & |U| > 1; \\ 1, & \text{otherwise.} \end{cases} \tag{7.15}$$

where $\mathcal{S}_\Omega(u_1, u_2)$ is the similarity of the users u_1 and u_2 obtained by only considering the permissions that are involved in Ω_i. Equation (7.15) 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

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

We can then offer the following definition:

Definition 7.7 (Conditioned Similarity) Given a partition $\Omega = \{\Omega_1, \ldots, \Omega_k\}$ for UP such that $UP = \bigcup_{i=1}^{k} \Omega_i$ and the induced sets of users $\Upsilon_i = \{u \in USERS \mid \exists p \in PERMS, \langle u, p\rangle \in \Omega_i\}$, we define the similarity index *conditioned by* Ω as

$$\mathcal{S}_\Omega(USERS) = \frac{\sum_{i=1}^{k} \mathcal{S}(\Upsilon_i)\left(\sigma_i + \binom{|\Upsilon_i|}{2}\right)}{\sum_{i=1}^{k}\left(\sigma_i + \binom{|\Upsilon_i|}{2}\right)} = \frac{\sum_{i=1}^{k} \sigma_i + \sum_{i=1}^{k} \mathcal{S}(\Upsilon_i)\binom{|\Upsilon_i|}{2}}{\sum_{i=1}^{k} \sigma_i + \sum_{i=1}^{k} \binom{|\Upsilon_i|}{2}}. \tag{7.16}$$

Notice that Equation (7.16) holds since $\mathcal{S}(\Upsilon_i) = 1$ when $\sigma_i = 1$. Another important observation is that the conditioned index Equation (7.16) is a sort of "modified" version of Equation (7.4), where the pairs of users that belong to different subsets are discarded.

As for risk analysis, Def. 7.7 can be extended in order to take into account the importance of each subset. In particular, we provide the following definition:

Definition 7.8 (Conditioned Similarity-Based Risk) Given the k-partition $\Omega = \{\Omega_1, \ldots, \Omega_k\}$ of UP and the induced sets of users $\Upsilon_i = \{u \in USERS \mid \exists p \in PERMS, \langle u, p\rangle \in \Omega_i\}$, we define the similarity-based risk *conditioned by* Ω as

$$Risk_{s_\Omega}(USERS, \Omega) = \frac{\sum_{i=1}^{k} \left(1 - s(\Upsilon_i)\right) C_i \left(\sigma_i + \binom{|\Upsilon_i|}{2}\right)}{\sum_{i=1}^{k} \left(\sigma_i + \binom{|\Upsilon_i|}{2}\right)}, \tag{7.17}$$

where C_i is the importance of the user group Υ_i.

In particular, $Risk_{s_\Omega}(USERS, \Omega)$ is a weighted average of the risks related to each subset, where the weights are proportional to the subset cardinalities.

Conditioned Minability and Minability-Based Risk

Given a k-partition $\Omega = \{\Omega_1, \ldots, \Omega_k\}$ of UP, according to Equation (7.9) the minability index of each subset Ω_i is

$$\mathcal{M}(\Omega_i) = \frac{1}{|\Omega_i|} \sum_{\omega \in \Omega_i} m_{\Omega_i}(\omega),$$

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

Definition 7.9 (Conditioned Minability) Given a k-partition $\Omega = \{\Omega_1, \ldots, \Omega_k\}$ of UP the minability index *conditioned by* Ω is

$$\begin{aligned} \mathcal{M}_\Omega(UP) = \frac{\sum_{i=1}^{k} \mathcal{M}(\Omega_i) |\Omega_i|}{\sum_{i=1}^{k} |\Omega_i|} &= \frac{1}{|UP|} \sum_{i=1}^{k} \sum_{\omega \in \Omega_i} m_{\Omega_i}(\omega) \\ &= \frac{1}{|UP|} \sum_{\omega \in UP} m_{\Omega_i}(\omega). \end{aligned} \tag{7.18}$$

It is possible to note that the conditioned index Equation (7.18) is similar to the basic minability index Equation (7.9), except that relationships between user-permission assignments that belong to different subsets are no longer considered.

As for risk analysis, Def. 7.9 can be extended in order to take into account the importance of each subset. In particular, we provide the following definition:

Definition 7.10 (Conditioned Minability-Based Risk) Given a k-partition $\Omega = \{\Omega_1, \ldots, \Omega_k\}$ of the set UP we define the minability-based risk *conditioned by* Ω as

$$Risk_{\mathcal{M}_\Omega}(UP, \Omega) = \frac{\sum_{i=1}^{k} \left(1 - \mathcal{M}\left(\Omega_i\right)\right) C_i \left|\Omega_i\right|}{\sum_{i=1}^{k} \left|\Omega_i\right|}, \tag{7.19}$$

where C_i is the importance of the subset Ω_i.

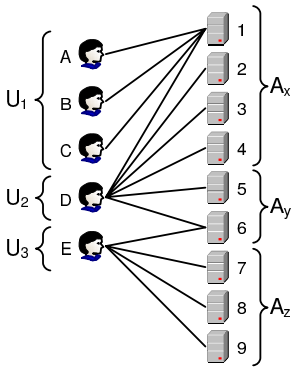
In particular, $Risk_{\mathcal{M}_\Omega}(UP, \Omega)$ is a weighted average of the risks related to each subset, where the weights are represented by the subset cardinalities.

7.4.3 Examples

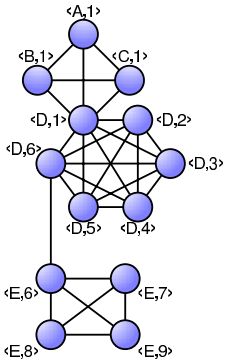
In this subsection we show a simple application of our indices. Let us assume that the access control configuration to analyze is the one depicted in Figure 7.2(a). The unipartite graph corresponding to the analyzed access control configuration is shown in Figure 7.2(b). The values of the indices are reported in the caption. We now try to split the problem into several sub-problems by leveraging some available business information in order to check whether the minability and similarity values increase, and consequently the risk indices decrease. For this purpose, suppose that we have two different business information at our disposal: the organization unit the user belongs to, and the applications involved by the given permission set. This information is depicted in Figure 7.2(a). In particular, the organization unit U_1 is composed of the users A, B, and C, while the organization units U_2 and U_3 are composed of the users D and E, respectively. As for the applications, A_x is composed of the permissions 1,2,3, and 4; A_y is composed of the permissions 5 and 6; while A_z is made up of permissions 7,8, and 9.

Given these pieces of information, we have to choose which one induces the partition that most simplifies the successive role mining steps. To ease exposition, we assume that all the subsets have the same importance. Figure 7.2(c) shows the subsets generated by partitioning according to the organization units. Notice that both minability and similarity indices are equal to 1 for each subset, whereas the risk values are 0. Hence, the conditioned basic and risk indices equal 1 and 0, respectively. Figure 7.2(d) shows the subsets generated by partitioning according to applications and the corresponding minability and similarity values for each subset.

When comparing the conditioned indices of the two partitions, it can be easily seen that the organization-units based partition is preferable to the

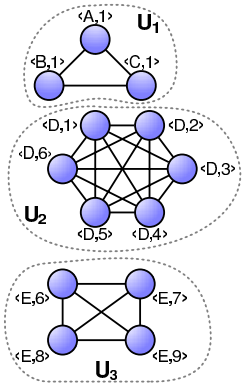


(a) Access configuration



$\mathcal{M}(UP) = 0.89,$
$\mathcal{S}(USERS) = 0.38,$
$Risk_{\mathcal{M}}(UP) = 0.11,$
$Risk_{\mathcal{S}}(USERS) = 0.62$

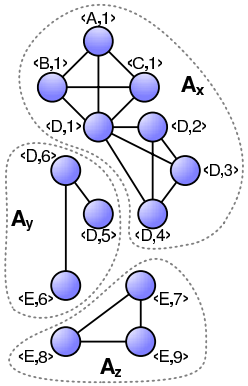
(b) Without partitioning



$$\left.\begin{array}{l}\mathcal{M}(UP_{U_1}) = 1\\ \mathcal{M}(UP_{U_2}) = 1\\ \mathcal{M}(UP_{U_3}) = 1\end{array}\right\} \Longrightarrow \left\{\begin{array}{l}\mathcal{M}_\Omega(UP) = 1\\ Risk_{\mathcal{M}_\Omega}(UP) = 0\end{array}\right.$$

$$\left.\begin{array}{l}\mathcal{S}(USERS_{U_1}) = 1\\ \mathcal{S}(USERS_{U_2}) = 1\\ \mathcal{S}(USERS_{U_3}) = 1\end{array}\right\} \Longrightarrow \left\{\begin{array}{l}\mathcal{S}_\Omega(USERS) = 1\\ Risk_{\mathcal{S}_\Omega}(USERS) = 0\end{array}\right.$$

(c) Partitioning by user attributes



$$\left.\begin{array}{l}\mathcal{M}(UP_{A_x}) = 0.91\\ \mathcal{M}(UP_{A_y}) = 0.67\\ \mathcal{M}(UP_{A_z}) = 1\end{array}\right\} \Longrightarrow \left\{\begin{array}{l}\mathcal{M}_\Omega(UP) = 0.88\\ Risk_{\mathcal{M}_\Omega}(UP) = 0.12\end{array}\right.$$

$$\left.\begin{array}{l}\mathcal{S}(USERS_{A_x}) = 0.67\\ \mathcal{S}(USERS_{A_y}) = 0.5\\ \mathcal{S}(USERS_{A_z}) = 1\end{array}\right\} \Longrightarrow \left\{\begin{array}{l}\mathcal{S}_\Omega(USERS) = 0.67\\ Risk_{\mathcal{S}_\Omega}(USERS) = 0.33\end{array}\right.$$

(d) Partitioning by permission attributes

Figure 7.2 A partitioning example

7.1 Approximation of the similarity index

```
1: procedure S̃(USERS, k)
2:     ℓ ← 0
3:     if |USERS| = 1 then
4:         return 1
5:     else
6:         for i = 1...k do
7:             Select u1, u2 ∈ USERS : u1 ≠ u2 uniformly at random
8:             ℓ ← ℓ + s(u1, u2)
9:         end for
10:        return ℓ/k
11:    end if
12: end procedure
```

applications-based partition. Indeed, the minability conditioned by applications is 0.88, which is even worse than the not-conditioned minability. Partitioning by organization units is also preferable when evaluating other conditioned indices.

7.5 Fast Index Approximation

In this section, we illustrate two algorithms to efficiently compute the indices introduced in Section 7.3.1 and Section 7.3.2.

7.5.1 Approximating the Similarity

Let us analyze the computation time required to determine the exact value of $\mathcal{S}(USERS)$. In particular, according to its definition Equation (7.4), it can be calculated in $O(|PERMS|\,|USERS|^2)$ time. Indeed, $O(|USERS|^2)$ time is required to identify all possible user pairs. For each pair $u_1, u_2 \in USERS$, the cardinality of both the intersection and the union of their granted permissions can be computed in $O(|PERMS|)$. In particular, by scanning UP only once, we can build a hashtable of permissions that each user has been granted. Notice that $|UP| \leq |USERS|\,|PERMS|$. Hence, checking if a permission is in $perms(u_1)$ requires $O(1)$. This check should be done for every permission in $perms(u_2)$, thus requiring $O(|PERMS|)$. Altogether, the similarity index can be calculated in $O(|PERMS|\,|USERS|^2)$.

To reduce the computation time, we propose the ε-approximated algorithm listed in Algorithm 7.1. The algorithm performs uniform sampling over all possible user pairs and then computes the average similarity among them. In particular, in each of the k sampling (Line 6), a user pair u_1, u_2 is randomly chosen (Line 7). Then, the variable ℓ is incremented by the similarity value of this pair (Line 8). In accordance with our definition, the returned result is ℓ/k. We now show that the algorithm is totally correct: it terminates in a finite time and provides a correct result. First, Algorithm 7.1 always terminates because its core is a finite loop. Then, the following theorem proves that the computed result is probabilistically correct:

Theorem 7.2 *The value $\widetilde{\mathcal{S}}(USERS, k)$ computed by a run of Algorithm 7.1 satisfies:*

$$\Pr\left(\left|\widetilde{\mathcal{S}}(USERS, k) - \mathcal{S}(USERS)\right| \geq \varepsilon\right) \leq 2\exp\left(-2k\varepsilon^2\right).$$

PROOF If $|USERS| = 1$ the proof is immediate. Let us consider the case when $|USERS| > 1$. We will use the Hoeffding inequality [51] to prove this theorem. The cited inequality states that if $X_1 \ldots X_k$ are independent random variables such that $0 \leq X_i \leq 1$, then

$$\Pr\left(\left|\sum_{i=1}^{k} X_i - \mathbb{E}\left[\sum_{i=1}^{k} X_i\right]\right| \geq t\right) \leq 2\exp\left(-\frac{2t^2}{k}\right), \tag{7.20}$$

where $\mathbb{E}[\cdot]$ indicates the expected value of a random variable. In our case, X_i indicates the similarity of a randomly chosen user pair. Equation (7.20) can be rewritten as

$$\Pr\left(\left|\frac{1}{k}\sum_{i=1}^{k} X_i - \mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} X_i\right]\right| \geq \varepsilon\right) \leq 2\exp\left(-2k\varepsilon^2\right), \tag{7.21}$$

where $\varepsilon = t/k$. Notice that the value $\frac{1}{k}\sum_{i=1}^{k} X_i$ is exactly the output of Algorithm 7.1. Hence, in order to prove that the algorithm gives an approximation of $\mathcal{S}(USERS)$, we have to prove that $\mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} X_i\right]$ is equal to $\mathcal{S}(USERS)$. Because of the linearity of the expectation, the following equation holds:

$$\mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} X_i\right] = \frac{1}{k}\sum_{i=1}^{k} \mathbb{E}[X_i]. \tag{7.22}$$

Since the user pair used to calculate X_i is picked uniformly at random, the corresponding similarity value is produced with a probability of $1/\binom{|USERS|}{2}$—that is, one out of all the possible (unordered) pairs. Thus, the expected value

of X_i is

$$\forall i \in 1 \dots k, \quad \mathbb{E}[X_i] = \sum_{\substack{u_1,u_2 \in USERS: \\ u_1 \neq u_2}} \frac{s(u_1,u_2)}{\binom{|USERS|}{2}}.$$

The previous equation is the definition of $\mathcal{S}(USERS)$ as in Equation (7.4) when $|USERS| > 1$, completing the proof. ■

For practical applications of Algorithm 7.1, it is possible to calculate the number of loops needed to obtain an expected error that is less than ε with a probability greater than p. The following is an application of Th. 7.2:

$$k > -\frac{1}{2\varepsilon^2} \ln\left(\frac{1-p}{2}\right). \tag{7.23}$$

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

Finally, we shall demonstrate that the computational complexity of Algorithm 7.1 is $O(|UP||PERMS|)$. Indeed, according to the observation made at the beginning of this section, we can build a hashtable of permissions possessed by users in $O(UP)$. The loop in Line 6 is repeated k times, and we reasonably assume that $k \in O(|UP|)$. Computing the similarity of two users in each loop requires $O(|PERMS|)$ thanks to the hashtable. Therefore, the total complexity is $O(|UP||PERMS|)$, that is advantageous when compared to the exact similarity calculation if the number of users is greater than the number of permissions and, most of all, when the user set is large.

7.5.2 Approximating the Minability

Here we will show that the computational complexity of calculating $\mathcal{M}(UP)$ is $O(|UP|^3)$. In the case of a large-size organization, computation may be unfeasible since UP can count hundreds of thousands of user-permission assignments. For this reason, we propose an approximation algorithm for $\mathcal{M}(UP)$ that has a computational complexity of $O(k|UP|)$.

First, let us consider the complexity of computing the exact value of $\mathcal{M}(UP)$. In Equation (7.11), the first sum is over all the user-permission assignments $\omega \in UP$, while the second one is over all the triples $\langle \omega_1, \omega_2 \rangle \in triples(\omega)$—that, for a given ω, are $(|UP|-1)(|UP|-2)$ in the worst case. Each addendum of the sum corresponds to checking whether the selected triple is also a triangle. It is a triangle if the two outer nodes $\omega_1 = \langle u, p \rangle$ and $\omega_2 = \langle u', p' \rangle$ of the selected triple induce a biclique. This occurs if $u = u'$, or $p = p'$, or other two edges $\omega_3 = \langle u, p' \rangle$ and $\omega_4 = \langle u', p \rangle$ exist in UP. It is possible to check if $u = u'$

7.2 Approximation of the minability index

```
1:  procedure M̃(UP, k)
2:      ℓ ← 0
3:      for i = 1...k do
4:          Select ω ∈ UP uniformly at random
5:          if triples(ω) ≠ ∅ then
6:              Select ⟨ω1, ω2⟩ ∈ triples(ω) uniformly at random
7:              if ω1 ∈ B(ω2) then
8:                  ℓ ← ℓ + 1
9:              end if
10:         else
11:             ℓ ← ℓ + 1
12:         end if
13:     end for
14:     return ℓ/k
15: end procedure
```

or $p = p'$ in a constant time. Instead, the search for the pair ω_3, ω_4 can be executed in $O(1)$ after having built a hashtable of all possible user-permission assignments in $O(|UP|)$. The total computational cost is thus $O(|UP|^3)$.

To reduce the computation time, we propose the ε-approximated algorithm listed in Algorithm 7.2, that is inspired by [85] but adapted to the bipartite graph case. In each of the k steps, a user-permission assignment ω is randomly chosen (Line 4). Then, two random user-permission pairs among those that induce a biclique together with ω (if any) are selected (Line 6). If these two user-permission assignments induce a biclique, the counter ℓ is incremented by 1 since we have found a triple that is also a triangle (Line 7). The ratio of the number of found triangles ℓ to the number of sampled triples k (Line 14) represents the approximated minability value.

In the following, we show that the algorithm terminates and returns a correct result. First, notice that the core of Algorithm 7.2 is a finite loop, thus it always outputs a result in a finite amount of time. The following theorem proves that this answer is probabilistically correct:

Theorem 7.3 *The value $\widetilde{\mathcal{M}}(UP, k)$ computed by a run of Algorithm 7.2 satisfies:*

$$\Pr\left(\left|\widetilde{\mathcal{M}}(UP, k) - \mathcal{M}(UP)\right| \geq \varepsilon\right) \leq 2\exp\left(-2k\varepsilon^2\right).$$

PROOF We follow the same proof schema of Th. 7.2. Let $X_1 \ldots X_k$ be independent random variables, where $X_i = 1$ if, for a randomly selected tuple

$\langle \omega, \omega_1, \omega_2 \rangle \subseteq UP \times UP \times UP$ such that $\langle \omega_1, \omega_2 \rangle \in triples(\omega)$, either $\omega_1 \in B(\omega_2)$ or $triples(\omega) = \emptyset$. Otherwise, $X_i = 0$. In this case,Equation (7.21) still holds and, in particular, $\frac{1}{k}\sum_{i=1}^{k} X_i$ is exactly the output of Algorithm 7.2. Hence, in order to prove that the algorithm gives an approximation of $\mathcal{M}(UP)$, we have to prove that $\mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} X_i\right]$ is equal to $\mathcal{M}(UP)$. Because of the linearity of the expectation,Equation (7.22) still holds. To calculate X_i we first pick a user-permission relationship uniformly at random, then we pick two user-permission relationships that make up a triple. Thus, the corresponding minability value is produced with a probability of $1/(|UP|\,|triples(\omega)|)$. Consequently, the expected value of X_i is

$$\forall i \in 1 \ldots k, \quad \mathbb{E}[X_i] = \sum_{\omega \in UP} \sum_{\langle \omega_1, \omega_2 \rangle \in triples(\omega)} \frac{Y(\omega, \omega_1, \omega_2)}{|UP|\,|triples(\omega)|} + \sum_{\omega \in UP : triples(\omega) = \emptyset} \frac{1}{|UP|}.$$

The previous equation is equivalent to the definition of $\mathcal{M}(UP)$ as in Equation (7.11), completing the proof. ∎

In the same way as the similarity index, it is possible to calculate the number of times it takes the loop in Algorithm 7.2 to obtain an expected error which is less than ε with a probability greater than p. By analyzing Th. 7.3 it can be seen that the same result Equation (7.23) holds for this case.

As for computational complexity, we will now show that Algorithm 7.2 requires a time $O(k\,|UP|)$ to run. The loop in Line 3 is repeated k times. In each loop, a random user-permission relationship $\omega \in UP$ can be selected in constant time (Line 4). Let us consider $\omega = \langle u, p \rangle$. In order to randomly select a pair $\langle \omega_1, \omega_2 \rangle$ that belongs to $triples(\omega)$, we have to calculate the set $B(\omega)$, then every possible pair of this set is in $triples(\omega)$ (Line 5). Equation (7.5) states that $B(\omega) = \{\langle u', p' \rangle \in UP \mid \langle u, p' \rangle, \langle u', p \rangle \in UP \wedge \langle u, p \rangle \neq \langle u', p' \rangle\}$. The number of elements of $B(\omega)$ is at most $|UP| - 1$, and each element can be found in $O(1)$ after having built a hashtable of all possible user-permission assignments in $O(|UP|)$. Then, the computational cost incurred to identify a biclique is at most $O(|UP|)$. Line 7 can be executed in $O(1)$ since it represents a search in the hashtable to verify the conditions in Equation (7.5). Therefore, the computational complexity of Algorithm 7.2 is $O(k\,|UP|)$, which greatly improves over the time required to calculate the exact value $\mathcal{M}(UP)$. This improvement is traded-off with a slight (tunable) decrease in the precision of the computed value $\widetilde{\mathcal{M}}(UP, k)$.

7.5.3 Approximation of Conditioned Indices

We now demonstrate that the amount of approximation introduced by the proposed randomized algorithms, when applied to the calculation of conditioned

indices, is comparable to the approximation of the non-conditioned indices.

Theorem 7.4 *Let $\mathcal{S}_\Omega(USERS)$, $Risk_{\mathcal{S}_\Omega}(USERS,\Omega)$, $\mathcal{M}_\Omega(UP)$, and $Risk_{\mathcal{M}_\Omega}(UP,\Omega)$ be the exact indices conditioned by a given partition Ω according to definitions 7.7, 7.8, 7.9, and 7.10, respectively. Let $\widetilde{\mathcal{S}}_\Omega(USERS,k)$, $\widetilde{Risk}_{\mathcal{S}_\Omega}(USERS,\Omega,k)$, $\widetilde{\mathcal{M}}_\Omega(UP,k)$, and $\widetilde{Risk}_{\mathcal{M}_\Omega}(UP,\Omega,k)$ be the corresponding approximated values computed by adopting Algorithm 7.1 and Algorithm 7.2 for each subset $\Omega_i \in \Omega$. Then:*

$$\Pr\left(\left|\widetilde{\mathcal{S}}_\Omega(USERS,k)-\mathcal{S}_\Omega(USERS)\right|\geq\varepsilon\right)\leq 2\exp\left(-2k\varepsilon^2\right)$$
$$\Pr\left(\left|\widetilde{Risk}_{\mathcal{S}_\Omega}(USERS,\Omega,k)-Risk_{\mathcal{S}_\Omega}(USERS,\Omega)\right|\geq\varepsilon\right)\leq 2\exp\left(-2k\varepsilon^2\right)$$
$$\Pr\left(\left|\widetilde{\mathcal{M}}_\Omega(UP,k)-\mathcal{M}_\Omega(UP)\right|\geq\varepsilon\right)\leq 2\exp\left(-2k\varepsilon^2\right)$$
$$\Pr\left(\left|\widetilde{Risk}_{\mathcal{M}_\Omega}(UP,\Omega,k)-Risk_{\mathcal{M}_\Omega}(UP,\Omega)\right|\geq\varepsilon\right)\leq 2\exp\left(-2k\varepsilon^2\right).$$

PROOF We first demonstrate that the theorem holds true for the approximation introduced by the given algorithms for the conditioned minability. Let ε_i be the approximation for each subset Ω_i, namely:

$$\widetilde{\mathcal{M}}(\Omega_i,k)=\mathcal{M}(\Omega_i)+\varepsilon_i$$

The approximated conditioned value will thus be:

$$\widetilde{\mathcal{M}}_\Omega(UP,k)=\frac{\sum\widetilde{\mathcal{M}}(\Omega_i,k)\left|\Omega_i\right|}{\sum\left|\Omega_i\right|}=\mathcal{M}_\Omega(UP)+\frac{\sum\varepsilon_i\left|\Omega_i\right|}{\sum\left|\Omega_i\right|} \tag{7.24}$$

According to Th. 7.3, we have that $\left|\varepsilon_i\right|\geq\varepsilon$ with probability less than $2\exp\left(-2k\varepsilon^2\right)$, hence $\left|\sum\varepsilon_i\left|\Omega_i\right|/\sum\left|\Omega_i\right|\right|\geq\varepsilon$ with probability less than $2\exp\left(-2k\varepsilon^2\right)$. Put another way, $\Pr\left(\left|\widetilde{\mathcal{M}}_\Omega(UP,k)-\mathcal{M}_\Omega(UP)\right|\geq\varepsilon\right)\leq 2\exp\left(-2k\varepsilon^2\right)$. Thus completing the proof for the approximated conditioned minability.

The proof for other conditioned indices can simply be obtained by replacing Equation (7.24) with the corresponding index definitions. ■

7.6 Analysis of a Real Case

To demonstrate the usefulness of the proposed indices, we show how they have been applied to a real case. Our case study has been carried out on a large private organization. We examined a representative organization branch that contained 1,363 users with 5,319 granted permissions, resulting in a total

of 84,201 user-permission assignments. To apply our approach we used two information sources: the organization unit (OU) chart, and a categorization of the users based on their job titles. In order to protect organization privacy, all names reported in this chapter for organization units and job titles are slightly different from the original ones. We calculated all the indices described in the previous sections by adopting Algorithm 7.1 and Algorithm 7.2 with $k = 5,000$, hence obtaining an error of less than 0.02 with a probability higher than 96%.

The remainder of this section is organized as follows. In Section 7.6.1 we will show two examples that have different values for minability and similarity, thus making it possible to better understand the meaning of having high or low values associated to these indices. In turn, in Section 7.6.2 we will apply our methodology in order to select the best available top-down information to decompose the problem. To further demonstrate the reliability of the methodology, we borrow from biology the methodology of introducing a *control* test. That is, we try to categorize users according to the first character of their surname. Since this categorization does not reflect any access control logic, we will analytically show that—as expected—it never helps the mining phase. Finally, in Section 7.6.3 we will use the proposed methodology in conjunction with the organizational unit chart to "drill-down" into smaller role mining problems.

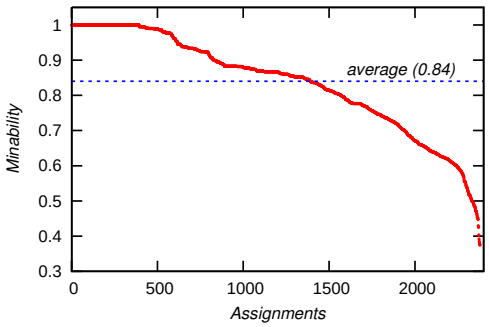
7.6.1 High and Low Values of Minability and Similarity

Figure 7.3 shows the user-permission assignments for two distinct sets of users that belong to two chosen branches of the analyzed organization. The two OUs are comparable in terms of number of users, permissions, and user-permission assignments: Figure 7.3(a) is related to 54 users who possess 285 permissions through 2,379 user-permission assignments; Figure 7.3(b) represents 48 users who possess 299 permissions through 2,081 user-permission assignments. Assignments are depicted in a matrix form, where each row represents a user, each column represents a permission, and a black cell indicates a user with a given permission granted. By using the role mining algorithm described in Chapter 4, we computed all possible maximal equivalent roles. Then, rows and columns have been sorted so that roles with the largest number of users and permissions appear as "big" areas of contiguous painted cells.

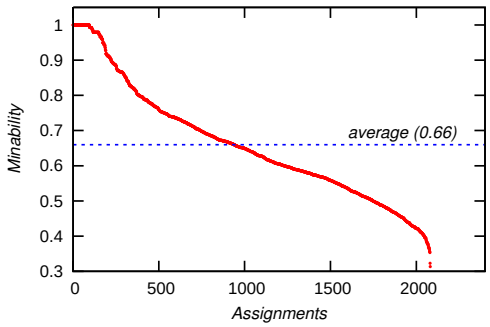
Figure 7.3(a) is an example of high values for minability (i.e., 0.84) and similarity (i.e., 0.43). It visually demonstrates how, in this case, it is easy to identify candidate roles—few groups of contiguous cells that cover most of the assignments can be easily identified via a visual inspection. The role identification task clearly requires more effort in Figure 7.3(b), in line with lower values

(a) Example of high similarity (0.43) and high minability (0.84)

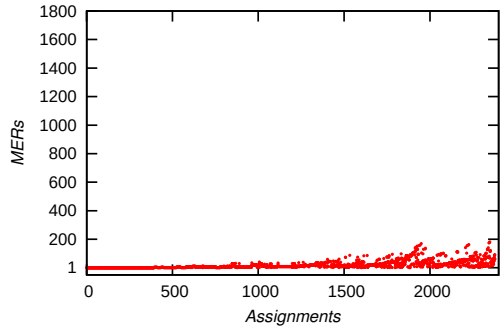
(b) Example of low similarity (0.21) and low minability (0.66)



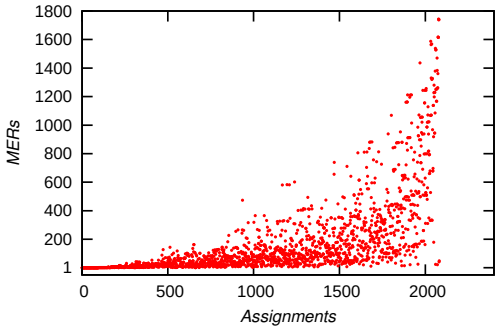
(c) Local minabilities for Figure (a)



(d) Local minabilities for Figure (b)



(e) MERs for Figure (a)



(f) MERs for Figure (b)

Figure 7.3 Examples of different values for similarity and minability. Figures (a) and (b) depict user-permission assignments in a matrix form, where each black cell indicates a user (row) that has a certain permission (column) been granted. Figures (e) and (f) show the number of MERs which cover each user-permission assignment, sorted by the descending local minability values reported in (c) and (d).

for minability (0.66) and similarity (0.21). Indeed, it is impossible to define an organizational role composed of as many users and permissions as in the previous case, and it is harder to identify roles in general. This intuition is also supported by Figure 7.3(e) and Figure 7.3(f). In these pictures we show the number of possible MERs that can be used to manage each user-permission assignment of Figure 7.3(a) and Figure 7.3(b), respectively. Assignments are sorted by descending local minabilities, and the corresponding minability values are reported in Figure 7.3(c) and Figure 7.3(d). In the first case, the number of assignments with a local minability close to 1 is higher than in the second case. This is reflected by the number of MERs that cover each user-permission assignment, that is lower in the first case. Put another way, the ambiguity of selecting the role to manage each user-permission assignment is lower in the first example. This is in line with Th. 7.1, which states that when the local minability of an assignment is equal to 1, there is only one MER to choose. The more the minability is far from 1, the more the number of MERs that can be used to manage that assignment increases, indicating that the identification of the "best" role-set requires more effort and, consequently, it is more error prone.

7.6.2 Selection of the Best Business Information

In this section we summarize an implementation of our divide-and-conquer approach. As anticipated before, we had at our disposal two top-down pieces of information—OU and job titles—, and we wanted to choose the one that mostly simplifies the subsequent mining steps. As a *control*, we also introduced a third "artificial" information without any relation to the business—the first letter of user's surname. We generated three groups of users: A–G, H–P, and Q–Z. Obviously, we did not expect that this information would help the identification of roles. Indeed, experimental results confirmed our expectations, as shown later on.

The information about job titles was only available inside OU branches at the second level of the OU tree. Therefore, we first decomposed the problem by using the first OU level. Table 7.1 sums up the results for one of the first level OUs, namely the branch Operations. As required by Def. 7.8 and Def. 7.10, for both OUs and job titles we estimated the impact of harmful administration actions for each potential group of users—due to the large number of involved job titles, in Table 7.2 we only report the impact classification for OUs. Each group of users had been classified as "Low", "Medium", or "High" impact. Adopting a three-point scale made it easier to reach consensus among administrators. The values assigned to those impact classes were conventionally set by administrators to 1, 5, and 10, respectively.

Table 7.1 Conditioned indices for the sample organization branch

Organization Unit	Index Type	Similarity	Minability	Similarity-Based Risk	Minability-Based Risk	Roles	msec
Operations							
Manufacturing							
	Not Conditioned	0.08	0.68	4.59	1.58	10,001	242
	Conditioned by Alphabetical Groups	0.08	0.74	7.30	1.70	4,422	134
	Conditioned by Organization Units	0.10	0.78	4.40	0.97	4,424	125
	Conditioned by Job Titles	0.29	0.89	3.23	0.53	918	59
Product Development							
	Not Conditioned	0.20	0.82	4.01	0.92	4,007	150
	Conditioned by Alphabetical Groups	0.20	0.84	6.24	1.11	2,511	73
	Conditioned by Organization Units	0.37	0.86	4.83	1.05	2,080	76
	Conditioned by Job Titles	0.38	0.90	5.63	0.64	818	36
Material Management							
	Not Conditioned	0.28	0.76	0.72	0.24	36,620	276
	Conditioned by Alphabetical Groups	0.28	0.80	5.58	1.28	3,504	48
	Conditioned by Organization Units	0.33	0.85	0.69	0.17	1,224	25
	Conditioned by Job Titles	0.28	0.82	0.72	0.21	21,614	105
Sales							
	Not Conditioned	0.07	0.63	4.64	1.85	61,933	471
	Conditioned by Alphabetical Groups	0.08	0.72	8.15	2.23	11,659	81
	Conditioned by Organization Units	0.11	0.67	1.63	1.39	50,314	301
	Conditioned by Job Titles	0.11	0.80	4.21	0.73	1,757	30
Quality							
	Not Conditioned	0.08	0.89	4.61	0.57	40	2
	Conditioned by Alphabetical Groups	0.08	0.94	8.35	0.58	36	1
	Conditioned by Organization Units	0.12	0.94	5.92	0.45	24	2
	Conditioned by Job Titles	0.21	0.96	3.61	0.22	25	1
Logistics							
	Not Conditioned	0.24	0.71	3.78	1.43	1,677	19
	Conditioned by Alphabetical Groups	0.24	0.80	6.47	1.60	642	13
	Conditioned by Organization Units	0.33	0.80	3.35	0.87	854	16
	Conditioned by Job Titles	0.64	0.83	0.59	0.45	357	12

For each index, the best values among all available partitions is highlighted in gray in Table 7.1. First, notice that alphabetical groups are never preferred, since they do not capture any pattern or commonality among users within access control data. For the other pieces of information, different cases can be identified:

- Within Manufacturing, partitioning by job titles is the best choice according to all indices. This means that job title is a good user's attribute to use when defining administration rules for the assignment of roles with users belonging to Manufacturing.
- Even though the job title concept is closer to the "role" concept, partitioning by job titles is not always the best choice. Material Management shows a case where the best partition is based on OUs. In this case, this is

justified by the fact that the majority of the users have the same job title. Hence, partitioning does not actually improve the mining complexity.

- The unit Sales shows a configuration where the minability and similarity indices suggest to partition by job title, but the risk indices promote the OU information. This happens because the unit Logistics contains many users that have a medium impact, and such users are not as similar among them as those having job titles with medium impact.

- Partitioning is *not* always advantageous. For instance, all users within the unit Product Development have some commonalities in their permissions that will be lost when decomposing—as a matter of fact, users within Marketing have a medium impact and are not similar among them. Thus, if the role engineering objective is to find organizational roles for Product Development, it is better to analyze the unit as a whole.

- As for the mining complexity, the number of maximal roles elicited by the role mining algorithm described in Chapter 4 is in line with the minability and similarity indices. Few roles also means less elaboration time, thus resulting in faster algorithm runs.

The previous examples also demonstrate that, in general, the choice of the best index to use (similarity, minability, similarity-based risk index, or minability-based risk index) depends on the main objective of the role engineering task.

7.6.3 Drill Down

We now show an application of the proposed methodology when hierarchical information is available. Suppose that the only available top-down information is the organizational unit chart. Table 7.2 shows index values obtained by iteratively applying a decomposition based on OUs for the unit Operations. First, notice that partitioning users according to the second level of the OU tree raised the values of both indices for most OUs. For instance, Product Development, which holds approximately one third of the branch Operations, has a minability of 0.82 and a similarity of 0.20. Conversely, Manufacturing still has low values for those indices. This means that it would be easier to find optimal organizational and functional roles for Product Development rather than for Manufacturing. Moreover, the average increase of minability is reflected by a lower number of possible MERs, dropped down from 219,086 to 114,278.

Another observation is that partitioning always reduces the number of users, permissions, and user-permission assignments to analyze, but the values

Table 7.2 Further decomposition of the sample organization branch

Organization Unit	Users	Permissions	User-Perms Assignments	Impact	Similarity	Minability	Similarity-Based Risk	Minability-Based Risk	Roles	msec
Operations	**946**	**3,647**	**56,905**	**Medium**	**0.07**	**0.65**	**4.63**	**1.73**	**219,086**	**3,637**
Manufacturing	**379**	**1,810**	**20,400**	**Medium**	**0.08**	**0.68**	**4.59**	**1.58**	**10,001**	**242**
Parent	1	64	64	Low	1.00	1.00	0.00	0.00	1	0
Technology	49	323	2,342	Low	0.35	0.92	0.65	0.08	127	6
Control	26	577	2,396	Low	0.30	0.81	0.70	0.19	254	7
Test & Quality	8	226	301	Low	0.09	0.92	0.91	0.08	17	1
Production	288	1,452	15,226	Medium	0.09	0.75	4.54	1.26	4,013	111
Plants	7	39	71	Low	0.12	0.83	0.88	0.17	12	0
Product Development	**319**	**1,341**	**14,222**	**Medium**	**0.20**	**0.82**	**4.01**	**0.92**	**4,007**	**150**
Parent	2	32	33	Low	0.03	0.98	0.97	0.02	3	0
Engineering	77	559	2,898	Medium	0.36	0.82	3.22	0.90	564	16
Design #1	88	361	3,275	Medium	0.41	0.84	2.94	0.80	10	1
Design #2	121	739	6,965	High	0.35	0.87	6.48	1.34	232	10
Design #3	6	115	214	Medium	0.14	0.93	4.28	0.36	1,205	47
Marketing	17	404	663	Medium	0.08	0.91	4.58	0.45	57	2
Innovation	8	92	174	Medium	0.23	0.97	3.85	0.16	9	0
Material Management	**58**	**1,038**	**8,670**	**Low**	**0.28**	**0.76**	**0.72**	**0.24**	**36,620**	**276**
Parent	3	286	368	Low	0.17	0.95	0.83	0.05	6	1
Purchase Dept #1	23	549	3,043	Low	0.27	0.80	0.73	0.20	745	10
Purchase Dept #2	13	407	2,136	Low	0.49	0.89	0.51	0.11	382	6
Purchase Dept #3	7	406	1,054	Low	0.29	0.77	0.71	0.23	56	3
Purchase Dept #4	5	311	972	Low	0.69	0.91	0.31	0.09	15	2
Saving Control	3	203	303	Medium	0.32	0.86	3.40	0.70	7	1
Analysis & Reporting	4	376	794	Low	0.35	0.87	0.65	0.13	13	2
Sales	**100**	**1,531**	**9,483**	**Medium**	**0.07**	**0.63**	**4.64**	**1.85**	**61,933**	**471**
Parent	1	68	68	Low	1.00	1.00	0.00	0.00	1	0
Logistics	36	1,142	6,836	Medium	0.24	0.63	3.78	1.84	49,256	279
Support	63	795	2,579	Low	0.07	0.76	0.93	0.24	1,057	22
Quality	**16**	**272**	**697**	**Medium**	**0.08**	**0.89**	**4.61**	**0.57**	**40**	**2**
Parent	1	20	20	Low	1.00	1.00	0.00	0.00	1	0
Certification	2	40	46	Low	0.15	0.92	0.85	0.08	3	0
Audit	6	188	352	High	0.20	0.94	8.00	0.60	7	1
Quality Center	7	151	279	Medium	0.07	0.93	4.67	0.36	13	1
Logistics	**73**	**1,078**	**3,432**	**Medium**	**0.24**	**0.71**	**3.78**	**1.43**	**1,677**	**19**
Parent	0	0	0	Low	-	-	1.00	1.00	0	0
Methodologies	2	350	549	Low	0.57	0.92	0.43	0.08	3	1
Planning	7	311	682	Low	0.41	0.88	0.59	0.12	38	2
Distribution #1	62	464	1,780	Medium	0.32	0.70	3.38	1.52	810	12
Distribution #2	2	351	421	Medium	0.20	0.93	4.00	0.37	3	1

of minability and the similarity do not rise proportionally. For example, Logistics has a minability of 0.71, but his child Distribution #1 has a minability of 0.70, indicating that the "mess" of Logistics is likely concentrated in Distribution #1, as confirmed by the number of MERs. Further, Product Development has much more users than Logistics, but it also has a higher minability. Moreover, high values for similarity imply high minability as well, but the inverse does not hold. If similarity is close to 1, then all users possess the same permissions; thus, minability is also close to 1. The opposite is false. For example, Distribution #2 shows a high minability (0.93) and a low similarity (0.20).

There are other examples of different trends for minability and similarity when compared to the number of users or permissions. For instance, let us consider Marketing and Purchase Dept #2 which have similar number of users and permissions. In the first case, we have 0.91 for minability and

0.08 for similarity, while in the second case we have 0.89 for minability (less than the previous case) and 0.49 for similarity (more than the previous case). Thus, this confirms that there is no direct relation between these two indices, but both are helpful to address role elicitation by highlighting two different aspects of the user-permission set. Indeed, if the objective of role engineers is to elicit organizational roles, the similarity helps to identify the OUs where an organizational role that covers a relevant number of user-permission assignment exists. This happens, for instance, for Purchase Dept #4 because of the similarity of 0.69. On the other hand, if the objective of role engineers is to find functional roles, they have to consider the minability index. For example, the sub-branch Innovation is likely to be an easily solvable sub-problem due to a minability of 0.97, as confirmed by the low number of possible MERs.

As for risk indices, Table 7.2 highlights the behavior with respect to minability and similarity. For example, although the unit Audit has higher values for minability and similarity than the parent unit Quality, the high impact of the tasks performed by involved users compels a careful role design. According to this example, decomposing is a possible way to highlight data that requires particular attention from a risk management perspective. Another aspect to take into account is the required granularity for the partition. The most sensible approach is probably to stop decomposing when the risk indices do not increase or even increase slightly. For example, as shown in Table 7.1, by partitioning Product Development according to its sub-units, the similarity-based risk index increases from 4.01 to 4.83, while the minability-based risk index grows from 0,92 to 1,05, reducing the gain in performing sub-unit driven analysis.

7.7 Ranking Users and Permissions

In this chapter, we introduce a risk analysis framework that allows to evaluate the risk incurred when managing users and permissions through RBAC. A distinguishing feature of our approach is that it can be used without having already defined roles, namely in a pre-engineering phase. By evaluating the risk level of a single user or a single permission, we make it possible to produce a ranking of users and permissions, highlighting those that most deviate from others in comparison to available user-permission relationships. Consequently, we are able to identify those users and permissions that represent the most (likely) dangerous and error prone ones from an administration point of view. Having this ranking available during the role engineering phase allows data analysts and role engineers to highlight users and permissions that are more prone to error and misuse when designed roles will be operating.

Definition 7.11 (Role Weight) Given a role $r \in ROLES$, let U_r and P_r be the sets of users and permissions associated to r, that is $U_r = \{u \in USERS \mid \langle u, r\rangle \in UA\}$ and $P_r = \{p \in PERMS \mid \langle p, r\rangle \in PA\}$. We indicate with $w\colon ROLES \to \mathbb{N}$ the *weight* function of roles, defined as $w(r) = |U_r| \times |P_r|$.

Definition 7.12 (t-stability) Let Σ_{UP} be the set of all RBAC states that cover the user-permission assignments of UP, that is all $\langle ROLES, UA, PA\rangle \in \Sigma_{UP}$ such that $\forall \langle u, p\rangle \in UP \implies \exists r \in ROLES : \langle u, r\rangle \in UA, \langle p, r\rangle \in PA$. Given $\langle u, p\rangle \in UP$, let $\mathcal{R} : UP \to 2^{(\bigcup_{\langle ROLES,UA,PA\rangle \in \Sigma_{UP}} ROLES)}$ be the function that identifies the roles which could be used to manage $\langle u, p\rangle$, that is:

$$\mathcal{R}(\langle u, p\rangle) = \bigcup_{\langle ROLES,UA,PA\rangle \in \Sigma_{UP}} \{r \in ROLES \mid \langle u, r\rangle \in UA,\ \langle p, r\rangle \in PA\}.$$

We say that $\langle u, p\rangle$ is *t-stable* if it can be managed with at least one role r with weight $w(r) \geq t$, namely $\exists r \in \mathcal{R}(\langle u, p\rangle) : w(r) \geq t$.

If an assignment $\langle u, p\rangle \in UP$ is t-stable, it is also $(t - i)$-stable for each $i = 1, \ldots, t$. We are thus interested in the maximal stability of a given assignment, namely the maximum t that verifies the t-stability condition:

Definition 7.13 (Maximal Stability) The *maximal stability* of an assignment $\langle u, p\rangle \in UP$ is the maximum t such that the assignment is t-stable. It is identified by the function $t^*\colon UP \to \mathbb{N}$ such that $t^*(\langle u, p\rangle) = max_{r \in \mathcal{R}(\langle u,p\rangle)}\, w(r)$.

The rational behind the introduction of the *stability* concept is that if an assignment can only be managed by roles with a limited weight, it represents an outlier. Indeed, only few users and permissions are involved in a role together with that assignment. System administrators are willing to manage roles with high weights—that is, which involve many users and many permissions—for several reasons. First, the benefits of using RBAC increase because there are fewer user-role and role-permission relationships to manage. Second, these roles represent relevant portions of the whole access control system of the company. Because of this relevance, they have a greater meaning for system administrators. Conversely, when an assignment cannot be managed with a high-weight role, it represents a portion of data which appears to be inconsistent with the remainder of that dataset. It might not be an error, but from the system administrator point of view, it is riskier than others. In other words, the risk of making mistakes when managing roles with a limited weight is higher: they are roles that are not used frequently, and are in some way obscure to administrators. We now introduce another function that will be used to evaluate the risk incurred when managing a single assignment:

Definition 7.14 The function $\mathcal{N} : UP \rightarrow 2^{UP}$ indicates the assignments of UP which can be managed together with the given assignment, namely:

$$\mathcal{N}(\langle u, p\rangle) = \{\langle u', p'\rangle \in UP \mid \langle u, p'\rangle, \langle u', p\rangle \in UP,\ \langle u, p\rangle \neq \langle u', p'\rangle\}.$$

The following Lemma relates $\mathcal{N}(\langle u, p\rangle)$ with the t-stability concept:

Lemma 7.3 *Given an assignment $\omega = \langle u, p\rangle \in UP$, then $|\mathcal{N}(\omega)|$ is an upper bound for $t^*(\omega)$.*

PROOF By definition, all the single assignments that could be managed in the same role together with $\langle u, p\rangle$ belongs to $\mathcal{N}(\omega)$. Hence, a role that contains more than $|\mathcal{N}(\omega)|$ assignments cannot exist, concluding the proof. ■

In Chapter 6 we proposed a practical approach to calculate $|\mathcal{N}(\omega)|$ for each $\omega \in UP$. We described two algorithms: a deterministic algorithm that is able to calculate the exact value for $|\mathcal{N}(\omega)|$ in $O(|UP|^2)$ time, while a randomized algorithm offers an ε-approximated result with a computational complexity of $O(k\,|UP|)$, where k is a parameter that can be arbitrarily chosen. Therefore, the joint usage of Lemma 7.3 and the algorithms described in Chapter 6 makes it possible to practically find an upper-bound for the maximal stability of each assignment belonging to UP. In the next section, we will show how to leverage this information to assign a risk level to a particular user or a particular permission.

We will adapt the risk model of Section 7.2.2. Further, we will use the t-stability concept to give to each user-permission assignment a probability of occurrence for each risk factor. In particular, given the assignment $\omega = \langle u, p\rangle \in UP$, we define the risk probability of ω as:

Definition 7.15 (Risk Probability of an Assignment) Given an assignment $\langle u, p\rangle \in UP$, the risk probability of $\langle u, p\rangle$ is a function ass_risk: $UP \rightarrow [0, 1]$ such that:

$$ass\_risk(\langle u, p\rangle) = 1 - \frac{t^*(\langle u, p\rangle)}{|UP|}.$$

The rationale behind the risk probability function is the following: The more $t^*(\langle u, p\rangle)$ is close to $|UP|$, the more the risk level of the assignment ω is close to 0. Indeed, if an assignment can be managed by a single role that covers almost all assignments in UP, the user-permission assignment reflects a permission granted to the majority of the users in the dataset. Note that we are not assuming the presence of such a role among those used in the RBAC configuration, but we are only saying that such a role can exist. This

consideration allows us to use our risk model in a pre-mining phase, when roles have not yet been decided on.

According to Lemma 7.3, we can quickly estimate an upper bound for $t^*(\langle u, p\rangle)$, and therefore a lower bound for the risk function:

Lemma 7.4 (Lower-Bound for Risk Probability of an Assignment) *Given an assignment $\langle u, p\rangle \in UP$, then*

$$ass\_risk(\langle u, p\rangle) \leq 1 - \frac{\left|\mathcal{N}(\langle u, p\rangle)\right|}{|UP|}.$$

PROOF The proof immediately follows from Definition 7.15 and Lemma 7.3.■

By leveraging the above concepts, we can evaluate the risk probability for users and permissions in the following way:

Definition 7.16 (Risk Probability of a User) Given an user $u \in USERS$, the risk probability of u is a function $user\_risk$: $USERS \rightarrow [0, 1]$ defined as:

$$user\_risk(u) = \sqrt{\frac{\sum_{p \in perms(u)} ass\_risk^2(\langle u, p\rangle)}{|perms(u)|}}. \tag{7.25}$$

Definition 7.17 (Risk Probability of a Permission) Given a permission $p \in PERMS$, the risk probability of p is a function $perm\_risk$: $PERMS \rightarrow [0, 1]$ defined as:

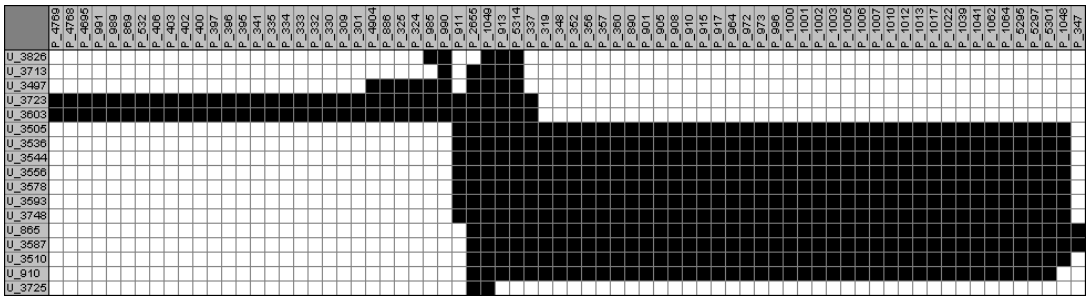
$$perm\_risk(p) = \sqrt{\frac{\sum_{u \in users(p)} ass\_risk^2(\langle u, p\rangle)}{\left|users(p)\right|}}. \tag{7.26}$$

By considering the root mean square instead of the arithmetic mean we give more importance to high risk values.

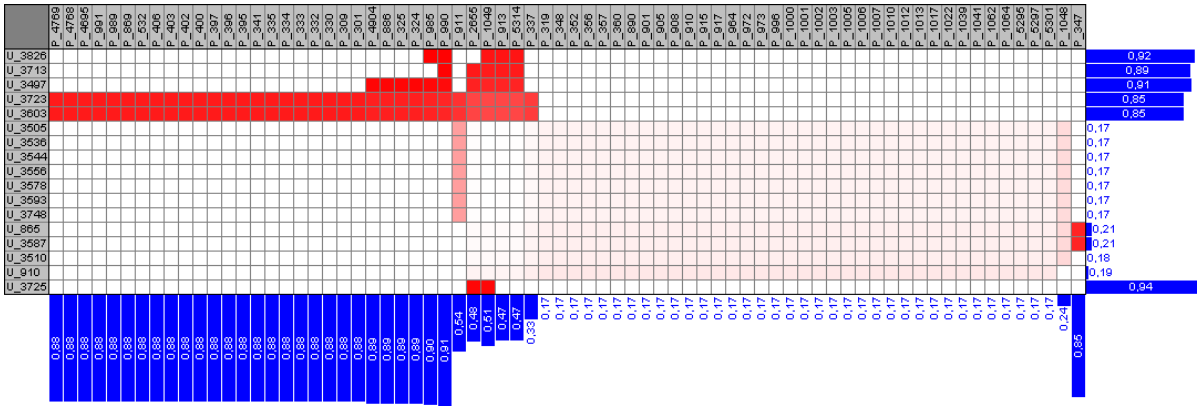
7.7.1 Experimental Results

We now show an application of our risk framework to a set of real data. Our case study has been carried out on a large private organization. Due to space limitation, we only report on a representative organization branch that contains 17 users and 72 permissions, counting 560 assignments.

Figure 7.4(a) depicts user-permission assignments in a matrix form, where each row represents a user, each column represents a permission, and a black cell indicates a user with a granted permission. Figure 7.4(b) depicts the same



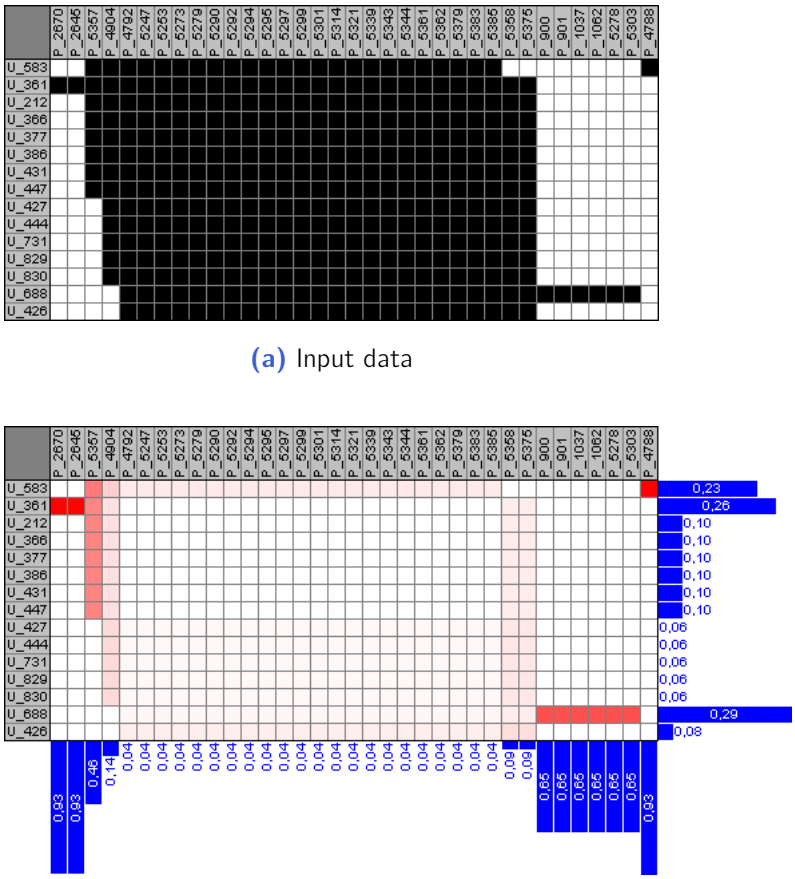
(a) Input Data



(b) Risk probabilities

Figure 7.4 Risk probability of users and permissions in *UP*

access control configuration, but the assignments colors indicate the corresponding risk probabilities. In particular, the cell color goes from red to white: Red means that the assignment has a high risk level when managed through RBAC; white means that it has a low risk level. Histograms on columns and rows borders respectively report the risk probability of managing permissions and users. Note that there are 6 users that are likely to be risky, mainly because they have a set of granted permissions that the majority of the other users do not have. This set is easily identifiable by looking at the permission histograms: Almost all the first half of the permissions are risky to manage. It is also possible to note that among the high risk users, two users have a slightly minor risk level compared to the other four. Indeed, these two users



(a) Input data

(b) Risk probabilities

Figure 7.5 Low risk users and permissions

have similar permissions granted, and this is recognized as a kind of pattern within the data that reduces the overall risk.

Figure 7.5(a) depicts another access control configuration relative to a different branch of the same organization, while Figure 7.5(b) depicts the result of our risk function applied to this branch. Here, the risk levels of all the users are lower than 0.30. It means that, when adopting RBAC, the risk level is generally lower than in the previous example. In other words, role administration should make less mistakes in this second branch than in the first one.

7.8 Final Remarks

This chapter describes a methodology that helps role engineers leverage business information during the role mining process. In particular, we demonstrate that by dividing data to analyze into smaller, more homogeneous subsets, it practically leads to more meaningful roles from a business perspective, hence decreasing the risk to make errors in managing them. To drive this process, two indices, referred to as minability and similarity, have been introduced. These indices are used to measure the expected complexity of analyzing the outcome of bottom-up approaches. In particular, we have shown how to apply such indices: to predict the effort needed to execute a role mining task over a set of user-permission assignments, thus being able to choose when to split a problem in several sub-problems; and, to select the top-down information that most simplifies the subsequent mining steps when more top-down information is available to role engineers. Leveraging these indices allows to identify the decomposition that increases business meaning in elicited roles in subsequent role mining steps, thus simplifying the analysis. We also introduced two fast probabilistic algorithms to efficiently compute such indices, making them also suitable for big organization with hundreds of thousands of users and permissions. The quality of the indices is also formally assured.

Several examples, developed on real data, illustrate how to apply the tools that implement the proposed methodology, as well as its practical implications. Achieved results support the quality and the viability of the proposal.

The risk management framework introduced in this chapter allows role engineers and system administrators of an RBAC system to highlight those users and permissions that are more prone to error and misuse when roles are operating. A distinguishing feature of our proposal, other than that of being rooted on sound theory, is that role definition is not an input parameter for the risk analysis to be performed; indeed, our model only needs to know the access control configuration of the organization, optionally enriched with other business information. Finally, it has been applied on a real case, and results obtained showed the usefulness and viability of the proposal.

8 Conclusion

This chapter concludes the thesis by offering some final remarks on the contributions first summarized in Chapter 1 and then detailed in the subsequent chapters 3–7. We also point out some future research directions of the candidate.

8.1 Remarks on Contributions

In this thesis we addressed a key challenge of RBAC-oriented solutions: devising a common set of roles that meet organization's requirements. We demonstrated that all the solutions to this problem should be characterized by two main aspects. First, we pointed out the importance of basing the role-definition process on a complete analysis of how an organization functions. A wide range of users, including IT administrators, business-line managers, and human resources, should feed this process. Most important, the alignment between business and IT is of utmost importance. Second, we demonstrated that the workload of security analysts and role engineers can largely be alleviated via automated approaches to role engineering.

Starting from the list of issues drawn up in Chapter 1, the following additional observations can be made:

Meaning of Roles By introducing the administration cost concept, Chapter 3 represented a first step towards the automation of deriving optimal role-to-permission and role-to-user assignments. Further, Chapter 5 described a first attempt to measure the business meaning of roles. Hence, the joint usage of such methodologies definitely help role engineers deliver meaningful and administrable role sets.

Algorithm Performance We tackled performance issues from several view-

points. In Chapter 4 we first described how to remove redundancy within user-permission assignments, thus leading to improved mining algorithm performances. Then, we described how to estimate the minimum number of roles identifiable in the given dataset, hence allowing for the implementation of fast, approximating role mining algorithms. Finally, in Chapter 5 we introduced a divide-and-conquer approach that, by dividing the dataset in smaller parts, allows for reduced running time of mining algorithms.

Noise Within Data We demonstrated that 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 6, showing that certain user's entitlements can be discarded from the analysis when they do not benefit from adopting RBAC. Furthermore, we showed that imputing "missing" assignments can also be beneficial, that is granting permissions to users without increasing the risk of unauthorized accesses.

Problem Complexity We accomplished a reduction of the role mining problem complexity in Chapter 6 through two different approaches: on the one hand, we offered a means to isolate the unavoidable noise within data; on the other hand, we described how to decompose a complex problem in simpler sub-problems.

Risk of Unmanageable Roles In Chapter 7 we offered a ranking of users and permissions to prioritize them according to their estimated risk level. This way, we can pay more attention to users and permission that likely lead to administration problems. Additionally, we proposed an adaptation of a traditional risk assessment framework to the access control case. To make this practicable, we described how to evaluate the likelihood that elicited roles would be misused.

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

8.2 Future Work

Possible extensions of the current work are:

- As for the cost function concept, we are currently striving to identify some "rule of thumb" to define cost functions as close as possible to the actual needs of the organization. Indeed, it is still not clear what is the best way to define such a function. We are also investigating other business models that can be used to estimate the business meaning of elicited roles.

- In this thesis we proposed an heuristic that leverages the cost concept to identify a sub-optimal role set. Nonetheless, other heuristic or approximation algorithms can be implemented to offer better solutions to the problem of minimizing the cost of deploying RBAC configurations. We also demonstrated that there is a sharp concentration of the number of elicited roles around the expected value. Hence, we could leverage this theoretical result to implement approximation algorithms.

- The divide-and-conquer approach to role mining is limited to the identification of subsets of users/permissions that share the same values for a single attribute. More complex attribute combinations can be thought in order to identify the partition that best "fit" the actual organization's way to work.

- As for the identification of stable user-permission assignments, it is worth investigating 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 are used for missing values. As a matter of fact, 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 several research directions.

Bibliography

[1] A. Colantonio and R. Di Pietro. CONCISE: COmpressed 'N' Composable Integer SEt. *Information Processing Letters*, 110:644–650, 2010.

[2] A. Colantonio, R. Di Pietro, and A. Ocello. An activity-based model for separation of duty. *CoRR*, abs/0810.5351, 2008.

[3] A. Colantonio, R. Di Pietro, and A. 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, 2008.

[4] A. Colantonio, R. Di Pietro, and A. 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.

[5] A. Colantonio, R. Di Pietro, A. Ocello, and N. V. 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.

[6] A. Colantonio, R. Di Pietro, A. Ocello, and N. V. Verde. Mining stable roles in RBAC. In *Proceedings of the IFIP TC 11 24th International Information Security Conference, SEC '09*, pages 259–269, 2009.

[7] A. Colantonio, R. Di Pietro, A. Ocello, and N. V. 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.

[8] A. Colantonio, R. Di Pietro, A. Ocello, and N. V. 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.

[9] A. Colantonio, R. Di Pietro, A. Ocello, and N. V. 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.

[10] A. Colantonio, R. Di Pietro, A. Ocello, and N. V. 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.

[11] A. Colantonio, R. Di Pietro, A. Ocello, and N. V. 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*.

[12] A. Colantonio, R. Di Pietro, A. Ocello, and N. V. Verde. Taming role mining complexity in RBAC. *Computers & Security*, 29:548–564, 2010. Special Issue on "Challenges for Security, Privacy & Trust".

[13] A. Colantonio, R. Di Pietro, A. Ocello, and N. V. Verde. Visual role mining: A picture is worth a thousand roles. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2011. *To appear*.

[14] A. Colantonio, R. Di Pietro, and N. V. Verde. A novel approach to impute missing values and detecting outliers in binary matrices. *Data Mining and Knowledge Discovery (DMKD)*, 2011. *Submitted for revision*.

[15] A. Colantonio, R. Di Pietro, and N. V. Verde. Privacy preserving role-engineering. 2011. *Submitted for revision*.

[16] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. *SIGMOD Record*, 22(2):207–216, 1993.

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

[18] A. V. Aho, M. R. Garey, and J. D. Ullman. The transitive reduction of a directed graph. *SIAM Journal on Computing*, 1(2):131–137, 1972.

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

[20] F. Angiulli and C. Pizzuti. Fast outlier detection in high dimensional spaces. In *PKDD '02: Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 15–26, 2002.

[21] E. H. Armin, A. O. Schmitt, J. Lange, S. Meier-ewert, H. Lehrach, and R. Shamir. An algorithm for clustering DNA fingerprints. *Genomics*, 66:249–256, 2000.

[22] S. D. Bay and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *KDD '03: Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 29–38, 2003.

[23] E. Bertino, C. Bettini, E. Ferrari, and P. Samarati. An access control model supporting periodicity constraints and temporal reasoning. *ACM Transactions on Database Systems*, 23(3):231–285, 1998.

[24] C. M. Bishop. Variational principal components. In *Proceedings 9th International Conference on Artificial Neural Networks, ICANN '99*, pages 509–514, 1999.

[25] B. Bollobás. The chromatic number of random graphs. *Combinatorica*, 8(1):49–55, 1988.

[26] E. Celikel, M. Kantarcioglu, B. Thuraisingham, and E. Bertino. A risk management approach to RBAC. *Risk and Decision Analysis*, 1(2):21–33, 2009. IOS Press.

[27] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2006.

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

[29] E. J. Coyne and J. M. Davis. *Role Engineering for Enterprise Security Management*. Artech House, Dec. 2007.

[30] R. Crook, D. Ince, and B. 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*, pages 9–10, 2002.

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

[32] S. De Capitani Di Vimercati, S. Foresti, P. Samarati, and S. Jajodia. Access control policies and languages. *International Journal of Computational Science and Engineering (IJCSE)*, 3(2):94–102, 2007. Inderscience Publishers.

[33] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., 2001.

[34] A. Ene, W. Horne, N. Milosavljevic, P. Rao, R. Schreiber, and R. 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.

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

[36] B. S. Everitt. *Cluster Analysis*. Edward Arnold and Halsted Press, 1993.

[37] 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.

[38] A. Figueroa, J. Borneman, and T. 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.

[39] M. Frank, D. Basin, and J. 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.

[40] M. Frank, J. M. Buhmann, and D. 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.

[41] M. Frank, A. P. Streich, D. Basin, and J. 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.

[42] M. P. Gallagher, A. O'Connor, and B. Kropp. The economic impact of role-based access control. Technical report, Planning report 02-1, National Institute of Standards and Technology (NIST), 2002.

[43] F. Geerts, B. Goethals, and T. Mielikäinen. Tiling databases. In *Discovery Science*, pages 278–289, 2004.

[44] A. Ghoting, S. Parthasarathy, and M. Otey. Fast mining of distance-based outliers in high-dimensional datasets. *Data Mining and Knowledge Discovery*, 16:349–364, 2008.

[45] V. Gligor. RBAC security policy model, preliminary draft report. Technical report, R23 Research and Development Department of the National Security Agency, 1995.

[46] D. A. Grable and A. Panconesi. Fast distributed algorithms for brooks-vizing colorings. *J. Algorithms*, 37(1):85–120, 2000.

[47] W. P. Gramm, J. A. S. Leach, and T. J. Bliley. Eu privacy protection directive 2002/58/ec. Directive of the European Parliament and of the Council of 12 July 2002.

[48] W. P. Gramm, J. A. S. Leach, and T. J. 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".

[49] R. Gupta, G. Fang, B. Field, M. Steinbach, and V. 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.

[50] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2 edition, 2006.

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

[52] P. Jaccard. Etude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[53] E. Kennedy and N. Kassebaum. US Health Insurance Portability and Accountability Act (HIPAA). Act of the United States Congress in 1996.

[54] 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.

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

[56] S. Kim, E. R. Dougherty, Y. Chen, K. Sivakumar, P. Meltzer, J. M. Trent, and M. Bittner. Multivariate measurement of gene expression relationships. *Genomics*, 67:201–209, 2000.

[57] E. M. Knorr and R. T. Ng. Algorithms for mining distance-based outliers in large datasets. In *VLDB '98: Proceedings of the 24th International Conference on Very Large Data Bases*, pages 392–403, 1998.

[58] M. Kuhlmann, D. Shohat, and G. 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.

[59] N. Li, J.-W. Byun, and E. Bertino. A critique of the ANSI standard on role-based access control. *IEEE Security & Privacy*, 5(6):41–49, 2007.

[60] R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. Wiley Series in Probability and Statistics. Wiley, 1st edition, 1987.

[61] J. Liu, S. Paulsen, X. Sun, W. Wang, A. B. Nobel, and J. 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.

[62] J. Liu, S. Paulsen, W. Wang, A. Nobel, and J. 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.

[63] H. Lu, J. Vaidya, and V. 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.

[64] T. Łuczak. The chromatic number of random graphs. *Combinatorica*, 11(1):45–54, 1991.

[65] M. 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.

[66] C. J. H. McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics: Invited Chapters at the 12th British Combinatorial Conference*, number 141 in London Mathematical Society Lecture Notes Series, pages 148–188, 1989.

[67] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005.

[68] I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. Calo, and J. 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.

[69] I. Molloy, N. Li, T. Li, Z. Mao, Q. Wang, and J. Lobo. Evaluating role mining algorithms. In *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies, SACMAT '09*, pages 95–104, 2009.

[70] G. Neumann and M. 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.

[71] S. Oba, M.-A. Sato, I. Takemasa, M. Monden, K.-I. Matsubara, and S. Ishii. A bayesian missing value estimation method for gene expression profile data. *Bioinformatics*, 19(16):2088–2096, November 2003.

[72] S. Oh and S. Park. Task-role-based access control model. *Information Systems*, 28(6):533–562, 2003. Elsevier Science Publishers Ltd.

[73] S. Oh, R. S. Sandhu, and X. Zhang. An effective role administration model using organization structure. *ACM Transactions on Information and System Security (TISSEC)*, 9(2):113–137, 2006.

[74] N. Perwaiz and I. Sommerville. Structured management of role-permission relationships. In *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies, SACMAT '01*, pages 163–169, 2001.

[75] K. Puolamäki, M. Fortelius, and H. Mannila. Seriation in paleontological data using Markov Chain Monte Carlo methods. *PLoS Computational Biology*, 2(2), 2006.

[76] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. *SIGMOD Record*, 29(2):427–438, 2000.

[77] S. Raychaudhuri, J. M. Stuart, and R. B. Altman. Principal components analysis to summarize microarray experiments: application to sporulation time series. *Pacific Symposium on Biocomputing*, pages 452–463, 2000.

[78] 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.

[79] D. B. Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976.

[80] D. B. Rubin. *Multiple imputation for nonresponse in surveys*. Wiley, 1987.

[81] R. Rymon. Method and apparatus for role grouping by shared resource utilization, Sept. 2003. United States Patent Application 20030172161.

[82] P. S. Sarbanes and M. G. 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".

[83] J. Schafer. *Analysis of Incomplete Multivariate Data*. Number 72 in Monographs on Statistics and Applied Probability. Chapman Hall/CRC, 1997.

[84] J. Schafer and J. Graham. Missing data: Our view of the state of the art. *Psychological Methods*, 7(2):147–177, 2002.

[85] T. Schank and D. Wagner. Approximating clustering coefficient and transitivity. *Journal of Graph Algorithms and Applications*, 9, 2005.

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

[87] D. Shin, G.-J. Ahn, S. Cho, and S. 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.

[88] I. Shmulevich and W. Zhang. Binary analysis and optimization-based normalization of gene expression data. *Bioinformatics*, 18(4):555–565, 2002.

[89] D. 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.

[90] O. G. Troyanskaya, M. Cantor, G. Sherlock, P. O. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17(6):520–525, 2001.

[91] J. Tuikkala, L. L. Elo, O. S. Nevalainen, and T. Aittokallio. Missing value imputation improves clustering and interpretation of gene expression microarray data. *BMC Bioinformatics*, 9(202), 2008.

[92] J. Vaidya, V. Atluri, and Q. 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.

[93] J. Vaidya, V. Atluri, Q. Guo, and N. 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.

[94] J. Vaidya, V. Atluri, and J. 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.

[95] S. Wasserman and K. Faust. *Social Network Analysis*, chapter 5, pages 169–219. Cambridge University Press, 1994.

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

[97] D. Williams. *Probability with Martingales*. Cambridge University Press, 1991.

[98] Y. Xiang, R. Jin, D. Fuhry, and F. F. Dragan. Succinct summarization of transactional databases: An overlapped hyperrectangle scheme. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '08*, pages 758–766, 2008.

[99] K. Yamanishi, J.-I. Takeuchi, G. Williams, and P. Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. In *KDD '00: Proceedings of the 6th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 320–324, 2000.

[100] G. Yang. Computational aspects of mining maximal frequent patterns. *Theoretical Computer Science*, 362(1):63–85, 2006. Elsevier Science Publishers Ltd.

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

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

[103] D. Zhang, K. Ramamohanarao, and T. 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.

[104] N. Zhang, M. Ryan, and D. P. Guelev. Synthesising verified access control systems through model checking. *Journal of Computer Security*, 16(1):1–61, 2008. IOS Press.