Taming Role Mining Complexity in RBAC

Alessandro Colantonio\textsuperscript{a,b}, Roberto Di Pietro\textsuperscript{b,c}, Alberto Ocello\textsuperscript{a}, Nino Vincenzo Verde\textsuperscript{b}

\textsuperscript{a}Engiweb Security, Roma, Italy
\textsuperscript{b}Università di Roma Tre, Dipartimento di Matematica, Roma, Italy
\textsuperscript{c}UNESCO Chair in Data Privacy, Tarragona, Spain

Abstract

In this paper we address the problem of reducing the role mining complexity in RBAC systems. 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—the proofs of our results are rooted in graph theory—that this methodology allows role engineers for the elicitation of stable candidate roles, by contextually simplifying the role selection task. Efficient algorithms to implement our strategy are also described. Further, we discuss practical applications of our approach. Finally, we tested our methodology on real data set. Results achieved confirm both the viability of our proposal and the analytical findings.

Key words: Role-Based Access Control, Stable Roles, Clustering Coefficient, Administration Cost Minimization, Role Mining Complexity

1. Introduction

RBAC (Role-Based Access Control \cite{1}) is a widely adopted access control model. According to this model, roles are created for various job functions within the organization. The permissions required to perform certain operations are assigned to specific roles. System users, in turn, are assigned to appropriate roles based on their responsibilities and qualifications. Through role assignments they acquire the permissions to perform particular system functions. By deploying RBAC systems we obtain several benefits such as simplified access control administration, improved organizational productivity, and security policy enforcement. In particular, the use of roles minimizes system administration effort due to the reduced number of relationships required to bind users to permissions \cite{2, 3}. Despite the benefits related to RBAC, many organizations are reluctant to adopt it, since there are still some important issues that need to be addressed. In particular, the model must be customized to capture the needs and functions of the company. For this purpose, the role engineering discipline \cite{4} has been introduced. Various approaches to role engineering have been proposed, which are usually classified as: top-down and bottom-up. The former requires a deep analysis of business processes to identify which access permissions are necessary to carry out specific tasks. The latter seeks to identify de facto roles embedded in existing access control information. Since bottom-up approaches usually resort to data mining techniques, the term role mining is often used as a synonym for bottom-up.

Bottom-up approaches to role engineering have been more attractive to researchers since it can be easily automated \cite{5}. Indeed, companies which plan to go for RBAC are usually in the situation of having several security systems to be unified, each of them providing an access control model different from RBAC \cite{6}. Thus, role mining is the application of data mining techniques to generate roles from the access control information of this collection of systems. Kuhlmann et al. \cite{6} first introduced the term “role mining”, trying to apply existing data mining techniques to elicit roles from existing access data. After that, several algorithms explicitly designed for role engineering were proposed \cite{7, 8, 9, 10, 11, 12, 13, 14, 15}. A recently addressed problem is the analysis of the effort incurred by administrators when managing the set of roles elicited by role mining algorithms. To this aim, \cite{16, 2} introduce the administration cost function. An optimal candidate role-set is a set of roles that correctly describes the existing permissions in such a way its administration cost is minimized. An important observation is that by introducing new users, new permissions, or new user-permission assignments within the access control system, there could be the need to reassess the role-set in use. In particular, roles could be unstable, in the sense that the introduction of few users or few permissions could require a complete re-design of such roles in order to reduce the overall administration cost. Unstable roles are thus difficult to manage as they likely change during their lifecycle. Conversely, a role is stable if it is not greatly affected by the introduction of new
users, new permissions, or new user-permission assignments, namely it still remains optimal according to the given administration cost function. That is why, when dealing with automated role mining algorithms, the stability of elicited candidate roles is a desirable property. Another key problem that typically affects role mining algorithms is how to propose roles that have business meaning. As a matter of fact, organizations are unwilling to deploy roles they cannot understand. Yet, most of the roles discovered by existing role mining algorithms could have no connection to the business practice, and only a small fraction of them are meaningful from a business perspective [2]. Hence, another property that is worth investigating is how to reduce the number of the possible candidate roles required to manage each user-permission assignment, then simplifying the selection of the most meaningful ones for the organization.

To address all the above mentioned issues, this paper 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 mentioned 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 paper is the adoption of the clustering coefficient as a metric to evaluate the role mining complexity. To our knowledge, we are the first in using the clustering coefficient within the RBAC model. 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.

The remainder of this paper is organized as follows: Section 2 sums up the concepts and the definitions used in this paper. In Section 3 the proposed model is detailed, and the methodology to prune unstable assignments is formally introduced. Section 4 shows that pruning unstable assignments also reduces the role engineering complexity, while Section 5 describes the pruning algorithms. Then, the viability of the proposed applications is demonstrated in Section 6 by discussing the results of tests on real data. Finally, Section 7 provides concluding remarks.

### 2. Background and Related Work

#### 2.1. Role Engineering Objectives

Before introducing the required formalism to describe the role engineering problem, we first review some concepts of the ANSI/INCITS RBAC standard [1] needed in the following. To ease the exposition, we do not consider sessions, role hierarchies or separation of duties constraints in this paper. In particular, we are only interested in the following entities:

- **PERMS, USERS, and ROLES** are the set of all access permissions, users, and roles, respectively;
- **UA ⊆ USERS × ROLES**, the set of all role-user relationships;
- **PA ⊆ PERMS × ROLES**, the set of all role-permission relationships.

In addition to RBAC concepts, this paper introduces other entities required to formally describe the proposed approach. In particular, we refer to **UP ⊆ USERS × PERMS** as the set of the existing user-permission relationships to be analyzed. In the rest of the paper, we will use the term assignment as a synonym for a user-permission relationship. Moreover, we need the following definitions:

**Definition 1 (System Configuration).** Given an access control system, we refer to its configuration as the tuple \( ϕ = \langle USERS, PERMS, UP \rangle \), namely an instance of all the sets that characterize the RBAC model.

A system configuration is the user authorization state before migrating to RBAC, or the authorizations derivable from the current RBAC implementation.

**Definition 2 (RBAC State).** An RBAC state is represented by tuple \( ψ = \langle ROLES, UA, PA \rangle \), namely an instance of all the sets that characterize the RBAC model.

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

**Definition 3 (Candidate Role-Set).** Given a system configuration \( ϕ \), a candidate role-set is the RBAC state \( ψ \) that “covers” all possible combinations of permissions possessed by users according to \( ϕ \), namely a set of roles whose union of permissions matches exactly with the permissions possessed by the user. Formally: \( ∀u ∈ USERS, ∃ R ⊆ ROLES : \bigcup_{r ∈ R} p ∈ PERMS | (p,r) ∈ PA \rangle = \{p ∈ PERMS | (u,p) ∈ UP \} \).

**Definition 4 (Cost Function).** Let \( Φ, Ψ \) be the set of all possible system configurations and RBAC states, respectively. The cost function is defined as cost: \( Φ × Ψ \rightarrow \mathbb{R} \). It represents an administration cost estimate for the state \( ψ \) used to obtain the configuration \( ϕ \).

By having introduced the above mentioned concepts, it is now possible to formally define the main objective of role engineering. In particular, given \( UP, PERMS, \) and \( USERS \), we are interested in determining the best setting for \( ROLES, PA \), and \( UA \) to cover the existing user-permission relationships. In this context, the word “best” means that the proposed roles should maximize the advantages offered by adopting RBAC (i.e., to minimize the cost function), thus simplifying business governance, risk mitigation, and regulatory compliance throughout the enterprise. Leveraging the cost metric makes it possible to find candidate role-sets which lead to the lowest possible effort for the administration of the resulting RBAC state. Formally:
Definition 5 (Optimal Candidate Role-Set). Given a configuration $\varphi$, an optimal candidate role-set is the corresponding configuration $\psi$ that simultaneously represents a candidate role-set for $\varphi$ and minimizes the cost function $\text{cost}(\varphi, \psi)$.

The cost function is thoroughly described in \cite{16, 2}. A similar concept is provided by \cite{3}, where the authors utilize 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 paper, we thus introduce the following metric for roles:

Definition 6 (Role Weight). Given a role $r \in \text{ROLES}$, let $P_r$ and $U_r$ be the sets of permissions and users associated to $r$, that is $P_r = \{p \in \text{PERMS} \mid \langle p, r \rangle \in \text{PA} \}$ and $U_r = \{u \in \text{USERS} \mid \langle u, r \rangle \in \text{UA} \}$. We indicate with $w : \text{ROLES} \rightarrow \mathbb{R}$ the weight function of roles, defined as
\[
w(r) = c_u |U_r| \oplus c_p |P_r|,
\]
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:

Definition 7 (Role Stability). Let $r \in \text{ROLES}$ 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 8 (Assignment Stability). Let the pair $\langle u, p \rangle \in \text{UP}$ be a given assignment, and $t \in \mathbb{R}$ be a real number that we refer to as a “threshold”. Let $R_{(u,p)}$ be the set of roles that contains the assignment $\langle u, p \rangle$, namely $R_{(u,p)} = \{ r \in \text{ROLES} \mid \langle u, r \rangle \in \text{UA}, \langle p, r \rangle \in \text{PA} \}$, and let $w$ be the role weight function. We say that $\langle u, p \rangle$ is stable with respect to $t$ if it belongs at least one stable role, namely $\exists r \in R_{(u,p)} : w(r) > t$. Otherwise, the assignment is unstable, that is $\forall r \in R_{(u,p)} : w(r) \leq t$.

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 1 shows an example of assignment addition in a context where assignments that belong to roles with different weights are present. In particular, Figure 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 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

---

\(^1\)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 $f : V \rightarrow W$ is a function between two vector spaces over a field $F$, then $f$ is said to be homogeneous of degree $k$ if $f(\alpha v) = \alpha^k f(v)$ for all nonzero $\alpha \in F$ and $v \in V$. 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 6.3.
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.

2.2. Graph Theory

We now summarize the relevant graph-related concepts that are required in the rest of the paper. In particular, a graph $G$ is an ordered pair $G = (V, E)$, 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 $N(v) = \{v' \in V | \langle v, v' \rangle \in E\}$. 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 $G = (V, E)$ 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 $G = (V_1 \cup V_2, E)$ 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 of $S$ of 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 paper 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 clique, Zaki and Ogihara [17] 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 [13, 8, 3, 9, 10].

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 NP-complete problems, like binary matrices factorization [18, 9] and tiling database [19] to cite a few. Several role mining approaches leverage these concepts [10, 20, 21, 14, 9].

2.3. Clustering Coefficient

Another mathematical tool used is this paper is the clustering coefficient. It was first introduced by Watts and Strogatz [22] in the social network field, to measure the cliquishness of a typical neighborhood. Given $G = (V, E)$, we indicate with $\delta(v)$ the number of triangles of $v$, formally:

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

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:

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

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

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

where

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

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 4 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.

3. Problem Modeling

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 [10]. 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.
3.1. Role Engineering and Biclique Cover

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

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

where two vertices $u \in \text{USERS}$ and $p \in \text{PERMS}$ are connected by an edge if the user $u$ is granted permission $p$, namely $(u, p) \in \text{UP}$. A biclique cover of the graph $G$ univocally identifies a candidate role-set $\psi = \langle \text{ROLES}, \text{UA}, \text{PA} \rangle$ for the configuration $\varphi$. Indeed, every biclique identifies a role, and the vertices of the biclique identify the users and the permissions assigned to this role [10, 20, 21]. Thus, finding the optimal role-set is equivalent to identifying the biclique cover such that the corresponding roles are optimal according to Definition 5.

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 $B: \text{UP} \rightarrow 2^{\text{UP}}$ that indicates all edges in UP which induces a biclique together with the given edge, namely:

$$B(\langle u, p \rangle) = \{\langle u', p' \rangle \in \text{UP} | \langle u, p' \rangle, \langle u', p \rangle \in \text{UP} \land \langle u, p \rangle \neq \langle u', p' \rangle\}. \quad (7)$$

Note that two edges $\omega_1 = \langle u_1, p_1 \rangle$ and $\omega_2 = \langle u_2, p_2 \rangle$ of $\text{UP}$ 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 $\langle u_1, p_2 \rangle, \langle u_2, p_1 \rangle \in \text{UP}$ exist. Moreover, given $\omega_1, \omega_2 \in \text{UP}$, 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:

$$G' = \langle V', E' \rangle = \langle \text{UP}, \{\langle \omega_1, \omega_2 \rangle \in \text{UP} \times \text{UP} | \omega_1 \in B(\omega_2)\} \rangle \quad (8)$$

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

To clarify this concept, Figure 2 show a simple example, where $\text{USERS} = \{A, B, C, D\}$, $\text{PERMS} = \{1, 2, 3, 4\}$, and $\text{UP} = \{\langle A, 1 \rangle, \langle A, 2 \rangle, \langle A, 3 \rangle, \langle B, 1 \rangle, \langle B, 2 \rangle, \langle B, 3 \rangle, \langle C, 3 \rangle, \langle C, 4 \rangle, \langle D, 4 \rangle\}$. In the figure, the assignment $\langle B, 2 \rangle$ represents an edge in the bipartite graph (Figure 2(a)) and a vertex in the unipartite graph (Figure 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 7; for example, $\langle B, 3 \rangle$ share the same user of $\langle B, 2 \rangle$, while $\langle A, 1 \rangle$ induce a biclique with $\langle B, 2 \rangle$ since the assignments $\langle B, 1 \rangle$ and $\langle A, 2 \rangle$ exist.

It is known that finding a clique partition of a graph is equivalent to finding a coloring of its complement [10, 20, 21]. 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 $\omega \in \text{UP}$, we indicate with $\overline{B}(\omega)$ the assignments that do not induce a biclique together with $\omega$, namely

$$\overline{B}(\omega) = (\text{UP} \setminus \{\omega\}) \setminus B(\omega). \quad (9)$$

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

$$\overline{G'} = \langle \overline{V'}, \overline{E'} \rangle = \langle \text{UP}, \{\langle \omega_1, \omega_2 \rangle \in \text{UP} \times \text{UP} | \omega_1 \in \overline{B}(\omega_2)\} \rangle \quad (10)$$

Any coloring of the graph $\overline{G'}$ identifies a candidate role-set of the given system configuration $\varphi = \langle \text{USERS}, \text{PERMS}, \text{UP} \rangle$, from which we have generated $G$. Thus, finding a proper coloring for $\overline{G'}$ means finding a candidate role-set that covers all possible combinations of permissions possessed by users according to $\varphi$; 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 3. In particular, Figure 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 3(b) represents the same information in the unipartite view in terms of clique partition. Figure 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.

3.2. 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 $G'$ 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 $G'$ with a limited degree. Since the third step corresponds to coloring $G'$, 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 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 $B$ be the biclique across $G_i$ and $G_k$, with $i \neq k$, and let $B_i$ and $B_k$ be the sets of vertices of $B$ belonging respectively to $G_i$ and $G_k$. From the biclique definition, it follows that edges between the two vertex sets of $B_i$ and $B_k$ must exist. But it is a contradiction, since $G_i$ and $G_k$ are two disconnected components, hence edges between their vertices cannot exist. □

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 $\phi = \langle\text{USERS, PERMS, UP}\rangle$. This lemma has an important implication:

**Theorem 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 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 $G'$ when the bipartite graph $G$ is disconnected.

3.3. Unstable Assignment Identification

In our model, the role mining problem corresponds to finding a proper coloring for the graph $G'$. 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 $G'$ by highlighting how this information can affect the assignment stability and, as a consequence, the administration effort.

According to Equation 10 the degree of the graph $G'$ can be expressed as:

$$\Delta(G') = \max_{\omega \in \text{UP}} |B(\omega)|.$$ (11)

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 $G'$ is a user-permission relationship in the set $\text{UP}$. An edge in $G'$ between two vertices $\omega_1$ and $\omega_2$ exists if the corresponding user-permission relationships cannot be in the same role, due to the fact that the user in $\omega_1$ does not have the permission in $\omega_2$, or the user in $\omega_2$ does not have the permission in $\omega_1$. Consequently, a vertex of $G'$ that has a high degree means that this

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figs/figure3.png}
\caption{Relationship among biclique cover, clique partition, and vertex coloring.}
\end{figure}
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, 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 \( \omega \). 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 2.** Let \( UP \) be the set of edges of the bipartite graph \( G \). Then:

\[
\forall \omega \in UP, \ |B(\omega)| > d \implies \Delta(G') \leq |UP| - d
\]

**Proof.** Since \( \forall \omega \in UP, \ |B(\omega)| > d \), according to Equation 9 the following holds: \( \forall \omega \in UP, \ |B(\omega)| < |UP| - d \). The proof follows from \( \Delta(G') = \max_{\omega \in UP} |B(\omega)| \). \( \square \)

Thus, given a suitable value for \( d \), the idea is to prune the graph \( G' \) 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 2.** The pruning operation based on removing from \( G' \) vertices \( \omega \) 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 \( \omega \) be the assignment we would like to prune since \( |B(\omega)| \leq d \). The corresponding vertex in \( G' \) 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 \( \omega \) belongs to. Let \( r \) be such a role. According to Definition 6, 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. \( \square \)

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 \( G' \), it allows for an efficient application of this class of algorithms. Without our pruning operation, the degree of the graph \( G' \) 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 \( \Delta(G') \) is very high. But, for Theorem 1, we can split the problem by considering the different components distinctly, and then join the results of each component.

### 3.4. Applications to Role Mining

Having a bound for \( \Delta(G') \) makes it possible to use many known algorithms to color a graph in addition to the classical role mining algorithms mentioned in Section 2. Indeed, finding a coloring for \( G' \) 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 \( \Delta \) can be colored with \( \Delta + 1 \) colors by choosing an arbitrary ordering of the vertices, and then coloring them one at a time by labeling each vertex with a color not already used by any of its neighbors. In other words, we can find \( \Delta(G') + 1 \) 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 \( G' \). Due to Theorem 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 \( \Delta(G') + 1 \) does not represent the minimal number of roles, but it is only an upper bound for the chromatic number of \( G' \).

**Randomized Approach.** By applying the randomized approach described in [23] to our case, it is possible to generate \( \Delta(G')/k \) roles that cover all the user-assignments which survive after the pruning, where \( k = O(\log \Delta(G')) \). 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 \( \Delta \)-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 \( G' \), thus obtaining a \( \Delta \)-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 \( G' \) 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 \( Pr(B) = (Pr(A))^3 \). 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 \( G' \), they compose a triangle. In other words, the probability to have a triangle in \( G' \) 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 prob-
ability \( p \) will be very high, and \( Pr(B) \) will be close to 0. However,
suppose that \( G' \) 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 \( G' \) cor-
responds 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.

### 4. Complexity of Role Engineering

In this section we will discuss about the application of the clustering coefficient (see Section 2.3) in RBAC. In particu-
lar, 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 3.3
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 clus-
tering 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:

\[
\text{assignments with unstable neighbors} \implies \text{low clustering coefficient} \implies \text{complex role engineering task.}
\]

#### 4.1. Clustering Coefficient in \( G' \)

Let \( G' \) be the unipartite graph derived from user-permission
assignments \( UP \) according to Equation 8. Consequent-
ly, Equation 3 becomes:

\[
\tau(\omega) = \left| \{(\omega_1, \omega_2) \in UP \times UP | \omega_1, \omega_2 \in B(\omega), \omega_1 \neq \omega_2\} \right|,
\]

namely \( \tau(\omega) \) is the set of all possible pairs of elements in \( UP \)
that both induce a biclique with \( \omega \). Further, Equation 2 be-
comes:

\[
\delta(\omega) = \left| \{(\omega_1, \omega_2) \in UP \times UP | \omega_1, \omega_2 \in B(\omega), \omega_1 \in B(\omega_2)\} \right|,
\]

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

The clustering coefficient index (Equation 4) of the graph
\( G' \) derived from an access control system configuration is thus
deefined as:

\[
C(G') = \frac{1}{|UP|} \sum_{\omega \in UP} c(\omega),
\]

where \( c(\omega) \) is the local clustering coefficient of \( \omega \) (see Equation 5) defined as:

\[
c(\omega) = \begin{cases} 
\delta(\omega), & \tau(\omega) \neq 0; \\
1, & \text{otherwise}. 
\end{cases}
\]

The value of \( c(\omega) \) quantifies how close \( \omega \) and its neighbors
are to being a biclique. In our model, this corresponds to mea-
sure how close \( \omega \) 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 bi-
cliques. That is, the easiness of identifying a candidate role set
for the analyzed data—see below what “easy” means. Notice
that, according to Equation 15, when a user-permission assign-
ment 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 14 and 15 only re-
quire \( UP \) and \( B(\cdot) \) to be provided, by neglecting whether we are
considering the bipartite or the unipartite graph. Thus, in the re-
mainder of this paper 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. Given a role \( \tau \in ROLES \),
let \( P_\tau = \{ p \in PERMS | \{ p, \tau \} \in PA \} \) be the set of its assigned
permissions, and \( U_\tau = \{ u \in USERS | \{ u, \tau \} \in UA \} \) be the set of
its assigned users. If the following equation holds

\[
\exists U \subseteq USERS, \exists P \subseteq PERMS : U \times P \subseteq UP, U \times P_{\tau} \subset U \times P,
\]

then the role \( \tau \) 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 \( \forall u \in U, \forall p \in P : \{ u, p \} \in UP \),
and is not a subset of the vertices of any larger complete
subgraph, that is \( \exists U' \subset USERS \) and \( \exists P' \subset PERMS \) such that
\( \forall u \in U', \forall p \in P' : \{ u, p \} \in UP \) and contextually \( U' \subset U \) and
\( P' \subset P \).

Informally, a role delineated by a maximal biclique is “repre-
sentative” for all the possible subset of permissions shared
by a given set of users. The key observation behind a maxi-
mal 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 adminis-
tration effort of the RBAC system by reducing the number of
required role-user assignments. The properties of roles repre-
sented by maximal bicliques are further detailed in [13], which
also proposes an efficient algorithm to identify all possible roles
associated to maximal biclique within \( G \).

The following theorem relates the clustering coefficient in-
dex to the complexity of the role mining problem in terms of
number of maximal bicliques:
Theorem 3. Let \( M \) be the set of all possible maximal bicliques that can be identified in \( G \). Given a user-permission assignment \( \omega \in UP \), let \( M(\omega) \subseteq M \) be the set of all possible maximal bicliques the given user-permission assignment belongs to. Then, the following holds:

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

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 \( P_r = \{ p \in PERMS \mid (p, r) \in PA \} \).

First, we analyze the case \( c(\omega) = 1 \). Let \( \Gamma \) be a role made up of the users and permissions involved by the assignments \( \omega \) and \( B(\omega) \), formally \( \Gamma_r = \{ u \in USERS \mid \exists p \in PERMS, \langle u, p \rangle \in B(\omega) \} \) and \( \Gamma_r = \{ p \in PERMS \mid \exists u \in USERS, \langle u, p \rangle \in B(\omega) \} \). We now demonstrate that at least one maximal biclique exists and it is represented by \( \Gamma \). According to Equation 7, \( \forall (u_1, p_1), (u_2, p_2) \in B(\omega) \) \( \propto \exists (u_1, p_2), (u_2, p_1) \in UP, \) namely both users \( u_1, u_2 \) have permissions \( p_1, p_2 \) granted. According to Equation 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 \( B(\omega) \cup \omega = U_r \times P_r \). We now prove by contradiction that \( \Gamma \) is a maximal biclique. If \( \Gamma \) were not a maximal biclique, two sets \( U \subseteq USERS \) and \( P \subseteq PERMS \) would exist such that \( U_r \times P_r \subseteq U \times P \subseteq UP \). Let \( \omega = (u. p) \). Yet, for each \( \langle u', p' \rangle \in (U \times P) \backslash (U \times P_r) \) 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 7, \( \langle u', p' \rangle \in B(\omega) \), meaning that \( (U \times P) \backslash (U \times P_r) = 0 \). Therefore, \( \Gamma \) is a maximal biclique. We now demonstrate that another maximal biclique that contains \( \omega \) cannot exist. Indeed, if \( \Psi \) is a maximal biclique containing \( \omega \) (i.e., \( \omega \in U_r \times P_r \)), for all \( \omega' \in U_r \times P_r \) \( \setminus \omega \) it can be shown that \( \omega' \in B(\omega) \). Hence, \( \Psi = \Psi \). Finally, having only one maximal biclique that contains \( \omega \) implies that \( c(\omega) = 1 \). Let \( \Psi \) be such a maximal biclique. Since it is the only maximal biclique, for each pair \( \omega_1, \omega_2 \in U_r \times P_r \) 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 \( \omega \) combined with each element of \( B(\omega) \). Let \( \omega = (u, p) \). First, observe that such maximal bicliques are distinct since \( c(\omega) = 0 \implies \delta(\omega) = 0 \). We want to show that for each \( \langle u_1, p_1 \rangle \in B((u, p)) \), the role \( \Psi \) is such that \( U_r = \{ u_1 \} \) and \( P_r = \{ p_1 \} \) is a maximal biclique. We now prove by contradiction that \( \Psi \) is a maximal biclique. If \( \Psi \) were not a maximal biclique, two sets \( U \subseteq USERS \) and \( P \subseteq PERMS \) would exist such that \( \{u_1, u_2\} \times \{p_1, p_2\} \subseteq U \times P \subseteq UP \). Let \( \langle u', p' \rangle \in (U \times P) \backslash \{u_1, u_2\} \times \{p_1, p_2\} \). It can be easily shown that \( \langle u', p' \rangle \in B((u, p)) \), thus \( \delta(\omega) \neq 0 \). But, according to equation 15, this means that \( c(\omega) > 0 \), which is a contradiction. Moreover, more than \( |B(\omega)| \) distinct maximal bicliques that contain \( \omega \) cannot exist. Indeed, let \( n \in \mathbb{N} : n > |B(\omega)| \) be the number of the distinct maximal bicliques that contain \( \omega \). Let \( \Psi \) indicate the \( n^\text{th} \) maximal biclique, and let \( \omega_1 \in (U_r \times P_r) \backslash \{\omega\} \). 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 \( \Psi \) indicate the \( n^\text{th} \) maximal biclique, and let \( \omega_i \in (U_r \times P_r) \backslash \{\omega\} \). Since the roles are distinct, \( \forall i, j \in 1 \ldots |B(\omega)| : i \neq j \) we have that \( \omega_i \notin B(\omega) \). Thus, \( \delta(\omega) = 0 \), and, according to equation 15, \( c(\omega) = 0 \).

Finally, by excluding the previous two cases we merely have that \( c(\omega) \in (0,1) \implies 1 < |M(\omega)| < |B(\omega)| \). □

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

4.2. Clustering Coefficient and Vertex Degree

In the previous section we demonstrated that the local clustering coefficient of a given assignment expresses the ambiguity in selecting the best maximal biclique to cover it when finding the best biclique cover. Hereafter, we show that the local clustering coefficient value and the number of assignments that induce a biclique are bound. In particular, we prove that the presence of unstable assignments decreases the maximum local clustering value allowed for stable assignments. Therefore, keeping unstable assignments within the data to analyze hinders the role engineering process by increasing the ambiguity in selecting the best roles to cover stable assignments.

Theorem 4. Let \( \omega \in UP \) be a user-permission assignment such that \( |B(\omega)| > 1 \). Then, the following holds:

\[
 c(\omega) \leq \frac{\text{avg } |B(\omega')| - 1}{|B(\omega)| - 1}. 
\]

Proof. According to its definition, the local clustering coefficient of a vertex in \( G^2 \) is the ratio between its triangles (Equation 13) and its triples (Equation 12). All the neighbors of a vertex \( \omega \) are represented by \( B(\omega) \). Thus, we have

\[
 \tau(\omega) = \left( \frac{|B(\omega)|}{2} \right)^2 = \frac{|B(\omega)||B(\omega)| - 1}{2}. 
\]

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

\[
 \delta(\omega) \leq \frac{1}{2} \sum_{\omega \in B(\omega)} |B(\omega')| - 1. 
\]
where ‘1/2’ is required to take into account that each triangle is considered twice. By combining the previous equations, we obtain:

\[
c(\omega) = \frac{\delta(\omega)}{\tau(\omega)} \leq \frac{1}{2} \sum_{\omega \in B(\omega)} |B(\omega')| - 1 \leq \frac{\text{avg}_{\omega \in B(\omega)} |B(\omega')| - 1}{|B(\omega)| - 1},
\]

completing the proof. \(\square\)

Notice that \(c(\omega) = 1\) means that all the neighbors of \(\omega\) in \(G\) have, among their neighbors, all the neighbors of \(\omega\). Thus, the right side of the inequality in Equation 17 is equal to or greater than 1. Similarly, \(c(\omega) = 0\) means that each pair of neighbors of \(\omega\) are not neighbors among them. Thus, the right side of the inequality in Equation 17 is equal to or greater than 0.

Finally, let us assume that all the neighbors of \(\omega\) have a degree that is lower than the degree of \(\omega\), 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 [24]. The problem is even worse for roles which cover many user-permission assignments, since they are more vulnerable to noise [25].

In Section 6 we will show through experiments on real data that the clustering coefficient increases when pruning unstable assignments.

5. 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 3. 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.

5.1. 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 \(\omega'\)”. This schema is perfectly equivalent to directly associating the value \(|B(\omega)|\) to \(\omega\), without increasing the complexity. Yet, it can be easily randomized, as we will see in the next section.

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

**Lemma 3.** Given an assignment \(\omega = \langle u, p \rangle \in UP\), let \(U_\omega = \{u' \in USERS\ | \langle u', p \rangle \in UP\}\) be the set of all users possessing the corresponding permission, and \(P_\omega = \{p' \in PERMS\ | \langle u, p' \rangle \in UP\}\) 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 \(u' \not\in U_\omega\) and/or \(p' \not\in P_\omega\). According to Equation 7, \(\langle u', p' \rangle \in B(\omega)\) 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 \(P_\omega, U_\omega\), 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. \(\square\)

Lemma 3 is used to define the procedure \(\text{Neighbors}\) in Algorithm 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 \(\text{Neighbors}\) 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 \(O(|UP|)\).

\(\text{CountNeighbors}\) 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 \(\text{count}[]\). 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 \(\text{CountNeighbors}\) is \(O(|UP|^2)\). Line 7 gives the resulting neighbor-counts. All the values are normalized by
It is reasonable to give an estimate sorted at the end of the procedure. This takes \( O(m \log |UP|) \), 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 \( O(m \log |UP|) \).

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 shows a practical application of this methodology in a real case.

5.2. 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 1, which compute in \( O(k|UP|) \) an approximated neighbor-counter value for the assignments, where \( k \) is a parameter that can be arbitrarily chosen. Moreover, we will show how to select the best value for \( k \), and, when \( k \ll |UP| \), it achieves good results in a significantly shorter time. Notice that the pruning procedure can still be performed in \( O(\log |UP|) \) only if the 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 2 describes RandomizedCountNeighbors as an alternative approach for the procedure CountNeighbors of Algorithm 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 5.** Let \( \omega = (u, p) \) be an assignment, and let \( \tilde{d}_k(\omega) \) be the output of the procedure RandomizedCountNeighbors described in Algorithm 2 for such an assignment. Then

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

where \( \varepsilon \) is the threshold.

Proof. We will use the Hoefting inequality [26] to prove this theorem. It says that if \( X_1, \ldots, X_n \) are independent random variables such that \( 0 \leq X_i \leq 1 \), then

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

Hence, Equation 18 can be rewritten as

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

where \( \varepsilon = \frac{t}{k} \). Notice that the value \( \frac{1}{n} \sum_{i=1}^n X_i \) is exactly the output of Algorithm 2. Hence, in order to prove that the algorithm gives an approximation of \( \frac{|B(\omega)|}{|UP|} \), we have to prove that \( \mathbb{E}\left[ \frac{1}{k} \sum_{i=1}^k X_i \right] \) is equal to \( \frac{|B(\omega)|}{|UP|} \). Because of the linearity of the expectation, the following equation holds:

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

Since the assignment \( \omega_i \) is picked uniformly at random, the probability to choose it is 1/|UP|. Thus,

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

completing the proof. \( \square \)

For practical applications of Algorithm 2, it is possible to calculate the number of samples needed to obtain an expected
error less than $\varepsilon$ with a probability greater than $p$. The following equation directly derives from Theorem 5:

$$k > -\frac{1}{2e^3} \ln \left( \frac{1 - p}{2} \right).$$

(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. Experimental Results

To prove the viability of our approach, we applied it to several real-world datasets at our disposal. In the following, we first report the application of our model to the access control configuration related to users of an organization unit of a large company. Then, by using the previous dataset, we highlight the effect of the pruning operation on the role mining complexity. Finally, we show how it is possible to compute the optimal threshold to use with our pruning strategy. In all the tests we used the approximated version of our pruning algorithm (with $k = 1000$), and normalized values for the pruning threshold, as detailed in Section 5.

6.1. Pruning Example

Figure 4 shows an example of our strategy when applied to a real dataset. Figure 4(a) represents the bipartite graph $G$ built from the access control configuration relative to users of an Organization Unit (OU) of a large company. The OU analyzed counts 7 users (nodes on the left) and 39 permissions (nodes on the right), with a total of 71 user-permission assignments. We have chosen an OU with few users and permissions to ease graph representation. According to a pruning threshold equal to 0.39, stable assignments are depicted with thicker edges, while unstable assignments with thinner edges. Figure 4(b) depicts the unipartite graph $G'$, built according to Equation 8. The user-permission assignments of $G$ correspond to the vertices of $G'$, and two vertices are connected by an edge if they induce a biclique. Dashed edges indicate that one of the two endpoints will be pruned. Figure 4(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.

6.2. Effects of the Pruning on the Mining Complexity

Theorem 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 5 graphically shows this behavior. The dataset analyzed is the same that has been used in Section 6.1. 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 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 10 and 50 have a clustering coefficient equal to 1. Note that, according to Theorem 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.

6.3. 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 [27]. One possible way to do this is combining different functions in a weighted sum, with the following general formulation:

$$\sum_{f \in F} \alpha_i f_i.$$  

(22)

$F$ is the set of the functions to optimize, and $\alpha_i$ 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

12
Figure 4: Our model applied to a real Organizational Unit

Figure 5: Effect of the pruning on the local clustering coefficient
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 \(|U_r| \times |P_r|\).

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

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

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

In Figure 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). Figures 6(a) and 6(b) represent the aggregated functions for these two organization unit. Figures 6(c) and 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(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(b)). In this case, the pruned assignments are 1,367, the average weight increased from 120 to 151, the number of maximal bicliques has been decreased from 3,000 to 266, while the clustering coefficient has been increased from 0.66 to 0.78. At first sight, it seems that we pruned too much assignments, but these results depend both on the dataset we are analyzing and on the targets that role engineers want to reach. Indeed, this dataset has a higher complexity with respect to the first one, and we provided high weights for Clustering Coefficient and Maximal Bicliques. If we gave less relevance to these two parameters, a lower threshold would have been a good trade off. In that case, the pruned assignments would have been less than 1,367, and the average weight would have been higher than the original one. In general, the role engineers mission is to establish the weights of the multi-objective aggregated function in such a way to get as close as possible to the target that they want to reach.

7. Final Remarks

In this paper 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.

Acknowledgements

We would like to thank the anonymous reviewers for their comments that helped to improve the presentation of this paper.

References

Figure 6: Finding the best threshold. Two real datasets similar in size, that only differ in clustering coefficient, are analyzed. 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. In both cases, the best threshold has been found by minimizing the multi-objective function \( \text{Index} = -\text{Clustering Coefficient} + 0.3 \times \text{Pruned Assignments}^\text{max} + 0.8 \times \text{Maximal Bicliques}^\text{max} - 0.8 \times \text{Average Weight}^\text{max} \). By using these weights, 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.