Computers & Security 162 (2026) 104808

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Full Length Article

A bag of words model for efficient discovery of roles in access control systems

Carlo Blundo [a], Stelvio Cimato [b,*]

[a] Dipartimento di Scienze Aziendali - Management & Innovation Systems, Universit'a degli Studi di Salerno, Italy
[b] Dipartimento di Informatica, Universit'a degli studi di Milano, Italy

ARTICLE INFO

Keywords:
Role mining
RBAC
Constrained role mining

ABSTRACT

The popularity of the Role-based Access Control (RBAC) model is determined by its flexibility and its adaptability in different contexts, easing the enforcement and the management of security policy. In some cases, different kinds of (cardinality) constraints are considered to adjust and adapt roles and their assignment to best represent the organization's security policy.

However, the process of role mining, whether based on an organizational scenario or on existing permission assignments, is a hard task, since the problem shows NP-hard computational complexity and in case of frequent policy updates, the dynamic adaptation of the roles can be challenging. Then, the only possibility of producing an RBAC model compliant with the security policy is to resort to heuristics, which may return an approximation of the optimal solution.

In this paper, we propose an innovative approach to explore the space of the solution based on the bag of word value, which is commonly deployed in the field of document representation and knowledge extraction. We propose different heuristics and validate our approach reporting the results of the application to standard datasets, and providing an evaluation under different metrics and indicators. We show that our technique returns improved results and provides an alternative way to produce valid solutions for constrained RBAC.

1. Introduction

RBAC models have become the standard method for complex organizations to assign permissions to users in order to control access to restricted resources. RBAC provides a high-level representation of access control policies, unlike access control lists, where permissions are managed and assigned to each user individually. Users are assigned to roles, and roles define the resources each user can access. In large companies, defining roles can be challenging and role engineering techniques, first introduced in Coyne (1995), are used to intelligently organize functions, which is crucial to implementing RBAC effectively. More precisely, role mining techniques are used to automatically derive roles either from the organizational structure of a company or by discovering patterns in existing user-permission assignments. In this paper, we focus on the second aspect. Hence, we consider role mining to be a data-driven process that focuses on finding patterns in existing permissions. The goal is to discover an optimal set of roles and assignments that best represent the current user-permission allocation, often with the objective of minimizing the number of roles or other metrics of *complexity*. The *meaning* of the mined roles and their alignment with the actual job functions are post-processing or validation steps. The advantages of adopting RBAC models lie in the reduction of costs they allow and in the simplification of the administration tasks, especially when changes in the structure are frequent and employees are engaged in new activities or modify their duties. This approach differs from scenarios where access control must be directly integrated with security measures, such as in data outsourcing.

To fit the organization needs, often some constraints are introduced during the mining process to limit, for example, the number of permissions that can be included in a role or the number of roles that can be assigned to a user. These *cardinality* constraints are integrated into the definition of the security policy of an organization and have an impact on the resulting role-set (John et al., 2012; Kumar et al., 2010; Blundo and Cimato, 2013; Blundo et al., 2020a). In some cases, the application of multiple cardinality constraints has been investigated (Ma et al., 2015; Li et al., 2015; Harika et al., 2015; Blundo et al., 2017). Together with cardinality constraints, other kinds of limitation have been considered for the definition of the set of roles. For example, *separation of duty constraints (SoD)* or *statically mutually exclusive roles (SMER)*, where conflicting roles from a given set cannot be assigned to a single user, can

* Corresponding author.
E-mail address: stelvio.cimato@unimi.it (S. Cimato).

https://doi.org/10.1016/j.cose.2025.104808
Received 8 May 2025; Received in revised form 1 November 2025; Accepted 12 December 2025
Available online 23 December 2025

be defined to model situations where uncontrolled assignment of tasks can lead to conflict of interests.

In any case, both the basic Role Mining Problem, that is, finding the minimum number of roles consistent with the starting situation of the organization, and some of the constrained variants, that is, defining some limitations on the number of roles/permissions to be defined/assigned to each user, have been proven to be NP-hard (Blundo et al., 2020a). Research has been oriented towards the definition of efficient heuristics enabling the mining of the role-set nearest to the optimal solution. In many cases, results coming from other related research fields, such as boolean matrix decomposition (Lu et al., 2008a), graph theory (Ene et al., 2008; Zhang et al., 2007), tiling problem for databases (Vaidya et al., 2010a), have been re-applied to the (constrained) role mining problems. Indeed, in many contexts, user-permission, user-role, and role-permission assignment can be easily represented using binary matrices, and different mathematical reformulations can be used to process such matrices.

The computational complexity of role mining is not the only obstacle in operational settings. Real-world deployments face the additional challenge of organizational dynamism: User tasks, responsibilities, and business processes change over time, which in turn alter the permissions that users need. These frequent changes require continuous policy adjustments, making it essential to design methods that address both the combinatorial nature of the problem and its evolving context (Anderer et al., 2023, 2025; Trnecka and Trneckova, 2020; Kang et al., 2023; Rao et al., 2021).

In this paper, we propose applying the very famous bag-of-words model, BoW for short, defined for text document representation, to the role mining scenario. BoW is a very simple and flexible way of extracting features from text, so that they become easily and automatically processable, such as in the case of machine learning algorithms. With this model, information about the structure and order of words in a text is discarded, while the frequency of the words contained in a given vocabulary is measured.

In our context, the basic idea is to map roles, that are collections of permissions, to documents and permissions to words, so that indexes and metrics defined for text processing can be easily reused for the role mining problem. In particular, we consider that the Inverse Document Frequency value, or IDF for short, usually applied to determine which words in a given document contain useful information, can be used to distinguish the most important permissions and roles in an RBAC model, so that roles can be formed appropriately. We embed this criterion in one framework, modifying the behavior of the resulting heuristic, and experimenting with different variants according to the possible choices that can be made in the selection of users and permissions when determining the next role-set candidate. We define the heuristic for the *Permission-Usage Cardinality Constraint* (PUCC) case, where the constraint limits the maximum number of permissions that can be included in each role.

We analyze its behavior considering publicly available benchmark datasets (some from HP lab Ene et al., 2008, others based on data released for the Kaggle competition `Amazon Employee Access Challenge` Hamner et al., 2013) and evaluate its performance comparing the results obtained with those obtained by previously proposed heuristics, showing the effectiveness of our technique. We evaluate the use of *IDF*-based strategies also in multi-constraint scenarios, considering two cases: - the simultaneous enforcement of the Role-Usage Cardinality Constraint (which limits the number of roles assigned to each user) and the Permission-Distribution Cardinality Constraint (which limits the number of roles a single permission can appear in), we denote the resulting procedure as “EURPDC Heuristics”, the combined enforcement of the Role-Usage Cardinality and Permission-Usage Cardinality constraints, we denote as “PRUCC Heuristic”. The data underlying this article are available on GitHub, at https://github.com/carblu/rm-idf.

Contribution. In this paper, we consider for the first time the application of the BoW model to the role mining problem and report some basic results involving:

- Frame the RBAC problem as a document classification problem, showing how some of the indexes and metrics usually devised for the analysis and automatic processing of textual documents can be redefined for the constrained RBAC problem.
- Define a framework in which the heuristics adopt the *IDF* value as the basic criterion to select the users and the permissions to be included in the candidate role for constrained RBAC settings. We show in detail the PUCC case and the extension of the framework to the multi-constraint scenarios, discussing in detail two cases.
- Provide a complete set of experimentation obtained after applying the heuristic to benchmark datasets, taking into account different metrics and different indicators and comparing the results with previously available heuristics.

The paper is organized as follows: in Section 2 we provide the formal definitions of the constrained role mining problems and recall that these problems are all NP-hard; in Section 3, we review the bag of words model and its application to role mining; in Section 4 we describe our heuristics for the constrained role mining problems based on the IDF value, focusing on a single constraint in Section 4.1, where we analyze the Permission-Usage Cardinality Constraint (PUCC) scenario, and multiple constraints in Section 4.2 for the combination of Role-Usage Cardinality Constraint (RUP) and Permission-Distribution Cardinality Constraint (PDP), and combination of RUP and PUCC in Section 4.3; we evaluate all the presented heuristics in Section 5, considering both real world datasets and synthetically generated data in Section 5.1.1; finally we discuss related works in Section 6, and draw some conclusions for our work in Section 7.

2. Role mining problem

In this section, we briefly recall the basic definitions for the RBAC model and discuss the computational complexity of the Role Mining problem and of some of its constrained variants, while a more complete description can be found in Blundo et al. (2020a). The notation we use is based on the NIST standard for *Core Role-Based Access Control* (Core RBAC, or RBAC 0), see Sandhu et al. (2000) and Ferraiolo et al. (2001). We denote by $\mathcal{U} = \{u_1, \dots, u_n\}$ the set of users, $\mathcal{P} = \{p_1, \dots, p_m\}$ the set of permissions, and $\mathcal{R} = \{r_1, \dots, r_k\}$ the set of roles. The following assignment relations are defined:

- $\mathcal{UA} \subseteq \mathcal{U} \times \mathcal{R}$ is a many-to-many mapping *user-to-role* assignment relation.
- $\mathcal{PA} \subseteq \mathcal{R} \times \mathcal{P}$ is a many-to-many mapping *role-to-permission* assignment relation.
- $\mathcal{UPA} \subseteq \mathcal{U} \times \mathcal{P}$ is a many-to-many mapping *user-to-permission* assignment relation.

The assignment relations can be represented by binary matrices where `UA` is the binary matrix representation for $\mathcal{UA}$'s reporting the user to role assignment, such that $\texttt{UA}[i][j] = 1$ if and only if $(u_i, r_j) \in \mathcal{UA}$ (the same holds for matrices `PA`, and `UPA`).

The *role mining problem* (see, for instance, Vaidya et al., 2007, Ene et al., 2008, and Frank et al., 2008) involves the *factorization* of the matrix `UPA` into the two binary matrices `UA` and `PA` having k columns and k rows, respectively, such that:

$$\texttt{UPA} = \texttt{UA} \otimes \texttt{PA} \tag{1}$$

where the operator $\otimes$ denotes the Boolean matrix multiplication defined, for $1 \le i \le n$ and $1 \le j \le m$, as

$$\mathtt{UPA}[i][j] = \bigvee_{h=1}^{k} (\mathtt{UA}[i][h] \wedge \mathtt{PA}[h][j]), \tag{2}$$

where the symbol $\vee$ (resp., $\wedge$) denotes the logical operator or (resp., and).

A *candidate* role can be described by a row of the matrix PA and a column of the matrix UA, while a *candidate role-set* consists of multiple candidate roles and is represented by the matrices UA and PA. A *complete* candidate role-set represents a solution for *equation* $\mathtt{UPA} = \mathtt{UA} \otimes \mathtt{PA}$ containing a (minimal) number of roles such that the permissions included in every row of UPA are exactly *covered* by the union of some of those roles. Then, the role mining problem consists in finding a complete candidate role-set having minimum cardinality, that is, finding a user-to-role assignment $\mathcal{UA}$ and a role-to-permission assignment $\mathcal{PA}$ such that the matrices UA and PA satisfy (1) and the number of columns (rows) of UA (PA) is minimized. This problem can be formalized as follows.

Problem 1. (Role Mining) Given $\mathcal{U}$, $\mathcal{P}$, and $\mathcal{UPA}$, what is the smallest integer $k \leq \min\{|\mathcal{U}|, |\mathcal{P}|\}$ for which there are $\mathcal{R}$, $\mathcal{UA}$, and $\mathcal{PA}$ such that $|\mathcal{R}| = k$ and $\mathtt{UPA} = \mathtt{UA} \otimes \mathtt{PA}$?

The decisional version of Role Mining has been studied in Vaidya et al. (2007) where it was proved to be NP-complete by reducing to the problem Set Basis (SP7 in Garey and Johnson, 1979), and an analogue proof has been reported in Stockmeyer (1975) proving that Set Basis is NP-complete by showing that Vertex Cover can be reduced to Set Basis.

Various *constrained* role mining problems have been considered, where limitations can be imposed on different characteristics of the set of roles, such as the size of roles or their usage frequency, to limit the scope or structure of the generated roles (John et al., 2012; Harika et al., 2015; Blundo et al., 2020a, 2017; Kumar et al., 2010; Hingankar and Sural, 2011). The main constrained role mining problems are as follows:

- Permission-Usage Cardinality Constraint Problem (PUCC): limits the number of permissions that can be assigned to each role (Kumar et al., 2010; Blundo et al., 2020a).
- Role-Usage Cardinality Constraint Problem (RUP): restricts the number of roles that can be assigned to any single user (Harika et al., 2015; John et al., 2012).
- Permission-Distribution Cardinality Constraint Problem (PDP): imposes a limit on how many roles a single permission can appear in Harika et al. (2015) and Blundo et al. (2020a).
- Role Usage and Permission Distribution Constraints Problem (RUPDC): enforces both RUP and PDP simultaneously (Harika et al., 2015).
- Permission-Role-Usage Cardinality Constraints Problem (PRUCC): enforces both RUP and PUCC simultaneously (Blundo et al., 2022).

In this paper, we focus on the PUCC, RUPDC, and PRUCC problems that are formally defined below, where the posive integers *mpr*, *mrcu*, and *mrcp* respectively represent the thresholds: maximum permissions per role, maximum role constraint on user, and maximum role constraint on permission.

Problem 2. (PUCC) Given a set of user $\mathcal{U}$, a set of permission $\mathcal{P}$, a user-permission assignment $\mathcal{UPA}$, and a positive integer *mpr* find a decomposition $(\mathtt{UA}, \mathtt{PA})$ for which the following conditions hold: $\mathtt{UPA} = \mathtt{UA} \otimes \mathtt{PA}$; for all roles $r \in \mathcal{R}$, we have $|\{p : (r, p) \in \mathcal{PA}\}| \leq mpr$; the cardinality of the role-set $\mathcal{R}$ is minimized.

Problem 3. (RUPDC) Given a set of users $\mathcal{U}$, a set of permissions $\mathcal{P}$, a user-permission assignment $\mathcal{UPA}$, and two positive integers *mrcp* and *mrcu*, find a decomposition $(\mathtt{UA}, \mathtt{PA})$ for which the following conditions hold: $\mathtt{UPA} = \mathtt{UA} \otimes \mathtt{PA}$; for all permissions $p \in \mathcal{P}$, we have $|\{r : (r, p) \in \mathcal{PA}\}| \leq mrcp$; for all users $u \in \mathcal{U}$, we have $|\{r : (u, r) \in \mathcal{UA}\}| \leq mrcu$; the cardinality of the role-set $\mathcal{R}$ is minimized.

Problem 4. (PRUCC) Given a set of users $\mathcal{U}$, a set of permissions $\mathcal{P}$, a user-permission assignment $\mathcal{UPA}$, and two positive integers *mpr* and *mrcu*, satisfying $mpr \cdot mrcu \geq |\{p : (u, p) \in \mathcal{UPA}\}|$, for any $u \in \mathcal{U}$, find a decomposition $(\mathtt{UA}, \mathtt{PA})$ for which the following conditions hold: $\mathtt{UPA} = \mathtt{UA} \otimes \mathtt{PA}$; for all roles $r \in \mathcal{R}$, we have $|\{p : (r, p) \in \mathcal{PA}\}| \leq mpr$; for all users $u \in \mathcal{U}$, we have $|\{r : (u, r) \in \mathcal{UA}\}| \leq mrcu$; the role-set $\mathcal{R}$ cardinality is minimized.

In John et al. (2012), Harika et al. (2015), Blundo et al. (2020a) and Blundo et al. (2022), the above problems have been shown to be NP-hard (their decisional versions are NP-complete). The computational complexity of role mining, along with several of its variants, has also been analyzed in Vaidya et al. (2007), Chen and Crampton (2009), Ene et al. (2008) and Vaidya et al. (2010a).

3. The bag of words model

The Bag-of-Words (BoW) model is one of the most commonly adopted approaches for document and image analysis, deployed in natural language processing and information retrieval. The BoW model is based on the extraction of high-frequency words that should provide a compact representation of the document.

More in detail, a text can be represented by the unordered collection of its words, not considering grammar rules and common terms. The BoW model produces a weighted representation of a document, where weights correspond to word frequencies.

A document d is then represented by a vector $d = (x_1, x_2, \ldots, x_n)$,where x_i denotes the number of occurrences of the i-th word w_i in the document and n is the size of the collection of terms. In this context, the BoW model allows one to map a document to a fixed size vector, the size being smaller than the allowed vocabulary.

Around the BoW model, different indexes have been proposed to study some features of a document and of the terms included in it. Term Frequency (TF) measures the frequency with which a term w_i appears in a document:

$$tf_{w_i} = w_i / \sum_{j=1}^{n} w_j. \tag{3}$$

Listing the term frequencies for a document does not necessarily return a meaningful representation of the text since most common words, such as articles or prepositions, will have the highest frequency. A normalization of the term frequency is adopted to achieve a better representation of a document, then weighting each word so that terms that occur very frequently in the document have less value than terms that occur rarely. Indeed, the Inverse Document Frequency (IDF) measures for each term its importance within the document and is computed as:

$$idf(t, d) = \log \frac{|D|}{|\{d \in D : t \in d\}|} \tag{4}$$

D is the set of all the documents in the collection and $|D|$ is the total number of documents. The denominator reports the count of documents containing the term t. The inverse document frequency is a measure of how much information the word provides. It will return a low value for the most common words, as they are present in almost all of the documents.

Finally, the Term Frequency - Inverse Document Frequency (TF_{IDF}) value is a measure that shows how important a word is to a document in a collection.

$$tf_{idf}(w_i, d) = tf_{w_i} \cdot idf(w_i, d) \tag{5}$$

The TF_{IDF} value returns a high value when a term has a high frequency in a given document and a low frequency in the collection of documents, filtering out the most common terms.

3.1. A BoW model for RBAC

In this paper, motivated by the success of the bag-of-words model in text document representation [16,17] and image analysis [18,19],

we propose a simple, yet effective, bag-of-words representation for the RBAC problem. The main idea is to map roles, that are collections of permissions, to documents and permissions to words. Following this analogy, most common permissions (words) are the resources that are allocated to many people and for this reason should be first allocated when forming a role. Most common permissions are identified by smaller values for the *IDF* value, that is, a permission with a small *IDF* value will be assigned to more users than a permission with a higher value. So, the construction of a candidate role can take into account those values both when considering the permissions to be included in a role and when selecting the user to be considered in forming a new role. In case a constraint needs to be satisfied, as in the case of PUCC where the maximum number of permissions included in a role is limited by a threshold, one can sort the permissions in increasing order and consider a new role composed of the first permissions allowed by the considered constraint. In the same spirit, when a selection is to be made among multiple candidate roles, one can select the role whose value, obtained by summing up the included permissions, has the smallest value. It is worth to notice that when trying to execute role mining procedures, the starting situation depicted in the initial UPA matrix provides "documents" in the form of rows where each permission/word appears at most once. So, analysis based on the frequency of words/permissions, or indexes such as TF_{IDF} provides no advantages in terms of selection criterion for candidate role-set.

Taking into account then the starting matrix UPA, for each permission $p \in \mathcal{P}$ assigned to at least one user, it is possible to associate the corresponding *IDF* value, that is, $idf(p)$, by applying Eq. (4), where N represents the number of users (i.e., the number of rows of UPA), and the denominator represents the number of users to whom permission p is assigned. Thus, the *IDF* value for the permission p is defined as

$$idf(p) = \log \frac{|\mathcal{U}|}{|\{u \in \mathcal{U} : (u,p) \in \mathcal{UPA}\}|}. \tag{6}$$

As detailed later, this *IDF* value serves as a criterion to select the next candidate role during the construction of the solution, in order to cover the input UPA. Incorporating the *IDF* value introduces a novel approach to exploring the solution space of potential role sets. Our empirical results demonstrate that this method can, in certain cases, yield improvements over existing heuristics. In literature on information retrieval, different interpretations have been assigned to the IDF and TF_{IDF} measures, trying to establish some theoretical basis in relation to information theory principles formalized by Shannon Robertson (2004) and Aizawa (2003). On this basis, the IDF associated to a term in a document can be interpreted as the amount of information the term conveys, and the TF_{IDF} index measures the significance of a term expressed as a product of the probability that it occurs and the amount of information that it represents (Aizawa, 2000, 2003).

In Aizawa (2003), TF_{IDF} is also interpreted as an approximation of the "probability-weighted amount of information" a term (permission) provides, where high values imply that a permission is both frequent within a specific context (user/role) and rare across the general population, thus conveying a significant and specific piece of information. This aligns perfectly with identifying the core responsibilities that define an optimal role-set. In our work, we defined different heuristics where the IDF values are used to select the next permission or the next role in the procedure for the construction of the solution. BoW and IDF/TF_{IDF} provide powerful feature engineering steps for role mining by transforming raw user-permission assignments into a numerical representation that captures co-occurrence since BoW highlights which permissions tend to appear together, and identifies specificity, since IDF weights permissions to emphasize those that are discriminative and unique to certain user groups/roles. The theoretical underpinning of their suitability lies in their ability to indirectly optimize for information-theoretic properties associated to the constructed role-set leading to more meaningful, efficient, and secure role assignments compared to simply grouping permissions based on raw counts.

4. Heuristics exploiting the *IDF* value

In this section, we present our heuristics for the constrained role mining problems obtained by modifying the basic procedure defined in Blundo et al. (2020a) and adapting it to consider the *IDF* value. We start by describing the basic heuristic and show the needed modification and possible variants. In fact, based on the *IDF* value, there are different possibilities for the selection of the user/row corresponding to the permissions that could be considered when forming the next candidate role. Using the same reasoning, within the permissions assigned to the selected user/row, there are different ways to select the permissions included in the next role, still satisfying the constraint.

4.1. PUCC heuristic

In the scenario considered, we pose a restriction on the number of permissions included in each role, providing a solution to the Permission-Usage Cardinality Constraint Role Mining problem, where from a given initial configuration $\rho = \langle \mathcal{U}, \mathcal{P}, \mathcal{UPA} \rangle$, an RBAC state is returned $\gamma = \langle \mathcal{R}, \mathcal{UA}, \mathcal{PA} \rangle$, where the size of each role is not greather than a given threshold *mpr*.

The heuristic we propose is derived from the RM algorithm introduced in Blundo et al. (2020a), with several adaptations to better suit our context. For a complete description of RM, we refer the reader to Blundo et al. (2020a); here, we briefly summarize its key components common to most role mining heuristics. To compute a decomposition (UA, PA) of the user-to-permission assignment matrix UPA, such that UPA = UA ⊗ PA, the RM heuristic iteratively processes the set UC of *uncovered* users, that is, users whose permissions are not yet fully represented (i.e., *uncovered*) by the current set of roles. In each iteration, a new candidate role, denoted *candidateRole*, is created. This role contains at most *mpr* permissions and is formed by first selecting a user with the fewest assigned permissions. Then, up to *mpr* of that user's permissions are included in the role. The user selection can be based either on the original matrix UPA or on a matrix of uncovered permissions, referred to as uncUPA. Likewise, the permissions in *candidateRole* can be chosen either as the first *mpr* permissions or as a random subset of that size. Next, the algorithm identifies the users affected by *candidateRole*, those whose assigned permissions include all permissions in the role. The matrices UA and PA are then updated accordingly to include *candidateRole*. If any user's permissions are now fully covered, that user is removed from the set UC, and the process repeats.

The *new* heuristic RM-IDF, introduced in this paper (see Algorithm 1), follows the same general framework common to most role mining heuristics. Its key innovation lies in the strategy used to select new candidate roles. Everything hinges on the procedure $\texttt{pickRole}_{\texttt{IDF}}$, presented in Algorithm 2. The chooseUser function (line 1) manages user selection by identifying the user u to be used in the construction of a new candidate role. Then, the function formRole (line 3) generates the candidate role by selecting up to *mpr* permissions from *u*'s uncovered permissions.

As for the heuristic RM, we adopt two fundamental strategies to select the permissions used to form a role. Permissions can be chosen from a row in the original user-permission assignment matrix UPA (strategy O), or from a row in the matrix uncUPA (stategy U). The first approach, based on UPA, leverages the full user-permission information to guide role construction. In contrast, the second approach uses uncUPA, focusing on a progressively reduced problem instance, as the number of uncovered permissions decreases with each mining step. Initially, the two matrices UPA and uncUPA are identical. Notice that uncUPA can be easily calculated starting from UPA, UA, and PA. However, the heuristic could maintain an up-to-date copy of uncUPA (as we did in our Python implementation), preventing repeated recomputation.

Given the user-permission assignment matrix (i.e, either UPA or uncUPA), user selection can be based on two criteria: the number of assigned permissions (i.e., the *length* of their permission set) or the total *IDF* value (i.e., the sum of the *IDF* values of their assigned permissions).

Algorithm 1 RM-IDF.

Require: $n \times m$ matrix UPA, threshold $mpr > 1$, and variant vr
Ensure: A decomposition (UA, PA) of UPA
1: UC ← $[n]$ ▷ Set of uncovered users
2: UA ← $[n][\cdot]$, PA ← $[\cdot][m]$ ▷ Initial *empty* decomposition of UPA
3: **while** UC ≠ ∅ **do**
4: $candidateRole$ ← $\texttt{pickRole}_{\texttt{IDF}}$(UPA, UA, PA, UC, mpr, vr)
5: $selectedU$ ← selectUsers(UPA, UA, PA, UC, $candidateRole$)
6: (UA, PA, UC) ← update(UA, PA, UC, $candidateRole$, $selectedU$)
7: **end while**
8: **return** (UA, PA)

Table 1
Heuristic variants.

Variant	Heuristic	Permissions matrix	User selection	Permissions selection
OLF	upa_len_first	UPA	len	first
OLR	upa_len_rnd	UPA	len	rnd
OLI	upa_len_idf	UPA	len	idf
OIF	upa_idf_first	UPA	idf	first
OIR	upa_idf_rnd	UPA	idf	rnd
OIF	upa_idf_idf	UPA	idf	idf
ULF	uncupa_len_first	uncUPA	len	first
ULR	uncupa_len_rnd	uncUPA	len	rnd
ULI	uncupa_len_idf	uncUPA	len	idf
UIF	uncupa_idf_first	uncUPA	idf	first
UIR	uncupa_idf_rnd	uncUPA	idf	rnd
UII	uncupa_idf_idf	uncUPA	idf	idf

In the first approach, the user with the fewest permissions is selected (strategy L); in the second, the user with the lowest total *IDF* value is chosen (strategy I).

Algorithm 2 $\texttt{pickRole}_{\texttt{IDF}}$.

Require: $n \times m$ matrix UPA, partial decomposition (UA, PA), uncovered users UC, threshold mpr, and variant vr
Ensure: A new role $candidateRole$
1: u ← chooseUser(UPA, UA, PA, UC, vr)
2: $unc Perms$ ← $\{j \in [m] : \texttt{UPA}[u][j]=1$ and $\bigvee_{h=1}^{k}(\texttt{UA}[u][h] \wedge \texttt{PA}[h][j])=0\}$
3: $candidateRole$ ← formRole($unc Perms$, mpr, vr)
4: **return** $candidateRole$

Once a user is selected, a candidate role is formed by choosing up to mpr of their assigned permissions. These permissions can be selected in one of three ways: according to a predefined *order* (strategy F), randomly according to a uniform distribution (strategy R), or by selecting the mpr permissions with the lowest *IDF* values (strategy I). The latter strategy is motivated by the fact that permissions with lower *IDF* values are typically shared by a larger number of users. In the ordered selection strategy, since permissions are either explicitly numbered or naturally ordered, the first mpr permissions are chosen. We deliberately chose not to adopt alternative selection strategies, such as selecting the subset of mpr permissions that occurs most frequently across rows in UPA, which would aim to maximize coverage. Although theoretically appealing, these methods were found to be computationally prohibitive in practice. Identifying such *optimal* set would require excessive processing time, making them impractical for real-world use.

In total, we have twelve variants of the *basic* $\texttt{pickRole}_{\texttt{IDF}}$ procedure. These variants lead to twelve heuristics, whose characteristics are summarized in Table 1. To better clarify their self-explanatory names, Table 1 emphasizes the underlying permission matrix for each heuristic and the criteria used for selecting users and permissions in the role formation process.

Based on the selected user strategy, either L or I, the function chooseUser (Algorithm 2, line 1) returns the user who meets the *minimum* criterion: the user with the smallest permission set (in terms of length) or the lowest total *IDF* value. Regardless of the chosen permission matrix strategy (either O or U), chooseUser can compute, for each user $u \in$ UC, the cardinality of their permission set or their total *IDF* value using the matrices UPA, UA, and PA. To improve efficiency, we assume that chooseUser maintains up-to-date *length* and/or *IDF* values for each user in UC. With this optimization, the worst-case computational complexity of chooseUser becomes $\mathcal{O}(|\texttt{UC}|)$. If the heuristic maintains an up-to-date copy of uncUPA (Algorithm 2, line 2), then the set $unc Perms$ can be computed in constant time. Finally, the function formRole (Algorithm 2, line 3) returns a role $candidateRole$ consisting of at most mpr permissions selected from $unc Perms$ according to variant vr. Selecting either the first mpr permissions (strategy F) or mpr random permissions (strategy R) requires $\mathcal{O}(mpr)$ time. In contrast, using strategy I, which selects the mpr permissions with the lowest *IDF* values, requires $\mathcal{O}(mpr \log mpr)$ time, assuming the permissions are not pre-sorted by *IDF*. Therefore, the worst-case computational complexity of formRole is $\mathcal{O}(mpr \log mpr)$. From the previous discussion, it follows that the overall complexity of the procedure $\texttt{pickRole}_{\texttt{IDF}}$ is $\mathcal{O}(|\texttt{UC}| + mpr \log mpr)$.

The function selectUsers (Algorithm 1, line 5) returns all the users whose set of assigned permissions includes all permissions in $candidateRole$, considering either UPA (if we selected the strategy O) or uncUPA (the strategy U was chosen). Without resorting to any optimization, these users can be computed in $\mathcal{O}(|\texttt{UC}| \cdot mpr)$ time.

For the reader's convenience, we provide below the procedure update, which is invoked in line 6 of Algorithm 1. The function getIndex (Algorithm 3, line 1) returns the index of the role corresponding to the permissions in $candidateRole$. If this role already exists in PA, the function returns the row index where it appears; otherwise, it returns the next available index, that is, $|\texttt{PA}[\cdot][m]| + 1$. Without resorting to any optimization, the function getIndex runs in $\mathcal{O}(|candidateRole| \cdot k) = \mathcal{O}(mpr \cdot k)$ time, since $|candidateRole| \leq mpr$. The function update also modifies the matrix uncUPA of uncovered permissions and, for each user, the counter UP, which tracks the number of permissions still uncovered (Algorithm 3, lines 4–9).

Algorithm 3 Update.

Require: The partial decomposition (UA, PA) of UPA, the set of uncovered users UC, the new role $candidateRole$, and the users $selectedU$ this role will be assigned to
Ensure: The updated partial decomposition (UA, PA) of UPA and the updated set of uncovered users UC
1: k ← getIndex(PA, $candidateRole$) ▷ Role's index
2: **for all** $i \in selectedU$ **do** ▷ Assign the new role to users
3: $\texttt{UA}[i][k] \leftarrow 1$
4: **for all** $j \in candidateRole$ **do**
5: **if** $\texttt{uncUPA}[i][j] == 1$ **then** ▷ The j-th permission was uncovered
6: $\texttt{uncUPA}[i][j] = 0$
7: $\texttt{UP}[i] = \texttt{UP}[i] - 1$
8: **end if**
9: **end for**
10: **if** $\texttt{UP}[i] == 0$ **then** ▷ The permissions of user i are covered
11: $\texttt{UC} = \texttt{UC} \setminus \{i\}$.
12: **end if**
13: **end for**
14: **if** $k > |\texttt{PA}[\cdot][m]|$ **then** ▷ Add the new role to PA
15: **for all** $i \in candidateRole$ **do**
16: $\texttt{PA}[k][i] \leftarrow 1$
17: **end for**
18: **end if**
19: **return** (UA, PA, UC)

It is straightforward to verify that the computational complexity of the function update is $\mathcal{O}(mpr \cdot k + |selectedU| \cdot |candidateRole| +$

$|candidateRole|$) which is $\mathcal{O}(mpr \cdot m + |\mathtt{UC}| \cdot mpr + mpr) = \mathcal{O}(mpr \cdot (m + |\mathtt{UC}|))$, since the number of roles k is upper bounded by the number m of initial permissions, at most $|\mathtt{UC}|$ users can appear in $selectedU$, and $|candidateRole| \leq mpr$.

To conclude, the worst-case computational complexity $\mathtt{C}_{\mathtt{RM\text{-}IDF}}$ of the heuristic `RM-IDF` is

$$\begin{aligned}\mathtt{C}_{\mathtt{RM\text{-}IDF}} &= \sum_{|\mathtt{UC}|=1}^{n} \left[\mathcal{O}(|\mathtt{UC}| + mpr \log mpr) + \mathcal{O}(mpr \cdot |\mathtt{UC}|) + \mathcal{O}(mpr \cdot (m + |\mathtt{UC}|))\right] \\ &= \sum_{|\mathtt{UC}|=1}^{n} \mathcal{O}(mpr \cdot (m + |\mathtt{UC}| + \log mpr)) \\ &= \mathcal{O}(mpr \cdot n \cdot (m + n + \log mpr)).\end{aligned}$$

The three terms in the first summation correspond, respectively, to the computational complexities of the functions $\mathtt{pickRole}_{\mathtt{IDF}}$, `selectUsers`, and `update`.

4.1.1. State-of-the-art heuristics

The PUCC scenario, to our knowledge, is addressed by three heuristics: `PUCC_R` and `PUCC_C` (Blundo et al., 2020a) and `CRM` (*Constrained RoleMiner*) (Kumar et al., 2010).

The `PUCC_R` heuristic, referred to as the `RM` heuristic in Section 4.1, operates by first selecting an uncovered user with the fewest assigned permissions, and then forming a role by selecting the first mpr permissions assigned to that user. Notably, `PUCC_R` and `upa_len_first` share the same role selection method. However, for the sake of completeness in the experimental evaluation in Section 5, we report the performance of both, noting that their execution times differ due to variations in their implementation.

The `PUCC_C` heuristic is a variant of `PUCC_R` that operates on the dual problem, focusing on the columns of the `UPA` matrix. Its core idea is to apply the same selection strategy used in `PUCC_R`, but to the transposed `UPA` matrix, effectively identifying roles from a permission-centric perspective. Specifically, the `PUCC_C` heuristic forms a role by first identifying a permission p_j within `UPA` that exhibits the lowest frequency of assignment across users. Subsequently, given the set of users $\{u_{i_1}, \ldots, u_{i_g}\}$ possessing permission p_j, the heuristic generates a candidate role comprising the initial mpr permissions common to all members of this set of users.

The `CRM` heuristic (Kumar et al., 2010), a derivative of the ORCA approach (Schlegelmilch and Steffens, 2005), employs a permission-based user clustering technique. Clusters are formed by grouping users that possess identical sets of permissions. A new role r is generated from the cluster with the maximum number of users, comprising at most mpr permissions associated with that cluster. This role is then assigned to any cluster c such that the permission set of c is a superset of the permissions in r, and these permissions are subsequently removed from c. The heuristic iteratively reduces permissions and covers users through a recursive application of this process.

In the following, we show how our newly proposed heuristics improve over the state-of-the-art ones in some cases. Specifically, an *adversarial* `UPA` matrix is constructed to evaluate the performance of the heuristics `PUCC_R` and `upa_len_idf`. An adversarial `UPA` matrix is a specially crafted `UPA` designed to test the limits of `PUCC_R`. Its purpose is to exploit the limitations of traditional heuristics, like `PUCC_R`, intentionally making it perform poorly (for example, by forcing `PUCC_R` to create more roles than actually needed). Both heuristics `PUCC_R` and `upa_len_idf` adopt the same criterion to select the user whose permissions will be used to form $candidateRole$ (e.g., the uncovered user having a minimum number of assigned permissions in `UPA`). Recall that, heuristic `PUCC_R` selects permissions according to their *positions* (i.e., it selects the first mpr permissions), while heuristic `upa_len_idf` selects permissions according to their *importance* (i.e., it selects mpr permissions having the smallest *IDF* value as a permission with low *IDF* value is assigned to more users than a permission with higher *IDF* value). Since the heuristic `upa_len_idf` selects the users considering the `UPA` matrix, the *IDF* values do not change during the mining process. Hence, the adversarial `UPA` matrix is designed based on the following rationales:

- **Common Permissions with Low *idf***: Some permissions are shared by most users (such permissions have a low *IDF* value).
- **Unique or Sparse Permissions with High *idf***: Other permissions are assigned more selectively and only appear in a few users (such permissions have a high *IDF* value).
- **Strategic Ordering**: The matrix is structured so that the heuristic `PUCC_R` that selects the first mpr permissions (by position) will repeatedly select less informative or suboptimal sets of permissions.
- **Controlled Growth**: A *base* matrix is replicated in blocks (e.g., $2\times$, $3\times$, ...) to test scalability and consistently challenge non-IDF-based heuristics.

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	#p
u_1	1	1	1	1	0	0	0	0	4
u_2	0	1	0	1	1	1	1	0	5
u_3	1	1	1	1	1	0	0	1	6
u_4	1	1	1	1	1	0	1	1	7
u_5	1	1	1	1	1	1	1	1	8
IDF	0.32	0.00	0.32	0.00	0.32	1.32	0.74	0.74	

Fig. 1. $\mathtt{UPA}_{\mathtt{B}}$ matrix.

With these principles in mind and setting $mpr = 2$, we construct the following matrix $\mathtt{UPA}_{\mathtt{B}}$ where two permissions (that is, p_1 and p_3) are assigned to all users, but u_2, and the other two permissions (that is, p_2 and p_4) are assigned to all users. The other four permissions are assigned to all users but u_1 in such a way that users $\{u_2, \ldots, u_5\}$ have an increasing number of permissions (i.e., user u_i has $3 + i$ permissions). The complete assignment is represented by the matrix $\mathtt{UPA}_{\mathtt{B}}$ in Fig. 1, where, at the bottom, we report the *IDF* values and, in the column headed by #p, the number of permissions assigned to users.

Both heuristics, `PUCC_R` and `upa_len_idf`, result in the same ordered selection of users, specifically $u_1, u_2, \ldots, u_5$, in that sequence, with the possibility of earlier termination. Once selected user u_1, forced by the *position* criterion, heuristic `PUCC_R` will first choose permissions p_1 and p_2, then permissions p_3 and p_4. On the other hand, heuristic `upa_len_idf` will first select permissions p_2 and p_4, then permissions p_1 and p_3, covering a larger part of the $\mathtt{UPA}_{\mathtt{B}}$ matrix. To amplify the effect of the $\mathtt{UPA}_{\mathtt{B}}$'s structure, we duplicate it as shown in Fig. 2.

The `PUCC_R` heuristic processes the `UPA` matrix in Fig. 2 by first selecting user u_1. This leads to the identification of roles $r_1 = \{p_1, p_2\}$, $r_2 = \{p_3, p_4\}$, $r_3 = \{p_9, p_{10}\}$, and $r_4 = \{p_{11}, p_{12}\}$. After assigning these roles to users u_1 through u_4, all permissions for u_1 are covered. Consequently, the next user selected by `PUCC_R` for mining is u_2, and the state of uncovered permissions at this point is represented by the matrix in Fig. 3.

Heuristic `PUCC_R` then processes user u_2's remaining permissions from Fig. 3 (i.e., $\{p_2, p_4, p_5, p_6, p_7, p_{10}, p_{12}, p_{13}, p_{14}, p_{15}\}$), creating roles $r_5 = \{p_2, p_4\}$, $r_6 = \{p_5, p_6\}$, $r_7 = \{p_7, p_{10}\}$, $r_8 = \{p_{12}, p_{13}\}$, and $r_9 = \{p_{14}, p_{15}\}$. With u_2's permissions now covered, the resulting `UA` matrix and uncovered permission matrix are shown in Fig. 4.

Next, the `PUCC_R` heuristic selects user u_3 and generates the roles $r_{10} = \{p_5, p_8\}$ and $r_{11} = \{p_{16}\}$, assigning both to users u_3, u_4, and u_5 (recall that role assignment is based on the permissions in `UPA` assigned to uncovered users). Consequently, the uncovered permission matrix `UPA` contains only one entry, indicating that user u_4 has the uncovered permission p_{15}. Therefore, the final mining step involves generating the role $r_{12} = \{p_{15}\}$ and assigning it to user u_4.

In conclusion, the `PUCC_R` heuristic generates twelve roles to satisfy the user-to-permission assignment matrix `UPA` in Fig. 2.

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	p_{13}	p_{14}	p_{15}	p_{16}	#p
u_1	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	8
u_2	0	1	0	1	1	1	1	0	0	1	0	1	1	1	1	0	10
u_3	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	12
u_4	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	14
u_5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	16
IDF	0.32	0.00	0.32	0.00	0.32	1.32	0.74	0.74	0.32	0.00	0.32	0.00	0.32	1.32	0.74	0.74	

Fig. 2. UPA matrix.

	p_2	p_4	p_5	p_6	p_7	p_8	p_{10}	p_{12}	p_{13}	p_{14}	p_{15}	p_{16}
u_2	1	1	1	1	1	0	1	1	1	1	1	0
u_3	0	0	1	0	0	1	0	0	1	0	0	1
u_4	0	0	1	0	1	1	0	0	1	0	1	1
u_5	0	0	1	1	1	1	0	0	1	1	1	1

Fig. 3. uncUPA matrix for PUCC_R.

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9
u_1	1	1	1	1	0	0	0	0	0
u_2	0	0	0	0	1	1	1	1	1
u_3	1	1	1	1	1	0	0	1	0
u_4	1	1	1	1	1	0	1	1	0
u_5	1	1	1	1	1	1	1	1	1

	p_5	p_8	p_{15}	p_{16}
u_3	1	1	0	1
u_4	1	1	1	1
u_5	0	1	0	1

Fig. 4. UA (left) and uncUPA (right) matrices for PUCC_R.

	p_5	p_6	p_7	p_8	p_{13}	p_{14}	p_{15}	p_{16}
u_2	1	1	1	0	1	1	1	0
u_3	1	0	0	1	1	0	0	1
u_4	1	0	1	1	1	0	1	1
u_5	1	1	1	1	1	1	1	1

Fig. 5. uncUPA matrix for upa_len_idf.

Similarly, the upa_len_idf heuristic begins with user u_1 and produces the roles $r_1 = \{p_2, p_4\}$, $r_2 = \{p_{10}, p_{12}\}$, $r_2 = \{p_1, p_3\}$, and $r_3 = \{p_9, p_{11}\}$. As with PUCC_R, these roles are assigned to users u_1 through and u_4, leading to the uncovered permission matrix in Fig. 5.
The upa_len_idf heuristic then processes user u_2's uncovered permissions from Fig. 5 (i.e., $\{p_5, p_6, p_7, p_{13}, p_{14}, p_{15}\}$), creating roles $r_5 = \{p_5, p_{13}\}$, $r_6 = \{p_7, p_{15}\}$, and $r_7 = \{p_6, p_{14}\}$. With u_2's permissions now covered, the resulting UA matrix and uncovered permission matrix are shown in Fig. 6.
In the final mining step, the upa_len_idf heuristic selects user u_3 and creates the role $r_8 = \{p_8, p_{16}\}$, which is then assigned to users u_3, u_4, and u_5.

For brevity, without detailing the step-by-step mining process, the resulting UA and PA matrices for the PUCC_C and CRM heuristics are provided below in Figs. 7–10.

	r_1	r_2	r_3	r_4	r_5	r_6	r_7
u_1	1	1	1	1	0	0	0
u_2	1	1	0	0	1	1	1
u_3	1	1	1	1	1	0	0
u_4	1	1	1	1	1	1	0
u_5	1	1	1	1	1	1	1

	p_8	p_{16}
u_3	1	1
u_4	1	1
u_5	1	1

Fig. 6. UA (left) and uncUPA (right) matrices for upa_len_idf.

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}	r_{11}	r_{12}	r_{13}	r_{14}	r_{15}	r_{16}
u_1	0	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0
u_2	1	1	0	0	0	1	0	0	1	0	0	0	1	0	0	1
u_3	0	0	1	0	1	0	1	0	0	1	0	1	0	1	0	0
u_4	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
u_5	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Fig. 7. UA matrix for PUCC_C.

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	p_{13}	p_{14}	p_{15}	p_{16}
r_1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
r_2	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
r_3	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r_4	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
r_5	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
r_6	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
r_7	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0
r_8	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
r_9	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0
r_{10}	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
r_{11}	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
r_{12}	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
r_{13}	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
r_{14}	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
r_{15}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
r_{16}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Fig. 8. PA matrix for PUCC_C.

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}	r_{11}	r_{12}	r_{13}	r_{14}	r_{15}	r_{16}	r_{17}
u_0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
u_2	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0
u_3	1	1	1	1	0	0	0	0	0	1	1	0	0	0	0	0	0
u_4	1	1	0	0	0	0	0	1	0	0	0	1	1	1	1	0	0
u_5	1	1	1	1	0	1	0	0	0	0	0	0	0	0	1	1	1

Fig. 9. UA matrix for CRM.

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	p_{13}	p_{14}	p_{15}	p_{16}
r_1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r_2	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
r_3	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
r_4	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
r_5	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
r_6	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
r_7	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0
r_8	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
r_9	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
r_{10}	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0
r_{11}	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
r_{12}	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
r_{13}	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
r_{14}	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
r_{15}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
r_{16}	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
r_{17}	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

Fig. 10. PA matrix for CRM.

Table 2
Heuristics' outcome.

Heuristic	$\lvert\mathcal{R}\rvert$	|UA|	|PA|
PUCC_R	12	38	22
PUCC_C	16	32	31
CRM	17	30	34
upa_len_first	12	38	22
upa_len_rnd	10	33	20
upa_len_idf	8	30	16
upa_idf_first	12	36	23
upa_idf_rnd	10	32	19
upa_idf_idf	8	30	16
uncupa_len_first	11	30	22
uncupa_len_rnd	10	30	20
uncupa_len_idf	8	30	16
uncupa_idf_first	11	30	22
uncupa_idf_rnd	10	30	20
uncupa_idf_idf	8	30	16

Notably, the upa_len_idf heuristic produces fewer roles than the three benchmark heuristics. A similar pattern is evident for our other heuristics, as summarized in Table 2.

This result is not accidental. Replicating the structural properties of the $\mathtt{UPA}_{\mathtt{B}}$ matrix depicted in Fig. 1, we can synthesize *adversarial* user-to-permission assignment matrices that accommodate a nearly arbitrary number of users and permissions. Under these conditions, our heuristics

$$\begin{pmatrix} \mathtt{UPA}_{\mathtt{B}} & \mathtt{UPA}_{\mathtt{B}} & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathtt{UPA}_{\mathtt{B}} & \mathtt{UPA}_{\mathtt{B}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathtt{UPA}_{\mathtt{B}} & \mathtt{UPA}_{\mathtt{B}} \end{pmatrix}$$

Fig. 11. Adversarial UPA matrix for $n = 3$.

consistently demonstrate superior performance relative to the state-of-the-art. Indeed, as for the construction of the matrix in Fig. 2, by replicating $2n$ times the matrix $\mathtt{UPA}_{\mathtt{B}}$, we can realize a UPA matrix having five users and $16n$ permissions. In this case, the PUCC_R, PUCC_C, CRM, and upa_len_idf generate $12n - 2\lfloor n/2 \rfloor$, $16n$, $\geq 17n$, and $8n$ roles, respectively. Following an analogous procedure, it is possible to construct a UPA matrix of size $5n \times 16n$ (i.e., $5n$ users and $16n$ permissions). This matrix is defined by having n instances of the matrix illustrated in Fig. 2 (comprising two contiguous $\mathtt{UPA}_{\mathtt{B}}$ matrices) positioned along its principal diagonal, with all other matrix elements set to zero. In Fig. 11, we provide an example of this matrix construction when $n = 3$, where the symbol 0 denotes a 5×16 block of zeros.
In this case, the PUCC_R, PUCC_C, CRM, and upa_len_idf generate $12n$, $16n$, $17n$, and $8n$ roles, respectively.

4.2. ERUPDC heuristic

The Role-Usage Cardinality Constraint Problem (RUP) and the Permission-Distribution Cardinality Constraint Problem (PDP) were investigated in Harika et al. (2015). The RUP restricts the number of roles that can be assigned to any single user, governed by the threshold parameter *mrcu* (maximum role constraint per user). Its dual, the PDP, imposes a limit on how many roles a single permission can appear in, defined by the threshold *mrcp* (maximum role constraint per permission). To address both constraints simultaneously, the authors of Harika et al. (2015) proposed a heuristic called ERUPDC (Enforce Role Usage and Permission Distribution Constraints), designed specifically to mine roles that satisfy both *mrcu* and *mrcp*.

The ERUPDC heuristic operates in two phases. In the first phase, it selects either a user or a permission based on one of four variants (described below). A user u (respectively, a permission p) can only be selected if the number of roles assigned to u (respectively, the number of roles in which p appears) is less than $mrcu - 1$ (respectively, $mrcp - 1$). In the second phase, the heuristic picks either the user with the highest number of uncovered permissions or the permission associated with the highest number of uncovered users. Only users and permissions that do not violate the cardinality constraints are considered. If a user u is selected, a role r is formed from all of u's permissions that do not violate the PDP constraint. This role is then assigned to all users who possess all permissions in r and do not exceed the RUP constraint. A symmetric approach is used if a permission p is selected. The four variants for the first phase are:

- NU and NP: Select either a user with the fewest uncovered permissions or a permission with the fewest uncovered users. NU prioritizes users in case of a tie; NP prioritizes permissions.
- XR: Chooses the user or permission with the greatest number of remaining role assignments allowed before hitting their constraint. In case of a tie, preference is given to the user with the fewest uncovered permissions or the permission with the fewest uncovered users.
- NR: Follows the same strategy as XR but selects the user or permission with the least number of roles remaining before reaching their constraint limit.

Because the two constraints can impose conflicting requirements, certain combinations of *mrcu* and *mrcp* may prevent the heuristic from finding a valid decomposition that satisfies both constraints. The heuristic fails to generate final UA and PA matrices satisfying Eq. (1). In such a case, the result is considered an *incomplete* role-set. Any remaining

uncovered permissions are handled separately through a *direct user-permission* assignment relation $\mathcal{DUPA} \subseteq \mathcal{U} \times \mathcal{P}$ (Molloy et al., 2008, 2010). Further details are provided in Sections 5 and 5.2.

The purpose of the remaining part of this section is not to introduce a new heuristic for enforcing RUP and PDP constraints simultaneously. Rather, we try to assess whether integrating *IDF* values into the ERUPDC heuristic can enhance its role construction process. In particular, variants `NU` and `NP` can be extended by selecting users or permissions based on their (total) *IDF* value. The *IDF* values can be calculated in two ways: Statically at the beginning of the heuristic using the original matrix `UPA` (strategy `F`); Dynamically, updating ir values after each role is mined, based on the uncovered permissions matrix `uncUPA` (strategy `D`). This leads to four additional variants for the first phase:

- `FIU` and `FIP`: Select either a user with the maximum total *IDF* value or a permission with the maximum *IDF* value, using precomputed values from the initial matrix (`F` strategy). In case of a tie, `FIU` prioritizes users, while `FIP` prioritizes permissions.
- `DIU` and `DIP`: Follow the same selection logic as `FIU` and `FIP`, respectively, but the *IDF* values are recomputed dynamically after each role is mined (`D` strategy).

The use of *IDF* values cannot be incorporated into the `XR` and `NR` variants, as their selection strategy is based solely on choosing the user or permission with the highest (or lowest, respectively) number of remaining role assignments before reaching the specified constraint.

It is immediately evident that, under the proposed *IDF*-based strategy, users are more likely to be selected than permissions (i.e., ties in variants `FIP` and `DIP` are not very common). This is because a user's selection is based on their total *IDF* value, whereas for permissions only the individual *IDF* value is considered.

Unlike the PUCC heuristic, here we prioritize users or permissions with the maximum (total) *IDF* value. This choice aligns with the need to enforce the RUP and PDP constraints. Since limiting the number of roles per user encourages the formation of *larger* roles (which typically result in fewer roles per user), we give preference to users whose assigned permissions are, overall, less common, that is, users with the highest total *IDF* values. These users are likely to be harder to cover while satisfying the constraints, so addressing them earlier can improve overall coverage efficiency. This approach is conceptually similar to the `NU` variant, which selects users with the fewest uncovered permissions. However, instead of focusing on quantity, we prioritize users with the least frequent permission sets. A similar rationale applies when breaking the ties in favor of permissions.

4.3. PRUCC heuristic

The combined enforcement of the Role-Usage Cardinality Constraint Problem (RUP) and the Permission-Usage Cardinality Constraint Problem (PUCC) was examined in Blundo et al. (2022), where the joint problem was referred to as the Permission-Role-Usage Cardinality Constraint Problem (PRUCC). To tackle this dual constraint scenario, the authors introduced two heuristics, PRUCC_1 and PRUCC_2. In this context, as in Section 4.2, the RUP constraint is governed by the threshold parameter *mrcu* (maximum number of roles assignable per user), while the PUCC constraint limits the number of permissions that can be included in any role, controlled by the parameter *mpr* as in Section 4.1.

In Blundo et al. (2022), four strategies were proposed for selecting permissions included in roles, using a classification similar to that introduced in Section 4.1. The selection of the user, whose permissions will be used to form a role, is based either on the original matrix `UPA` (strategy `O`) or on the uncovered permissions matrix `uncUPA` (strategy `U`). Both heuristics in Blundo et al. (2022) select a user with the fewest assigned permissions, following strategy `L`. Once a user is chosen, the permissions are always selected from `uncUPA`. To choose up to *mpr* permissions from those assigned to the selected user, the heuristics either select the first *mpr* permissions (strategy `F`) or uniformly sample *mpr* permissions. As a result, the four heuristic variants introduced in Blundo et al. (2022) are:

- `OLF`: Uses the original matrix (`UPA`) and selects the first *mpr* permissions;
- `OLR`: Uses the original matrix and selects *mpr* uniformly distributed permissions;
- `ULF`: Uses the uncovered matrix (`uncUPA`) and selects the first *mpr* permissions;
- `ULR`: Uses the uncovered matrix and selects *mpr* uniformly distributed permissions.

As in Section 4.1, we can extend the heuristics PRUCC_1 and PRUCC_2 by incorporating strategies based on *IDF* values. Specifically, we select the user with the lowest total *IDF* value and choose the *mpr* permissions with the lowest individual *IDF* values, this approach is referred to as strategy `I`. Including this extension, we obtain a total of twelve heuristic variants: `OLF`, `OLR`, `OLI`, `OIF`, `OIR`, `OII`, `ULF`, `ULR`, `ULI`, `UIF`, `UIR`, and `UII` (the original variants introduced in Blundo et al. (2022) are highlighted in green).

Simultaneously enforcing constraints on the maximum number of roles per user and the maximum number of permissions per role suggests two guiding principles: (i) minimize, as much as possible, the assignment of overlapping roles to the same user, and (ii) generate roles that are as large as possible without exceeding the threshold *mpr*. To this end, prioritizing the coverage of the most frequently occurring permissions, through *IDF*-based strategies, may prove advantageous.

5. Experimental results

In this section, we present the application of the proposed technique to explore the solution space in constrained role mining scenarios. For the PUCC case, we compare the twelve heuristic variants described in Section 4.1 with the state-of-the-art methods, namely PUCC_R and PUCC_C (Blundo et al., 2020a), and CRM (Kumar et al., 2010). In scenarios involving multiple simultaneous constraints, we evaluate whether incorporating *IDF* values into the ERUPDC and PRUCC heuristics (Sections 4.2 and 4.3) improves the quality of role construction. Our comparison considers both the quality of the solutions, using standard metrics, and the performance in terms of execution time.

All heuristics have been implemented in Python and are available on GitHub (Last commit: db6aa7f, 2025). The heuristics have been tested on a MacBook Pro-running macOS 15.5 on a 2.3 GHz Intel Core i9 8 core CPU with 16 GB 2667 MHz DDR4 RAM. For evaluation, we consider eleven benchmark datasets that have been obtained from real-world data and are commonly used (Ene et al., 2008; Molloy et al., 2009; Kumar et al., 2010; Harika et al., 2015; John et al., 2012; Nobi et al., 2022; Tyagi et al., 2024). We report the parameters of the eleven datasets in Table 3.

Table 3 summarizes, for each dataset, the following information: the number of users $|\mathcal{U}|$, the number of permissions $|\mathcal{P}|$, the number of user-to-permission assignments $|\mathcal{UPA}|$, the minimum and maximum number of permissions assigned to each user (respectively, min#P and max#P), the minimum and the maximum number of users sharing the same permission (respectively, min#U and max#U), and the density of the matrix $\mathcal{UPA}$, calculated as $|\mathcal{UPA}|/(|\mathcal{U}| \times |\mathcal{P}|)$.

For eight of the datasets, the optimal solutions for the unconstrained scenario are known and reported in Ene et al. (2008). Table 4 summarizes the parameters of these optimal solutions. The columns provide key characteristics of each solution, including the number of roles, the minimum and maximum number of permissions per role, the number of roles assigned to users, the number of roles associated with each permission, and the number of users assigned to each role.

The first nine datasets listed in Table 3, originally used for evaluation in Ene et al. (2008), were provided by researchers at HP Labs. Several of these datasets originate from network access control rules

Table 3
Characteristics of benchmark datasets.

Dataset	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{P}\rvert$	$\lvert\mathcal{UPA}\rvert$	min#P	max#P	min#U	max#U	Density
Americas large	3485	10,127	185,294	1	733	1	2812	0.00525
Americas small	3477	1587	105,205	1	310	1	2866	0.01907
Apj	2044	1164	6841	1	58	1	291	0.00288
Customer	10,021	277	45,427	1	25	1	4184	0.01637
Domino	79	231	730	1	209	1	52	0.04000
Emea	35	3046	7220	9	554	1	32	0.06772
Firewall 1	365	709	31,951	1	617	1	251	0.12347
Firewall 2	325	590	36,428	6	590	46	298	0.18998
Healthcare	46	46	1486	7	46	3	45	0.70227
Amazon UPA 1	9298	7226	30,872	1	36	1	836	0.00046
Amazon UPA 2	9298	7226	32,769	1	36	1	839	0.00045

Table 4
Characteristics of optimal RBAC states.

Dataset	$\lvert\mathcal{R}\rvert$	$\overset{min}{ppr}$	$\overset{max}{ppr}$	$\overset{min}{rpu}$	$\overset{max}{rpu}$	$\overset{min}{rpp}$	$\overset{max}{rpp}$	$\overset{min}{upr}$	$\overset{max}{upr}$
Americas large	398	1	733	1	4	1	129	1	2777
Americas small	178	1	263	1	12	1	43	1	2809
Apj	453	1	52	1	8	1	15	1	278
Domino	20	1	201	1	9	1	6	1	51
Emea	34	9	554	1	1	1	31	1	2
Firewall 1	66	1	395	1	9	1	18	1	203
Firewall 2	10	2	307	1	3	1	4	1	239
Healthcare	14	1	32	1	6	1	4	1	27

used by Hewlett-Packard (HP) to manage external business partner connectivity. Specifically, two user profiles, *Americas small* and *Americas large*, were extracted from Cisco firewalls, authenticating external users and grant them restricted access to the HP network based on their profiles. Additional, smaller datasets, *Apj* and *Emea*, were collected in a similar manner. The *Firewall 1* and *Firewall 2* datasets were generated through Checkpoint firewalls analysis. These datasets capture information on whether service-related packets can access particular network destinations. The *Healthcare* dataset was sourced from the U.S. Department of Veterans Affairs, which maintains a comprehensive list of healthcare-related permissions that may be granted to licensed or certified providers. The *Domino* dataset was based on user and access profiles for a Lotus Domino server. Finally, the *Customer* dataset was derived from an access control graph obtained through collaboration with the IT department of an HP customer.

The last two datasets listed in Table 3, *Amazon UPA 1* and *Amazon UPA 2*, were derived from the *Amazon 1* dataset (Nobi et al., 2021), which was originally developed to predict employees' access needs based on their job roles (Nobi et al., 2022; Tyagi et al., 2024). *Amazon 1* itself is based on data released for the Kaggle competition `Amazon Employee Access Challenge` (Hamner et al., 2013), which contains historical access records from Amazon collected in 2010 and 2011. In this dataset, access decisions (grants or denials) were manually made for employees who attempted to access various resources. Each entry in the original Kaggle dataset consists of eight metadata fields describing the user, a resource identifier, and a binary label indicating whether access was granted. One of the metadata fields identifies the user's manager, assuming that each employee has only one manager at a time. In Nobi et al. (2022), the authors transformed the Kaggle dataset into a new dataset, *Amazon 1*, with authorization tuples in the following format: *uid*, a unique user identifier; *rid*, a unique resource identifier; eight user metadata attributes; one resource metadata attribute; access label (1 = allow, 0 = deny). From this *Amazon 1* dataset, we derived two datasets:

- *Amazon UPA 1*: includes only the tuples with access granted (access = 1). For each such tuple, we set $\texttt{UPA}[uid][rid] = 1$, interpreting `rid` as a permission identifier.
- *Amazon UPA 2*: includes all tuples, regardless of the access label. In this case, $\texttt{UPA}[uid][rid] = 1$ for every entry in *Amazon 1*.

The comparisons in the following sections will consider, in addition to execution time, several key metrics: the number of roles, user-to-role assignments, role-to-permission assignments, and the *Weighted Structural Complexity* (WSC) (Li et al., 2007; Molloy et al., 2008), defined below.

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

$$wsc(\gamma, W) = w_r \cdot |\mathcal{R}| + w_u \cdot |\mathcal{UA}| + w_p \cdot |\mathcal{PA}| + w_h \cdot |t_{reduce}(\mathcal{RH})| + w_d \cdot |\mathcal{DUPA}|, \quad (7)$$

where $t_{reduce}(\mathcal{RH})$ is the transitive reduction of the role hierarchy $\mathcal{RH} \subseteq \mathcal{R} \times \mathcal{R}$ (a.k.a, *inheritance* relation) and $\mathcal{DUPA} \subseteq \mathcal{U} \times \mathcal{P}$ is the *direct user-permission* assignment relation, while $|\cdot|$ denotes the size of the relation.

The the role hierarchy $\mathcal{RH}$ was introduced in Sandhu et al. (1996), when defining Hierarchical RBAC (or RBAC 1). It corresponds (Ferraiolo et al., 2001; Sandhu et al., 2000) to a partial order over $\mathcal{R}$ and represents a *inheritance* relation among roles. That is, the role r_1 *inherits* r_2 if and only if all permissions assigned to r_2 are also assigned to r_1, and all users assigned to r_1 are also assigned to r_2. The direct user-permission assignment relation $\mathcal{DUPA}$ is useful when considering *incomplete* role-sets (meaning that not all the permissions have been covered in `UPA` matrix). The relation $\mathcal{DUPA}$ is not considered in standard RBAC models (Sandhu et al., 2000), but this approach is more general and can handle anomalous situations, such as when a permission assignment to a user cannot be easily justified by an existing role, or when creating a new role for just one permission does not make practical sense. A complete discussion of WSC can be found in Li et al. (2007) and Molloy et al. (2008). In our experiments in Sections 5.1 and 5.3, we consider the weight vector $W = \langle 1, 1, 1, 0, \infty \rangle$, since our solutions do not consider hierarchical RBAC and return a complete role-set. For the experiments in Section 5.2 we will consider $W = \langle 1, 1, 1, 0, 1 \rangle$, since when the heuristics are unable to return a solution for Eq. (1) they will handle uncovered permissions via a direct user-permission relation $\mathcal{DUPA}$ (Molloy et al., 2008, 2010).

5.1. PUCC evaluation

In this section, we focus on the PUCC scenario, that is, finding a candidate role set where there is a threshold on the number of permissions that can be included in each role. We evaluated the twelve heuristic variants described in Section 4.1 with state-of-the-art methods, namely PUCC_R and PUCC_C (Blundo et al., 2020a), and CRM (Kumar et al., 2010). The heuristics were compared considering four values for the constraint *mpr*, setting it to 25 %, 50 %, 75, and 100 % of the maximum number of permissions assigned to each user (see column max#P in Table 3). We completed a set of experiments, considering the eleven datasets and the different PUCC scenarios, as reported in Last commit:

db6aa7f (2025). For the sake of conciseness, here we discuss the results relative to the *Americas large* and the *Amazon UPA 1* datasets.

Table 5 reports the results for the *Americas large* dataset considering all 12 variants of the proposed heuristics, and a constraint on the number of permissions in any role ranging from 25 %, 50 %, to 75 % of the threshold, corresponding to the values for *mpr*, of 183, 366, and 549, respectively (best results highlighted in red). When $mpr = 733$, which is equivalent to running the procedure without constraints, since the optimal solution does not have a role that includes a greater number of permissions, it can be seen that the introduced criterion based on `idf` achieves the best value for the minimum number of roles among all possible heuristics, that is, 413. This value is obtained operating on `UPA` and independently of the criterion selected to form a new $candidateRole$. In general, it is possible to observe that in this case the heuristics return results grouped according to the choice made for the starting matrix (`UPA` or `uncUPA`) and for the chosen row election criterion (`len` or `idf`), while changing the rule for selecting the permissions among `len`, `rnd`, and `idf` does not change the resulting values. This is not true for the other considered runs of the procedure when the threshold *mpr* assumes different values.

When $mpr = 183$, it is possible to notice that heuristic `ULI` obtains the minimum value for the number of roles among all the other heuristics, while the best values for C and $|\mathcal{PA}|$ are achieved by `CRM` heuristic and by `UII` heuristic for $|\mathcal{UA}|$. In any case, it is worth noting that *IDF* contributes to the way in which the space of possible solutions can be explored, providing results that are different from those previously reported by other heuristics. When $mpr = 366$, heuristic `UII` achieves the best value for $|\mathcal{UA}|$ and `ULF` achieves the best values for both WSC and $|\mathcal{PA}|$. Both heuristics improve definitely heuristic `CRM`, whose solution provides the minimum number of roles. Finally, in case of $mpr = 549$, heuristics starting from matrix `upa` and selecting the row with minimum `idf` provide solutions that report better (and different) values for $|\mathcal{UA}|$. In particular heuristics `UIF` and `UII` achieve the second and third best value for $|\mathcal{R}|$, while the minimum value is still returned by `CRM`.

In Table 6 we report the results obtained for the *Amazon UPA 1* dataset, considering different values for the threshold on the number of permissions allowed in each role. We use the symbol `ET` to indicate that the heuristic did not complete within the allotted execution time that was set for our experiments at a time limit of three minutes per run. Note that, since times are reported in milliseconds, this threshold is approximately four times longer than the maximum time required by the other heuristics to finish. Regarding the results obtained, it is worth noting that heuristics based on `uncUPA` achieve the best results in terms of the number of returned roles, improving over previous heuristics presented in the literature, and also return the best results in some cases considering also WSC and $|\mathcal{PA}|$.

5.1.1. Synthetic datasets

In this section, following the methodology proposed in Vaidya et al. (2006), we evaluate the performance of the twelve variants of our heuristic using datasets generated by a synthetic data generator. These experiments are designed to assess the scalability of our approach on increasingly large datasets. The random data generator, first introduced in Vaidya et al. (2006), uses five parameters to simulate various access control configurations: the number of users *nu*; the number of roles *nr*; the number of permissions *np*; the maximum number of roles a user can have *mru*; the maximum number of permissions a role can have *mpr*. The generation process involves three main steps: *Role-Permission Assignment*, for each role, a random number of permissions (up to *mpr*) are randomly selected and assigned; *User-Role Assignment*, each user is assigned a random number of roles (up to *mru*); *User-Permission Computation*: finally, user-permission assignments are calculated based on the established user-role and role-permission relationships. It's important to note that the data produced by this generator is unstructured, treating each user, role, and permission as statistically independent entities.

Table 5
Results for the dataset *Americas large*.

mpr	Measure	PUCC_R	PUCC_C	CRM	OLF	OLR	OLI	OIF	OIR	OII	ULF	ULR	ULI	UIF	UIR	UII
183	$\|\mathcal{R}\|$	578	2850	590	578	728	601	580	723	601	589	730	**572**	589	733	594
	WSC	61,205	481,679	**52115**	61,205	88,180	65,363	61,895	87,861	65,796	63,519	89,103	59,615	63,719	88,552	63,783
	$\|\mathcal{UA}\|$	4540	8717	8242	4540	4667	4626	4466	4662	4477	4536	4817	4516	4463	4750	**4441**
	$\|\mathcal{PA}\|$	56,087	470,112	**43283**	56,087	82,785	60,136	56,849	82,476	60,718	58,394	83,556	54,527	58,667	83,069	58,748
	time	**267**	13,846	1381	277	399	301	269	374	274	10,074	11,533	9922	9847	12,240	10,018
366	$\|\mathcal{R}\|$	495	1500	**454**	495	537	508	495	538	504	499	542	497	502	537	500
	WSC	**73922**	445,757	74,744	**73922**	89,958	78,405	75,094	90,532	78,237	74,847	91,127	74,844	76,888	89,708	75,995
	$\|\mathcal{UA}\|$	4220	5594	5004	4220	4494	4190	4142	4200	**4102**	4260	4442	4239	4195	4351	4177
	$\|\mathcal{PA}\|$	**69207**	438,663	69,286	69,207	84,927	73,707	70,457	85,794	73,631	70,088	86,143	70,108	72,191	84,820	71,318
	time	264	7327	1919	267	347	291	**254**	318	265	8795	9606	8524	8771	9063	8539
549	$\|\mathcal{R}\|$	434	687	**419**	434	440	435	432	438	433	438	442	434	437	439	437
	WSC	91,071	233,518	**86870**	91,071	93,117	91,060	90,886	92,822	90,875	91,610	93,582	88,898	90,970	92,782	90,498
	$\|\mathcal{UA}\|$	4030	4052	4090	4030	4039	4025	3959	3981	**3954**	4147	4155	4129	4066	4108	4044
	$\|\mathcal{PA}\|$	86,607	228,779	**82361**	86,607	88,638	86,600	86,495	88,403	86,488	87,025	88,985	84,335	86,467	88,235	86,017
	time	283	3075	4927	283	300	407	**258**	268	289	7699	7807	8184	8401	8123	7818
733	$\|\mathcal{R}\|$	415	422	415	415	415	415	**413**	**413**	**413**	415	415	415	415	415	415
	WSC	93,255	101,675	**91677**	93,255	93,255	93,255	93,070	93,070	93,070	93,294	93,294	93,294	93,206	93,206	93,206
	$\|\mathcal{UA}\|$	3974	**3714**	4132	3974	3974	3974	3903	3903	3903	4075	4075	4075	4007	4007	4007
	$\|\mathcal{PA}\|$	88,866	97,539	**87130**	88,866	88,866	88,866	88,754	88,754	88,754	88,804	88,804	88,804	88,784	88,784	88,784
	time	287	1897	7133	294	284	315	**257**	355	271	7648	7544	7637	7404	7347	7393

Table 6
Results for dataset *Amazon UPA 1*.

mpr	Measure	PUCC_R	PUCC_C	CRM	OLF	OLR	OLI	OIF	OIR	OII	ULF	ULR	ULI	UIF	UIR	UII
9	$\|\mathcal{R}\|$	4932	5172	ET	4932	4930	4932	4896	4895	4896	4818	**4816**	4818	4818	4817	4818
	WSC	40,622	42,939	ET	40,622	40,617	40,621	40,539	40,536	40,538	40,404	**40398**	40,404	40,404	40,400	40,404
	$\|\mathcal{U}.\mathcal{A}\|$	25,334	**17742**	ET	25,334	25,339	25,332	25,566	25,565	25,564	26,429	26,427	26,424	26,430	26,429	26,425
	$\|\mathcal{P}.\mathcal{A}\|$	10,356	20,025	ET	10,356	10,348	10,357	10,077	10,076	10,078	9157	9155	9162	9156	**9154**	9161
	time	7127	11,899	ET	7151	7090	6074	**6042**	6183	40,591	40,484	40,472	37,471	37,675	39,471	41,257
18	$\|\mathcal{R}\|$	4840	4773	ET	4840	4840	4840	4807	4807	4807	**4741**	**4741**	**4741**	**4741**	**4741**	**4741**
	WSC	40,438	40,449	ET	40,438	40,438	40,438	40,361	40,361	40,361	**40250**	**40250**	**40250**	**40250**	**40250**	**40250**
	$\|\mathcal{U}.\mathcal{A}\|$	25,238	**16362**	ET	25,238	25,238	25,238	25,473	25,473	25,473	26,347	26,347	26,347	26,348	26,348	26,348
	$\|\mathcal{P}.\mathcal{A}\|$	10,360	**19314**	ET	10,360	10,360	10,360	10,081	10,081	10,081	9162	9162	9162	**9161**	**9161**	**9161**
	time	7223	10,702	ET	7190	7254	**6061**	6075	6178	40,973	41,211	40,728	37,750	38,304	38,050	40,065
27	$\|\mathcal{R}\|$	4834	4738	ET	4834	4834	4834	4801	4801	4801	**4735**	**4735**	**4735**	**4735**	**4735**	**4735**
	WSC	40,426	**40151**	ET	40,426	40,426	40,426	40,349	40,349	40,349	40,238	40,238	40,238	40,238	40,238	40,238
	$\|\mathcal{U}.\mathcal{A}\|$	25,232	**16306**	ET	25,232	25,232	25,232	25,467	25,467	25,467	26,341	26,341	26,341	26,342	26,342	26,342
	$\|\mathcal{P}.\mathcal{A}\|$	10,360	19,107	ET	10,360	10,360	10,360	10,081	10,081	10,081	9162	9162	9162	**9161**	**9161**	**9161**
	time	7278	10,221	ET	7201	7217	**6027**	6125	6090	40,730	40,907	40,722	37,946	38,591	38,278	39,780
36	$\|\mathcal{R}\|$	4833	4735	ET	4833	4833	4833	4800	4800	4800	**4734**	**4734**	**4734**	**4734**	**4734**	**4734**
	WSC	40,424	**40127**	ET	40,424	40,424	40,424	40,347	40,347	40,347	40,236	40,236	40,236	40,236	40,236	40,236
	$\|\mathcal{U}.\mathcal{A}\|$	25,231	**16303**	ET	25,231	25,231	25,231	25,466	25,466	25,466	26,340	26,340	26,340	26,341	26,341	26,341
	$\|\mathcal{P}.\mathcal{A}\|$	10,360	19,089	ET	10,360	10,360	10,360	10,081	10,081	10,081	9162	9162	9162	**9161**	**9161**	**9161**
	time	7286	10,498	ET	7101	7123	6289	6172	**6086**	40,631	40,550	40,912	37,905	38,427	38,413	38,373

Table 7
Varying permissions.

Dataset	*nr*	*nu*	*np*	*mru*	*mpr*	Density
d1	100	2000	100	3	10	0.105
d2	100	2000	500	3	50	0.102
d3	100	2000	1000	3	100	0.098
d4	100	2000	2000	3	200	0.102

In our analysis, we evaluate performance using five key metrics: role-set size, WSC, *Accuracy*, *Similarity*, and execution time (expressed in milliseconds). The metric *Accuracy* quantifies how many of the original, synthetically generated roles our heuristic successfully discovers. If $\mathcal{R}_G$ represents the set of roles from the dataset generator and $\mathcal{R}_M$ is the set of roles identified by a given variant of our heuristic, then *Accuracy* is calculated as the ratio of the intersection of $\mathcal{R}_G$ and $\mathcal{R}_M$ to the size of $\mathcal{R}_G$: that is, $|\mathcal{R}_G \cap \mathcal{R}_M|/|\mathcal{R}_G|$. This approach of measuring the percentage of original roles found is consistent with evaluation methods used in Vaidya et al. (2006, 2010c). The *Similarity* metric, based on the Jaccard index, estimates how closely the mined role set $\mathcal{R}_M$ resembles the original role set $\mathcal{R}_G$. This measure was originally introduced in Molloy et al. (2009) and later used in Frank et al. (2010). For simplicity, and with a slight abuse of notation, we use r to denote both a role and the set of its assigned permissions. Given two roles r_i and r_j, their Jaccard similarity is defined as:

$$Jaccard(r_i, r_j) = \frac{|r_i \cap r_j|}{|r_i \cup r_j|}.$$

The similarity between the role sets $\mathcal{R}_M$ and $\mathcal{R}_G$ is computed by first calculating, for each role in $\mathcal{R}_M$, the highest Jaccard similarity with any role in $\mathcal{R}_G$, and then averaging these maximum values:

$$\mathsf{Sim}_O(\mathcal{R}_M, \mathcal{R}_G) = \frac{\sum_{r_i \in \mathcal{R}_M} \max_{r_j \in \mathcal{R}_G} Jaccard(r_i, r_j)}{|\mathcal{R}_M|}.$$

Following the approach in Benedetti and Mori (2018), we compute the final similarity as the average of this measure in both directions:

$$\mathsf{Sim}(\mathcal{R}_G, \mathcal{R}_M) = \frac{\mathsf{Sim}_O(\mathcal{R}_M, \mathcal{R}_G) + \mathsf{Sim}_O(\mathcal{R}_G, \mathcal{R}_M)}{2}.$$

This similarity score ranges from 0 to 1, where a value of 1 indicates that the mined roles exactly match the original ones (i.e., $\mathcal{R}_M = \mathcal{R}_G$).

To test our heuristics, we ran the randomized dataset generator five times for each set of parameters. We then evaluated all twelve variants of our heuristic against each of these generated datasets. The results presented for a specific parameter set are the average across these five runs. We based our tests on the parameters used in Vaidya et al. (2006), where the relationship $mru \times mpr \approx np/3.33$ was maintained. Three different configurations were considered: Varying Permissions (we kept the number of roles and users constant while changing the number of permissions); Varying Users (we maintained a constant number of roles and permissions and varied the number of users); Varying Roles and Users (we kept the number of permissions constant and varied both the number of roles and users). All three scenarios resulted in sparse UPA matrices with a density of approximately 10 %.

In Table 8, where the best results are highlighted in red, we report experimental results for the datasets generated using the parameters described in Table 7. For the full set of experimental results, please refer to the online resources (Last commit: db6aa7f, 2025). Our experiments consistently showed that variants employing strategy O (selecting permissions from a row in the original user-permission assignment matrix UPA) outperformed those using strategy U (selection based on the uncUPA matrix). Indeed, analyzing the results in Table 8, we observe that using strategy U leads to the generation of a significantly larger number of roles. Specifically, for the parameter set d1 (see Table 7), strategy U produces a role set that is approximately 0.5 times larger than the one generated by the dataset generator. For other parameter sets (e.g., d2

Table 8
Varying permissions.

Dataset	Measure	OLF	OLR	OLI	OIF	OIR	OII	ULF	ULR	ULI	UIF	UIR	UII
d1	$\|\mathcal{R}\|$	**100**	**100**	**100**	**100**	**100**	**100**	158	158	158	150	150	150
	WSC	**6950**	**6950**	**6950**	**6950**	**6950**	**6950**	19,148	19,148	19,148	19,561	19,561	19,561
	accuracy	**0.6**	**0.6**	**0.6**	**0.6**	**0.6**	**0.6**	0.19	0.19	0.19	0.17	0.17	0.17
	similarity	**0.91**	**0.91**	**0.91**	**0.91**	**0.91**	**0.91**	0.44	0.44	0.44	0.42	0.42	0.42
	time	75	74	72	**65**	66	67	513	504	505	478	481	476
d2	$\|\mathcal{R}\|$	**100**	**100**	**100**	**100**	**100**	**100**	995	995	998	995	995	997
	WSC	**6846**	**6846**	**6846**	**6846**	**6846**	**6846**	39,763	39,766	39,773	39,315	39,305	39,321
	accuracy	**0.95**	**0.95**	**0.95**	**0.95**	**0.95**	**0.95**	0.2	0.2	0.2	0.19	0.19	0.19
	similarity	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	0.68	0.68	0.68	0.68	0.68	0.68
	time	119	113	121	137	**108**	109	6989	6975	7070	6733	6781	6731
d3	$\|\mathcal{R}\|$	**100**	**100**	**100**	**100**	**100**	**100**	1287	1286	1288	1301	1300	1299
	WSC	**9237**	**9237**	**9237**	**9237**	**9237**	**9237**	85,616	85,616	85,662	85,314	85,275	85,283
	accuracy	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	0.2	0.2	0.2	0.2	0.2	0.2
	similarity	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	0.75	0.75	0.76	0.75	0.75	0.75
	time	168	171	171	187	**165**	170	17,994	17,964	17,928	17,287	18,002	17,854
d4	$\|\mathcal{R}\|$	**100**	**100**	**100**	**100**	**100**	**100**	1493	1493	1493	1491	1493	1493
	WSC	**13935**	**13935**	**13935**	**13935**	**13935**	**13935**	182,179	182,027	182,200	182,143	182,137	182,213
	accuracy	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	0.16	0.16	0.16	0.15	0.15	0.15
	similarity	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	0.78	0.78	0.79	0.78	0.78	0.79
	time	443	274	269	269	**262**	269	42,632	41,888	42,003	41,066	40,360	40,515

through d4), the size of the mined role sets is between 5 and 18 times larger than the original, randomly generated ones.

This behavior may be attributed to the lack of underlying structure in the initial UPA matrix, where users, roles, and permissions are treated as statistically independent during generation. When strategy U is applied, the process of updating uncUPA may further degrade any latent structure that might exist, leading to less efficient role creation. Moreover, the increased number of roles generated by strategy U, combined with the overhead of updating the *IDF* values of permissions in uncUPA after each role is mined, substantially slows down execution. For example, with parameter set d3, heuristics using strategy O complete in an average of 172 milliseconds, while those using strategy U take approximately 17,838 milliseconds, more than 100 times longer.

When evaluating *Accuracy* and *Similarity*, we again find that variants using strategy O outperform those based on strategy U. For parameter set d1, O-based variants achieve an *Accuracy* of 60 % and a *Similarity* score of 0.91. The results improve further for parameter sets d2, d3, and d4, where *Accuracy* ranges from 95 % to 100 %, indicating that our heuristic is able to recover nearly all original roles. In these cases, the *Similarity* is consistently 1, meaning that the mined roles are identical to the ones used to generate the datasets.

It is worth noting that, for example in parameter set d2, an *Accuracy* of 95 % alongside a *Similarity* of 1 is not a contradiction. This simply indicates that the few unmatched roles are very similar to the originals, and the rounding in the Jaccard-based *Similarity* metric results in a perfect score (e.g., in some of our experiments, the online resources (Last commit: db6aa7f, 2025), the actual *Similarity* associated to an *Accuracy* of 93 % was equal to 0.9973345297778046). The same interpretation holds for the results under parameter set d3.

By contrast, the U-based variants yield significantly worse outcomes, with *Accuracy* scores between 15 % and 20 % and *Similarity* values around 0.43 for d1. For the other parameter sets, *Similarity* values range from 0.68 to 0.75-implying that while the mined roles are not completely different, they deviate from the originals.

A similar trend, where strategy U underperforms compared to O, was observed in Blundo et al. (2022) as well. In that work, two heuristics were proposed to manage multiple constraints simultaneously, specifically limiting both the number of permissions per role and the number of roles assigned to each user.

Finally, to evaluate the scalability of the O-based variants, we tested them on progressively larger datasets. Specifically, we adopted the *Varying Users* scenario, in which the number of roles and permissions was fixed at 300 and 3000, respectively, while the number of users was varied ($nu \in \{10,000, 20,000, 30,000, 50,000, 100,000\}$). For all experiments in this scenario, we set $mru = 20$ and $mpr = 50$. The results are summarized in Table 9.

Across all parameter settings, the results are very encouraging. For dataset d1, the variants generated a role set containing only nine more roles than the original, while for the remaining configurations, the number of generated roles matched the original exactly. Both *Accuracy* and *Similarity* scores are consistently high, *Accuracy* ranges from 92 % to 95 %, and *Similarity* is either exactly 1 or very close to it. For the largest dataset (10,0000 users and 3000 permissions), the fastest variant, OLF, completed in approximately 88 seconds, while the slowest, OLR, took about 116 seconds. It's worth noting that these results were obtained using a Python implementation with minimal code optimization, running on a machine suitable for general workloads but not a high-end system.

5.2. ERUPDC evaluation

In this section, we present a selection of experiments conducted to assess the performance of the new variants of the heuristic ERUPDC introduced in Section 4.2. For this evaluation, we use the eleven benchmark datasets described at the beginning of Section 5. These datasets have been widely adopted in the literature for analyzing the performance of unconstrained role mining algorithms. Using the same datasets allows for a meaningful comparison and helps evaluate the impact of enforcing constraints on the number of roles generated by the proposed approaches. To evaluate the heuristic ERUPDC, we needed to define appropriate values for the parameters $mrcu$ and $mrcp$. For this purpose, we adopted the settings summarized in Table 2 of Harika et al. (2015) and Table 12 of its accompanying Supplemental Material. To further stress-test the heuristic, we also included extreme parameter values. Specifically, for $mrcu$, we used both the minimum and maximum number of permissions assigned to any user (see columns min#P and max#P in Table 3). Similarly, for $mrcp$, we used the minimum and maximum number of users sharing the same permission (see columns min#U and max#U in Table 3). For example, in the *Americas Large* dataset, the tested values for $mrcu$ were $1, 3, 4, 5, 6, 733$, and for $mrcp$ they were $1, 100, 110, 120, 130, 140, 144, 2812$. Some representative results for PRUCC_1 and PRUCC_2 testing are reported in Table 11 and Table 12, respectively.

Table 9
Varying users scenario, $nu \in \{10,000, 20,000, 30,000, 50,000, 100,000\}$.

Dataset	Measure	OLF	OLR	OLI	OIF	OIR	OII
d1	$\|\mathcal{R}\|$	**309**	**309**	**309**	**309**	**309**	**309**
	WSC	**124132**	**124132**	**124132**	124,733	124,733	124,733
	accuracy	**0.93**	**0.93**	**0.93**	0.92	0.92	0.92
	similarity	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**
	time	4728	4433	4378	**4194**	5022	4314
d2	$\|\mathcal{R}\|$	**300**	**300**	**300**	**300**	**300**	**300**
	WSC	**230944**	**230944**	**230944**	**230944**	**230944**	**230944**
	accuracy	**0.94**	**0.94**	**0.94**	**0.94**	**0.94**	**0.94**
	similarity	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	time	10,329	10,729	9956	10,197	**9351**	10,146
d3	$\|\mathcal{R}\|$	**300**	**300**	**300**	**300**	**300**	**300**
	WSC	**339678**	**339678**	**339678**	**339678**	**339678**	**339678**
	accuracy	**0.94**	**0.94**	**0.94**	**0.94**	**0.94**	**0.94**
	similarity	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	time	15,483	17,347	16,666	**14283**	15,371	16,731
d4	$\|\mathcal{R}\|$	**300**	**300**	**300**	**300**	**300**	**300**
	WSC	**558002**	**558002**	**558002**	**558002**	**558002**	**558002**
	accuracy	**0.95**	**0.95**	**0.95**	**0.95**	**0.95**	**0.95**
	similarity	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	time	27,284	26,951	29,129	26,492	28,506	**26381**
d5	$\|\mathcal{R}\|$	**300**	**300**	**300**	**300**	**300**	**300**
	WSC	**1120015**	**1120015**	**1120015**	**1120015**	**1120015**	**1120015**
	accuracy	**0.95**	**0.95**	**0.95**	**0.95**	**0.95**	**0.95**
	similarity	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	time	**88334**	116,861	108,748	115,707	118,840	116,577

As noted in Section 4.2, certain combinations of *mrcu* and *mrcp* may prevent the heuristic from finding a valid decomposition that satisfies both constraints. Two scenarios can arise in such cases. First, the specified constraint values may inherently prevent any valid decomposition of the UPA matrix into UA and PA matrices. Second, while a valid solution may exist, the heuristic may simply fail to find it. To address this, we introduced a *fail-safe* mechanism in our implementation of the ERUPDC heuristic. When the heuristic is unable to organize all user permissions into roles while respecting the constraints, it falls back on assigning the remaining permissions directly to users via a direct user-permission relation $\mathcal{DUPA}$ (Molloy et al., 2008, 2010). Although this deviates from standard RBAC models (Ferraiolo et al., 2001; Sandhu et al., 2000), where permissions are assigned strictly through roles, this approach offers greater flexibility. It accommodates edge cases such as noisy data (Molloy et al., 2009), constraint violations (Kumar et al., 2010), or assignments that do not align with the inferred role structure (Frank et al., 2010). Importantly, the presence of a non-empty $\mathcal{DUPA}$ does not imply that no valid constrained decomposition exists-it only indicates that the heuristic was unable to find one.

The experimental results show that, for several combinations of *mrcu* and *mrcp*, our four variants outperform those originally proposed in Harika et al. (2015). Below, we present a selection of representative results, where the best results are highlighted in red. The full set of experimental outcomes is available in the online supplementary materials (Last commit: db6aa7f, 2025).

If the security policy underlying the RBAC deployment permits direct user-permission assignments, some of our variants produce both a smaller role set and fewer direct assignments. For example, in the Healthcare dataset results shown in Table 10, the FIU and FIP variants improve over NU and NP in these respects. Overall, based on our experimental findings, we conclude that incorporating *IDF*-based strategies into the ERUPDC heuristic can improve its role construction capabilities.

5.3. PRUCC evaluation

In this section, we report a subset of experiments conducted to evaluate the performance of the twelve heuristic variants of PRUCC_1 and PRUCC_2 introduced in Section 4.3. As in the previous evaluations, we rely on the nine benchmark datasets presented at the beginning of Section 5.

To evaluate the PRUCC_1 and PRUCC_2 heuristics, we adopted the parameter values for *mpr* and *mrcu* used in Blundo et al. (2022) and its accompanying supplemental material (Blundo et al., 2020b). In Blundo et al. (2022), the parameters were selected to ensure that valid UA and PA matrices satisfying Eq. (1) always exist. Specifically, the parameters *mpr* and *mrcu* were chosen such that $mpr \cdot mrcu \geq \max\#\text{P}$, where $\max\#\text{P}$ denotes the maximum number of permissions assigned to any user. This condition guarantees that each user can be assigned a set of disjoint roles, each containing at most *mpr* permissions, without exceeding the *mrcu* limit on the number of roles per user. As a result, a feasible solution to the PRUCC problem is always possible under this assumption. The heuristics PRUCC_1 and PRUCC_2 were evaluated using up to five different values for each parameter.

The results for the PRUCC scenario show slightly less improvement compared to those observed in the EURPDC setting. Nonetheless, the experiments indicate that, for some combinations of *mpr* and *mrcu*, our proposed variants either match or outperform those described in Blundo et al. (2022). Below, we present a selection of representative results, with the original variants from Blundo et al. (2022) highlighted in blue and the best-performing outcomes marked in red. The full set of experimental results is available in the online supplementary materials (Last commit: db6aa7f, 2025).

6. Related works

Since the introduction of *role mining* (Kuhlmann et al., 2003a), a number of data mining techniques have been developed with the goal of returning a role set corresponding to the user-permission assignments defined in the input matrix. Different role mining techniques have been defined trying to minimize the number of roles (Vaidya et al., 2007), or the complexity of the role hierarchy structure (Guo et al., 2008), or trying to pursue different criteria (Molloy et al., 2008, 2009; Lu et al., 2008b) for the returned solution. The experimental results in Molloy et al. (2008) showed the effectiveness of the proposed approaches. The experiments used a synthetic dataset created from a template for a uni-

Table 10
Some representative results for ERUPDC testing.

Dataset ($mrcp, mrcu$)	Measure	NU	NP	XR	NR	FIU	FIP	DIU	DIP
Americas Large (130, 5)	$\lvert\mathcal{R}\rvert$	418	419	471	420	434	436	**415**	**415**
	$\lvert\mathcal{UA}\rvert$	3841	3827	4507	3803	3890	3894	3701	**3700**
	$\lvert\mathcal{PA}\rvert$	93,277	92,902	**52295**	95,097	83,102	83,213	93,101	92,945
	$\lvert\mathcal{DUPA}\rvert$	**0**	**0**	4037	**0**	636	636	**0**	**0**
Americas Small (75, 22)	$\lvert\mathcal{R}\rvert$	212	213	201	235	**191**	**191**	206	207
	$\lvert\mathcal{UA}\rvert$	11,330	11,435	4639	4903	**4264**	4265	4282	4308
	$\lvert\mathcal{PA}\rvert$	9661	9447	**5942**	8949	8505	8198	9725	9677
	$\lvert\mathcal{DUPA}\rvert$	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Apj (291, 58)	$\lvert\mathcal{R}\rvert$	456	456	470	476	**454**	**454**	455	456
	$\lvert\mathcal{UA}\rvert$	3128	3064	2630	2613	**2321**	**2321**	2324	2322
	$\lvert\mathcal{PA}\rvert$	2175	2243	2269	**1721**	2414	2414	2441	2448
	$\lvert\mathcal{DUPA}\rvert$	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Domino (5, 7)	$\lvert\mathcal{R}\rvert$	17	**11**	19	21	**11**	**11**	**11**	**11**
	$\lvert\mathcal{UA}\rvert$	166	**27**	138	145	**27**	**27**	**27**	**27**
	$\lvert\mathcal{PA}\rvert$	248	597	**168**	301	597	597	597	597
	$\lvert\mathcal{DUPA}\rvert$	325	109	218	**0**	109	109	109	109
Emea (16, 200)	$\lvert\mathcal{R}\rvert$	**34**	**34**	**34**	59	**34**	**34**	**34**	**34**
	$\lvert\mathcal{UA}\rvert$	**35**	**35**	**35**	179	**35**	**35**	**35**	**35**
	$\lvert\mathcal{PA}\rvert$	7093	7093	7077	**5245**	7093	7093	7093	7093
	$\lvert\mathcal{DUPA}\rvert$	121	121	134	**0**	121	121	121	121
Firewall 1 (26, 21)	$\lvert\mathcal{R}\rvert$	69	69	68	72	**65**	**65**	67	67
	$\lvert\mathcal{UA}\rvert$	1778	1767	1401	**604**	619	619	747	747
	$\lvert\mathcal{PA}\rvert$	3496	3509	2399	**1797**	3551	3551	3761	3761
	$\lvert\mathcal{DUPA}\rvert$	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Healthcare (5, 5)	$\lvert\mathcal{R}\rvert$	10	10	13	15	**9**	**9**	13	**9**
	$\lvert\mathcal{UA}\rvert$	73	73	132	111	**33**	**33**	73	**33**
	$\lvert\mathcal{PA}\rvert$	162	162	**92**	134	168	168	165	168
	$\lvert\mathcal{DUPA}\rvert$	827	827	236	**5**	572	572	483	572

versity RBAC system, as detailed in Stoller et al. (2007). This template defined the roles, permissions, role hierarchy, and the role-permission assignment relationship. The resulting dataset included 493 users and 56 permissions. In Molloy et al. (2009), nine role-mining algorithms were compared using both synthetic and real-world datasets to highlight their strengths and weaknesses. The synthetic datasets were generated from three sources: the Random Data Generator (Vaidya et al., 2006), a Tree-Based Data Generator proposed by the authors, and an ERBAC Data Generator based on Kern et al. (2003). For these datasets, the number of users and permissions was fixed at 500 and 1000, respectively, with other parameters adjusted to achieve a dataset density of 5 % to 10 %. The same synthetic dataset from Stoller et al. (2007) was also used in their experiments. The real-world datasets came from researchers at HP Labs and had been previously used for evaluation in Ene et al. (2008). A complete survey on role mining can be found in Mitra et al. (2016).

Cardinality constraints, often tied to specific organizational contexts and security policies, are used to adjust and refine roles and their assignments to better reflect the organization's access control requirements. By integrating these constraints directly into the mining process, organizations can ensure that the generated roles are aligned with policy goals and practical administrative needs. *Role-usage* constraints, where the restriction is on the number of roles that a user can be assigned, were considered in John et al. (2012), Lu et al. (2013, 2015), Harika et al. (2015) and Blundo et al. (2020a). *Permission-usage* constraints, where the limitation is on the number of permissions included in a role, have been proposed in Kumar et al. (2010), Blundo and Cimato (2013) and Blundo et al. (2020a). *Permission-distribution* constraints limit the number of roles where a permission can be included and have been proposed in Harika et al. (2015) and Blundo et al. (2020a). *Role-distribution* constraints, where there is a limit on the number of users that can be assigned to any given role, were first introduced in Hingankar and Sural (2011). The authors proposed three heuristics relying on a graph modeling approach, where there is a mapping between the role mining problem and the biclique cover for a bipartite graph. The same problem has been further analyzed in Blundo and Cimato (2023), where a variant of the problem and novel heuristics have been proposed. To demonstrate the effectiveness of the proposed heuristics, the aforementioned studies conducted experiments using either the benchmark datasets from HP Labs (Ene et al., 2008) or synthetic datasets, mainly generated with the Random Data Generator (Vaidya et al., 2006).

The application of multiple cardinality constraints has also been explored in the literature. In Ma et al. (2015), a role mining technique was proposed to satisfy both role-distribution and role-cardinality constraints. Role-usage and permission-distribution constraints were addressed in Harika et al. (2015) and Li et al. (2015), while role-usage and permission-usage constraints were examined in Blundo et al. (2017), Wang et al. (2020), Blundo et al. (2022) and Blundo et al. (2023). The experiments in these studies were conducted using either the first nine real-world datasets listed in Table 3 of Section 5 or synthetic datasets generated as described in Section 5.1.1.

Searching for meaningful roles or providing interpretability to the selected roles have become an optimization target for different techniques (Xu and Stoller, 2012). In Kang et al. (2023), Kang et al. provide a formal definition of the interpretable role mining problem in the presence of data noise in order to have meaningful and understandable roles for practical applications. The authors also provide an algorithm which optimizes reconstruction error and role interpretability, analyzing the influence of role number and weighting factor and comparing it with existing algorithms. In Yang et al. (2025), authors proposes an Interpretable Role Mining Algorithm Based on Overlapping Clustering (IRMAOC) that relies on user similarity calculated on the permission and attribute of the role to evaluate interpretability and define a metric for the obtained role set. The technique creates a user association graph based on user-permission matrix and user attribute matrix, and then uses the LFM (Local Fitness Method) algorithm to generate candidate roles based on the graph. In Rao et al. (2021), a role recommendation model (R-RBAC) is presented with the goal of optimizing user-role assignments on the basis of user behaviour patterns. The R-RBAC system

Table 11
Some representative results for PRUCC_1 testing.

Dataset $(mpr, mrcu)$	Measure	*OLF*	*OLR*	OLI	OIF	OIR	OII	*ULF*	*ULR*	ULI	UIF	UIR	UII
Americas Large (2, 367)	$\|\mathcal{R}\|$	5914	8773	5953	5890	8622	5963	5899	8151	5921	5885	8269	**5848**
	WSC	113,521	126,609	114,213	113,301	125,590	114,041	113,092	122,702	113,602	112,590	122,758	**111705**
	time	2485	4923	2670	53,966	68,206	55,132	**2323**	4177	2520	42,059	61,061	42,264
Americas Small (156, 79)	$\|\mathcal{R}\|$	**196**	**196**	**196**	**196**	**196**	**196**	207	207	207	209	209	209
	WSC	11,140	11,156	**11120**	11,164	11,164	11,164	11,644	11,622	11,623	11,660	11,660	11,660
	time	**71**	**71**	74	681	665	715	78	78	82	545	526	547
Customer (14, 12)	$\|\mathcal{R}\|$	501	503	505	**500**	503	504	502	504	505	502	504	505
	WSC	**46,313**	46,347	46,355	46,367	46,390	46,395	46,385	46,405	46,410	46,385	46,404	46,410
	time	**399**	534	544	1229	1274	1096	404	458	**399**	741	1024	868
Domino (2, 157)	$\|\mathcal{R}\|$	134	135	135	134	134	133	134	134	133	134	136	**131**
	WSC	840	844	840	843	843	837	843	845	837	843	848	**828**
	time	**3**	**3**	4	7	7	7	**3**	**3**	4	12	12	12
Firewall 1 (156, 4)	$\|\mathcal{R}\|$	94	95	92	93	94	**91**	95	96	93	95	96	93
	WSC	5644	5801	5330	5623	5780	**5309**	5652	5810	5341	5634	5791	5320
	time	**47**	49	49	177	178	180	49	50	65	141	143	157

Table 12
Some representative results for PRUCC_2 testing.

Dataset $(mpr, mrcu)$	Measure	*OLF*	*OLR*	OLI	OIF	OIR	OII	*ULF*	*ULR*	ULI	UIF	UIR	UII
Americas Small (79, 4)	$\|\mathcal{R}\|$	316	364	324	**314**	366	324	325	371	330	322	362	329
	WSC	16,642	20,371	17,300	**16465**	20,579	17,265	16,758	20,346	17,192	16,914	20,068	17,474
	time	**144**	151	149	370	376	379	151	160	173	381	394	405
Apj (16, 30)	$\|\mathcal{R}\|$	466	466	466	466	**465**	466	467	466	467	467	466	467
	WSC	5189	5187	5191	5174	5175	5174	5141	**5138**	5140	5217	5217	5217
	time	111	**105**	109	350	351	349	**105**	**105**	149	465	465	459
Customer (20, 12)	$\|\mathcal{R}\|$	462	463	464	**461**	**461**	463	462	462	465	462	462	465
	WSC	**46282**	46,310	46,296	46,335	46,335	46,339	46,351	46,351	46,357	46,351	46,351	46,357
	time	**361**	392	426	1007	966	1011	380	376	389	652	669	710
Domino (2, 157)	$\|\mathcal{R}\|$	135	136	135	134	135	133	134	135	133	134	135	**131**
	WSC	840	848	839	843	845	837	843	847	837	843	847	**828**
	time	**3**	**3**	4	7	7	8	**3**	**3**	4	12	13	12
Firewall 1 (156, 4)	$\|\mathcal{R}\|$	97	99	95	95	98	**94**	98	100	96	96	98	95
	WSC	6117	6492	5802	5936	6469	**5779**	6122	6531	5808	5945	6290	5788
	time	**79**	80	80	137	137	134	**79**	80	94	121	125	138

which is based on an Hidden Markov Model, has the task of revoking obsolete assignments and of updating dynamically the roles for each user, examining past or similar behaviour, providing a good approximation to the optimal solution. Authors provided extended experimentation of the proposed model considering both an anonymized access sample dataset provided by Amazon (UC Irvine Machine Learning Repository, 2011) and synthetically generated data, reporting good results as regards speed and size of the generated solution.

Some other variants of the basic RMP problem have been identified, when the original UPA allows for some inexactness, that can be caused by mistakes or other failures. Vaidya et al. (2007) have proposed the δ-consistent RMP, meaning that any UPA divergent within δ from the original UPA satisfies the definition. In Vaidya et al. (2010b), the Minimal Noise Role Mining Problem (Min-Noise RMP) is introduced, where noise refers to a permission being recorded as a denial or vice-versa. The goal is to fix the number of roles that one would like to find and looking for roles that have minimal difference with respect to the original UPA.

Related problems have also been defined, such as the Optimal Recruitment Problem (ORP), where the goal is to select the minimum number of new employees from a set of candidates to fill the vacant positions created by retired employees. The authors provide a formal proof for the complexity of the problem, showing that the ORP problem is NP-hard, and propose a greedy heuristic (Roy et al., 2021). In Roy et al. (2021), extensive experiments were conducted on synthetic datasets randomly generated for each combination of the ORP parameters. The results of these experiments successfully demonstrated the effectiveness and efficiency of the proposed solution.

Several real-world case studies have demonstrated the practical application and effectiveness of role mining in diverse organizational contexts. In Kuhlmann et al. (2003b) the authors presented a new methodology for role-engineering named *SAM Role Miner*. They described a way to find roles in large enterprises using data mining technology to extract knowledge contained in existing access rights. This work was done to support projects migrating to RBAC within the framework of the Security Administration Manager (SAM) software product. *SAM Role Miner* was evaluated through two large-scale deployments. The first case study was conducted in one of the top five insurance companies in Germany, employing approximately 140,000 people, where the tool was used to extract business roles from extensive access control data. The second study focused on a mid-sized organization with 10,000 users and about 400 roles, enabling the authors to assess the tool's adaptability across different organizational scales. To address the challenges posed by noisy and inconsistent access control data, Huang et al. (2009) proposed a noise-preprocessing method. The authors highlighted the importance of cleaning legacy user-permission data before applying role mining algorithms to improve the quality of the mined roles. To demonstrate the efficacy of the proposed method, the authors performed case studies in industrial setting with realistic systems. The method was applied to four systems within `SS Corporation`, a global corporation that provides financial services in IT, whose IT development and support is distributed worldwide. For confidentiality, the name and description of the analyzed system were changed a bit. However, access control data were based on real systems. The largest system had more than 3400 user and over 1020 permissions. Role mining from a visualization perspective was explored in Colantonio et al. (2012) where the authors conducted a case study on a large private company. Their visual role mining procedure demonstrated how graphical representations could help analysts identify meaningful role structures more intuitively, supporting both role discovery and refinement. Their methodology comes from a real case study that has been carried out in a large private company. To protect company privacy, they did not reveal any details of the results. These case studies collectively illustrate how role mining has been successfully applied in organizations of varying sizes and sectors, from large multinational corporations to medium-sized enterprises, and how complementary techniques, such as noise preprocessing and visual analytics, can enhance the effectiveness of role mining in practice.

7. Conclusions

Role mining is an essential technique for implementing and maintaining Role-Based Access Control (RBAC) systems in complex organizations. Rather than relying exclusively on manual engineering of roles from job functions, role mining automates the process by analyzing existing user-permission assignments to discover roles (i.e., a *candidate role set*) (Vaidya et al., 2007; Mitra et al., 2016). Despite its benefits, practical implementations face several challenges:

- *Noisy Data*: Flawed or outdated permissions result in the derivation of incorrect roles (Huang et al., 2009; Kang et al., 2023).
- *Role Explosion*: Mining too many roles makes the RBAC system difficult to manage (Colantonio et al., 2012).
- *Lack of Semantic Meaning*: The automatically generated roles may not align with an organization's business functions (Kuhlmann et al., 2003a; Frank et al., 2008; Molloy et al., 2008).
- *Dynamic Environments*: Frequent changes in user-permission assignments necessitate continuous re-mining of roles (Nobi et al., 2022; Tyagi et al., 2024; Nobi et al., 2025; Anderer et al., 2023).

Integrating role mining into existing RBAC management tools enhances its practical value. In this setup, mining can be performed offline on current user-permission data to generate candidate roles, which are then refined and deployed within governance platforms such as Oracle IAM (Identity and Access Management), SAP GRC (Governance, Risk, and Compliance), and IBM SVG (Security Verify Governance). Integration is typically achieved through import/export pipelines, plugins, admin dashboards, and automated monitoring. Modern identity governance solutions are already beginning to include these features.

Given the dynamic nature of organizations, role mining can be run periodically to: Identify deviations, such as excessive direct user-permission assignments; Propose merging or splitting roles, or revoking outdated ones to prevent privilege creep. This process requires integration with the audit logs and access review workflows of RBAC tools (Anderer et al., 2023); for example, IBM SVG already allows periodic role discovery and reconciliation with existing policies. From a research perspective, organizational dynamism remains a key driver of innovation.

This work does not primarily aim to introduce entirely new heuristics for constraint enforcement. Instead, our goal is to evaluate whether incorporating *IDF* values into existing heuristics can improve the role construction process. In fact, in this paper, we introduce a novel role mining approach leveraging the *bag-of-words* (BoW) model, a technique borrowed from text processing. Roles are treated as "documents" and permissions as "words", enabling the application of text-based metrics to role mining. Specifically, we employ the *IDF* value (Inverse Document Frequency value) to identify significant permissions and roles for constructing Role-Based Access Control (RBAC) models. This IDF criterion is integrated into a framework designed to guide heuristics for selecting users and permissions under: the Permission-Usage Cardinality Constraint (PUCC) (Kumar et al., 2010; Blundo et al., 2020a), which limits the maximum number of permissions per role; the Role Usage and Permission Distribution Constraints (RUPDC) (Harika et al., 2015), that simultaneously limits the number of roles that can be assigned to any single user and the number of roles a single permission can appear in; and the Permission-Role-Usage Cardinality Constraints Problem (PRUCC) (Blundo et al., 2022), that restricts both the number of roles that can be assigned to any single user and the number of permissions that can be assigned to each role.

In the PUCC scenario, we propose several variants derived from a common base procedure that leverage *IDF*-based information. These variants differ in two key dimensions: the strategy for user selection, either based on the number of uncovered permissions or on the aggregate

IDF value of uncovered permissions, and the strategy for role formation, where permissions are chosen according to one of three policies (first encountered, randomly sampled, or those with the lowest *IDF* values), subject to the PUCC constraint. The proposed technique is evaluated on eleven benchmark datasets, with results compared against state-of-the-art heuristics such as PUCC_R, PUCC_C, and CRM (see, Blundo et al., 2020a and Kumar et al., 2010). The comparison focuses on metrics like the number of roles, user-to-role assignments, role-to-permission assignments, and the Weighted Structural Complexity (WSC). The scalability of our twelve variants was assessed using datasets generated by a synthetic data generator proposed in Vaidya et al. (2006).

Experimental results on the *Americas large* dataset demonstrate that the IDF-based approach can yield a lower minimum number of roles compared to other heuristics in certain scenarios. The effectiveness of user and permission selection strategies (length-based vs. *IDF*-based) varies based on the PUCC's maximum permission limit. The findings suggest that the *IDF* value provides a distinct exploration of the solution space, which could lead to improved or alternative solutions for constrained role mining problems. For the ERUPDC scenario, we extended the four heuristic variants introduced in Harika et al. (2015) by incorporating *IDF*-based strategies, resulting in four additional variants. Experimental results demonstrate that, under several combinations of constraints on the number of roles per user and the number of role permissions can appear in, these *IDF*-enhanced variants consistently outperform the original ones proposed in Harika et al. (2015).

For the PRUCC scenario, the improvements are less pronounced than those observed in the ERUPDC setting. Nevertheless, the experiments show that, for certain combinations of constraints on the number of roles per user and the number of permissions per role, our proposed variants perform on par with, or better than, those reported in Blundo et al. (2022).

In conclusion, role mining can be conceptualized as a *support module* within RBAC management frameworks: not only for initial role discovery, but also for refinement and long-term adaptability. Research directions include developing meta-heuristics for selecting role-mining strategies based on system characteristics, leveraging machine learning for semantic alignment of roles, and embedding mining algorithms into continuous governance pipelines. Recent advances suggest that modern identity governance solutions are progressively incorporating these features, but open challenges remain in scalability, adaptability, and explainability.

CRediT authorship contribution statement

Carlo Blundo: Writing – review & editing, Writing – original draft, Validation, Supervision, Software, Methodology, Investigation, Data curation, Conceptualization; **Stelvio Cimato:** Writing – review & editing, Writing – original draft, Validation, Supervision, Software, Methodology, Investigation, Data curation, Conceptualization.

Data availability

Github repositories link is included in the submission.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Stelvio Cimato reports financial support was provided by NGEU. Stelvio Cimato reports a relationship with European Union that includes: funding grants. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

We're grateful to the anonymous reviewers for their time and effort in evaluating our manuscript. Their insightful suggestions and constructive comments significantly improved the quality of the article.

This work was supported in part by project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU. However, the views and opinions expressed are those of the authors alone and do not necessarily reflect those of the European Union or the Italian MUR. Neither the European Union nor the Italian MUR can be held responsible for them.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.cose.2025.104808

References

Aizawa, A., 2000. The feature quantity: an information theoretic perspective of tfidf-like measures. In: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 104–111.

Aizawa, A., 2003. An information theoretic perspective of TF-IDF measures. Inf. Process. Manag. 39, 45–65. https://doi.org/10.1016/S0306-4573(02)00021-3

Anderer, S., Kempter, T., Scheuermann, B., Mostaghim, S., 2023. Dynamic optimization of role concepts for role-based access control using evolutionary algorithms. SN Comput. Sci. 4 (4), 416. https://doi.org/10.1007/S42979-023-01805-1

Anderer, S., Schrader, F., Scheuermann, B., Mostaghim, S., 2025. Mining two-level role concepts using evolutionary algorithms. SN Comput. Sci. 6 (6), 563. https://doi.org/10.1007/S42979-025-04046-6

Benedetti, M., Mori, M., 2018. Parametric RBAC maintenance via max-sat. In: Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies, SACMAT 2018. ACM, New York, Indianapolis, IN, USA, pp. 15–25.

Blundo, C., Cimato, S., 2013. Constrained role mining. In: STM 2012, 8th International Workshop on Security and Trust Management, Revised Selected Papers. Springer, pp. 289–304. https://doi.org/10.1007/978-3-642-38004-4_19

Blundo, C., Cimato, S., 2023. Role mining under user-distribution cardinality constraint. J. Inf. Secur. Appl. 78, 103611. https://www.sciencedirect.com/science/article/pii/S2214212623001953. https://doi.org/10.1016/j.jisa.2023.103611

Blundo, C., Cimato, S., Siniscalchi, L., 2017. PRUCC-RM: permission-role-usage cardinality constrained role mining. In: 41st IEEE Annual Computer Software and Applications Conference, COMPSAC 2017, Turin, Italy, July 4–8, 2017. Volume 2, pp. 149–154. https://doi.org/10.1109/COMPSAC.2017.195

Blundo, C., Cimato, S., Siniscalchi, L., 2020a. Managing constraints in role based access control. IEEE Access 8, 140497–140511. https://doi.org/10.1109/ACCESS.2020.3011310

Blundo, C., Cimato, S., Siniscalchi, L., 2020b. Supplemental material for: role mining heuristics for permission-role-usage cardinality constraints. https://github.com/RoleMining/ConstrainedRM.

Blundo, C., Cimato, S., Siniscalchi, L., 2022. Role mining heuristics for permission-role-usage cardinality constraints. Comput. J. 65 (6), 1386–1411. https://doi.org/10.1093/comjnl/bxaa186

Blundo, C., Cimato, S., Siniscalchi, L., 2023. Heuristics for constrained role mining in the post-processing framework. J. Ambient Intell. Humaniz. Comput. 14 (8), 9925–9937. https://doi.org/10.1007/s12652-021-03648-1

Chen, L., Crampton, J., 2009. Set covering problems in role-based access control. In: Backes, M., Ning, P. (Eds.), ESORICS 2009, Proceedings of 14th European Symposium on Research in Computer Security. Springer-Verlag, pp. 689–704.

Colantonio, A., Pietro, R.D., Ocello, A., Verde, N.V., 2012. Visual role mining: a picture is worth a thousand roles. IEEE Trans. Knowl. Data Eng. 24 (6), 1120–1133. https://doi.org/10.1109/TKDE.2011.37

Coyne, E.J., 1995. Role engineering. In: Youman, C.E., Sandhu, R.S., Coyne, E.J. (Eds.), Proceedings of the First ACM Workshop on Role-Based Access Control, RBAC 1995, Gaithersburg, MD, USA, November 30–December 2, 1995. ACM, pp. 54–57. https://doi.org/10.1145/270152.270159

Ene, A., Horne, W.G., Milosavljevic, N., Rao, P., Schreiber, R., Tarjan, R.E., 2008. Fast exact and heuristic methods for role minimization problems. In: Ray, I., Li, N. (Eds.), 13th ACM Symposium on Access Control Models and Technologies, SACMAT 2008, Estes Park, CO, USA, June 11–13, 2008, Proceedings. ACM, pp. 1–10.

Ferraiolo, D.F., Sandhu, R.S., Gavrila, S.I., Kuhn, D.R., Chandramouli, R., 2001. Proposed NIST standard for role-based access control. ACM Trans. Inf. Syst. Secur. 4 (3), 224–274.

Frank, M., Basin, D.A., Buhmann, J.M., 2008. A class of probabilistic models for role engineering. In: ACM Conference on Computer and Communications Security. ACM, pp. 299–310.

Frank, M., Buhmann, J.M., Basin, D.A., 2010. On the definition of role mining. In: 15th ACM Symposium on Access Control Models and Technologies, SACMAT 2010, Proceedings. ACM, Pittsburgh, Pennsylvania, USA, pp. 35–44.

Garey, M.R., Johnson, D.S., 1979. Computers and intractability, A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, New York.

Guo, Q., Vaidya, J., Atluri, V., 2008. The role hierarchy mining problem: discovery of optimal role hierarchies. In: Computer Security Applications Conference, 2008. ACSAC 2008. Annual, pp. 237–246. https://doi.org/10.1109/ACSAC.2008.38

Hamner, B., kenmonta, Cukierski, W., 2013. Amazon.com – Employee Access Challenge. https://www.kaggle.com/competitions/amazon-employee-access-challenge/.

Harika, P., Nagajyothi, M., John, J.C., Sural, S., Vaidya, J., Atluri, V., 2015. Meeting cardinality constraints in role mining. IEEE Trans. Dependable Sec. Comput. 12 (1), 71–84. https://doi.org/10.1109/TDSC.2014.2309117

Hingankar, M., Sural, S., 2011. Towards role mining with restricted user-role assignment. In: Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology (Wireless VITAE), 2011 2nd International Conference on, pp. 1–5. https://doi.org/10.1109/WIRELESSVITAE.2011.5940855

Huang, C., Sun, J., Wang, X., Si, Y., Wu, D., 2009. Preprocessing the noise in legacy user permission assignment data for role mining - an industrial practice. In: 25th IEEE International Conference on Software Maintenance (ICSM 2009), September 20–26, 2009, Edmonton, Alberta, Canada. IEEE Computer Society, pp. 403–406. https://doi.org/10.1109/ICSM.2009.5306288

John, J.C., Sural, S., Atluri, V., Vaidya, J.S., 2012. Role mining under role-usage cardinality constraint. In: Gritzalis, D., Furnell, S., Theoharidou, M. (Eds.), Information Security and Privacy Research. Springer Berlin Heidelberg. Vol. 376 of *IFIP Advances in Information and Communication Technology*, pp. 150–161. https://doi.org/10.1007/978-3-642-30436-1_13

Kang, H., Liu, G., Wang, Q., Zhang, Q., Niu, J., Luo, N., 2023. An improved minimal noise role mining algorithm based on role interpretability. Comput. Secur. 127, 103100. https://www.sciencedirect.com/science/article/pii/S016740482300010X. https://doi.org/10.1016/j.cose.2023.103100

Kern, A., Schaad, A., Moffett, J.D., 2003. An administration concept for the enterprise role-based access control model. In: Ferrari, E., Ferraiolo, D.F. (Eds.), 8th ACM Symposium on Access Control Models and Technologies, SACMAT 2003, Villa Gallia, Como, Italy, June 2–3, 2003, Proceedings. ACM, pp. 3–11. https://doi.org/10.1145/775412.775414

Kuhlmann, M., Shohat, D., Schimpf, G., 2003a. Role mining - revealing business roles for security administration using data mining technology. In: Ferrari, E., Ferraiolo, D.F. (Eds.), 8th ACM Symposium on Access Control Models and Technologies, SACMAT 2003, Villa Gallia, Como, Italy, June 2–3, 2003, Proceedings. ACM, pp. 179–186.

Kuhlmann, M., Shohat, D., Schimpf, G., 2003b. Role mining - revealing business roles for security administration using data mining technology. In: Ferrari, E., Ferraiolo, D.F. (Eds.), 8th ACM Symposium on Access Control Models and Technologies, SACMAT 2003, Villa Gallia, Como, Italy, June –3, 2003, Proceedings. ACM, pp. 179–186. https://doi.org/10.1145/775412.775435

Kumar, R., Sural, S., Gupta, A., 2010. Mining RBAC roles under cardinality constraint. In: Proceedings of the 6th International Conference on Information Systems Security. Springer-Verlag, Berlin, Heidelberg, pp. 171–185. http://portal.acm.org/citation.cfm?id=1940366.1940383.

Last commit: db6aa7f, 2025. Bag of Words Model for Role Mining. https://github.com/carblu/rm-idf. GPL-3.0 license.

Li, N., Molloy, I., Wang, Q., Bertino, E., Calo, S., Lobo, J., 2007. Role Mining for Engineering and Optimizing Role Based Access Control Systems. Technical Report. Purdue University.

Li, R., Li, H., Gu, X., Li, Y., Ye, W., Ma, X., 2015. Role mining based on cardinality constraints. Concurr. Comput. Pract. Exper. 27 (12), 3126–3144. https://doi.org/10.1002/cpe.3456

Lu, H., Hong, Y., Yang, Y., Duan, L., Badar, N., 2013. Towards user-oriented RBAC model. In: Data and Applications Security and Privacy XXVII - 27th Annual IFIP WG 11.3 Conference, DBSec 2013. Proceedings. Springer, Newark, NJ, USA, pp. 81–96.

Lu, H., Hong, Y., Yang, Y., Duan, L., Badar, N., 2015. Towards user-oriented RBAC model. J. Comput. Secur. 23 (1), 107–129. https://doi.org/10.3233/JCS-140519

Lu, H., Vaidya, J., Atluri, V., 2008a. Optimal boolean matrix decomposition: application to role engineering. In: Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7–12, 2008, CancÚN, MÉXico, pp. 297–306. https://doi.org/10.1109/ICDE.2008.4497438

Lu, H., Vaidya, J., Atluri, V., 2008b. Optimal boolean matrix decomposition: application to role engineering. In: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering. IEEE Computer Society, Washington, DC, USA, pp. 297–306. https://doi.org/10.1109/ICDE.2008.4497438

Ma, X., Li, R., Wang, H., Li, H., 2015. Role mining based on permission cardinality constraint and user cardinality constraint. Sec. Commun. Netw. 8 (13), 2317–2328. https://doi.org/10.1002/sec.1177

Mitra, B., Sural, S., Vaidya, J., Atluri, V., 2016. A survey of role mining. ACM Comput. Surv. 48 (4), 50:1–50:37. https://doi.org/10.1145/2871148

Molloy, I., Chen, H., Li, T., Wang, Q., Li, N., Bertino, E., Calo, S.B., Lobo, J., 2008. Mining roles with semantic meanings. In: Ray, I., Li, N. (Eds.), 13th ACM Symposium on Access Control Models and Technologies, SACMAT 2008, Estes Park, CO, USA, June 11–13, 2008, Proceedings. ACM, pp. 21–30.

Molloy, I., Chen, H., Li, T., Wang, Q., Li, N., Bertino, E., Calo, S.B., Lobo, J., 2010. Mining roles with multiple objectives. ACM Trans. Inf. Syst. Secur. 13 (4), 36:1–36:35.

Molloy, I., Li, N., Li, T., Mao, Z., Wang, Q., Lobo, J., 2009. Evaluating role mining algorithms. In: Carminati, B., Joshi, J. (Eds.), 14th ACM Symposium on Access Control Models and Technologies, SACMAT 2009, Stresa, Italy, June 3–5, 2009, Proceedings. ACM, pp. 95–104.

Nobi, M.N., Gupta, M., Krishnan, R., Rana, M.S., Praharaj, L., Abdelsalam, M., 2025. Machine learning in access control: a taxonomy [systematization of knowledge paper]. In: Stoller, S.D., Chowdhury, O., Lee, A.J., Masoumzadeh, A. (Eds.), Proceedings of the 30th ACM Symposium on Access Control Models and Technologies, SACMAT 2025, Stony Brook, NY, USA, 10 July 2025. ACM, pp. 145–156. https://doi.org/10.1145/3734436.3734453

Nobi, M.N., Krishnan, R., Huang, Y., Shakarami, M., Sandhu, R.S., 2021. DlbacAlpha. https://github.com/dlbac/DlbacAlpha.

Nobi, M.N., Krishnan, R., Huang, Y., Shakarami, M., Sandhu, R.S., 2022. Toward deep learning based access control. In: Joshi, A., Fernández, M., Verma, R.M. (Eds.), CODASPY '22: Twelveth ACM Conference on Data and Application Security and Privacy, Baltimore, MD, USA, April 24–27, 2022. ACM, pp. 143–154. https://doi.org/10.1145/3508398.3511497

Rao, K.R., Nayak, A., Ray, I.G., Rahulamathavan, Y., Rajarajan, M., 2021. Role recommender-RBAC: optimizing user-role assignments in RBAC. Comput. Commun. 166, 140–153.

Robertson, S., 2004. Understanding inverse document frequency: on theoretical arguments for IDF. J. Documen. 60 (5), 503–520.

Roy, A., Sural, S., Majumdar, A.K., Vaidya, J., Atluri, V., 2021. Optimal employee recruitment in organizations under attribute-based access control. ACM Trans. Manage. Inf. Syst. 12 (1). https://doi.org/10.1145/3403950

Sandhu, R., Ferraiolo, D., Kuhn, R., 2000. The NIST model for role-based access control: towards a unified standard. In: Proceedings of the Fifth ACM Workshop on Role-based Access Control. ACM, New York, NY, USA, pp. 47–63. https://doi.org/10.1145/344287.344301

Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E., 1996. Role-based access control models. Computer 29 (2), 38–47. https://doi.org/10.1109/2.485845

Schlegelmilch, J., Steffens, U., 2005. Role mining with ORCA. In: Ferrari, E., Ahn, G. (Eds.), 10th ACM Symposium on Access Control Models and Technologies, SACMAT 2005, Stockholm, Sweden, June 1–3, 2005, Proceedings. ACM, pp. 168–176.

Stockmeyer, L.J., 1975. The Minimal Set Basis Problem is NP-Complete. Technical Report RC 5431. IBM Research.

Stoller, S.D., Yang, P., Ramakrishnan, C.R., Gofman, M.I., 2007. Efficient policy analysis for administrative role based access control. In: Ning, P., De Capitani di Vimercati, S., Syverson, P.F. (Eds.), Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28–31, 2007. ACM, pp. 445–455. https://doi.org/10.1145/1315245.1315300

Trnecka, M., Trneckova, M., 2020. An incremental algorithm for the role mining problem. Comput. Secur. 94, 101830. https://doi.org/10.1016/J.COSE.2020.101830

Tyagi, S., Prasad, Y., Jinwala, D.C., Bhattacharjee, S., 2024. A fast access control method in iot using XGB. SN Comput. Sci. 5 (8), 1084. https://doi.org/10.1007/S42979-024-03467-Z

UC Irvine Machine Learning Repository, 2011. Amazon access samples data set. https://archive.ics.uci.edu/ml/datasets/Amazon+Access+Samples.

Vaidya, J., Atluri, V., Guo, Q., 2007. The role mining problem: finding a minimal descriptive set of roles. In: Lotz, V., Thuraisingham, B.M. (Eds.), 12th ACM Symposium on Access Control Models and Technologies, SACMAT 2007, Sophia Antipolis, France, June 20–22, 2007, Proceedings. ACM, pp. 175–184.

Vaidya, J., Atluri, V., Guo, Q., 2010a. The role mining problem: a formal perspective. ACM Trans. Inf. Syst. Secur. 13 (3).

Vaidya, J., Atluri, V., Guo, Q., Lu, H., 2010b. Role mining in the presence of noise. pp. 97–112. https://doi.org/10.1007/978-3-642-13739-6_7

Vaidya, J., Atluri, V., Warner, J., 2006. Roleminer: mining roles using subset enumeration. In: CCS '06. ACM, pp. 144–153. https://doi.org/http://dx.doi.org/10.1145/1180405.1180424

Vaidya, J., Atluri, V., Warner, J., Guo, Q., 2010c. Role engineering via prioritized subset enumeration. IEEE Trans. Dependable Sec. Comput. 7 (3), 300–314. https://doi.org/10.1109/TDSC.2008.61

Wang, J., Dong, J., Tan, Y., 2020. Role mining algorithms satisfied the permission cardinality constraint. Int. J. Netw. Secur. 22 (3), 373–382. http://ijns.jalaxy.com.tw/contents/ijns-v22-n3/ijns-2020-v22-n3-p373-382.pdf.

Xu, Z., Stoller, S.D., 2012. Algorithms for mining meaningful roles. In: Proceedings of the 17th ACM Symposium on Access Control Models and Technologies, pp. 57–66.

Yang, Y., Li, J., Zhang, T., Chen, L., Huang, G., Lv, Z., 2025. IRMAOC: an interpretable role mining algorithm based on overlapping clustering. Cybersecur. 8 (1), 54. https://doi.org/10.1186/S42400-024-00348-Z

Zhang, D., Ramamohanarao, K., Ebringer, T., 2007. Role engineering using graph optimisation. In: Lotz, V., Thuraisingham, B.M. (Eds.), 12th ACM Symposium on Access Control Models and Technologies, SACMAT 2007, Sophia Antipolis, France, June 20–22, 2007, Proceedings. ACM, pp. 139–144.