

Article

MFC-RMA (Matrix Factorization and Constraints- Role Mining Algorithm): An Optimized Role Mining Algorithm

Fubao Zhu ¹, Chenguang Yang ¹, Liang Zhu ¹ , Hongqiang Zuo ² and Jingzhong Gu ^{1,2,*}

¹ College of Computer Science and Technology, Zhengzhou University of Light Industry, Zhengzhou 450000, China; fbzhu@zzuli.edu.cn (F.Z.); 18839179007@163.com (C.Y.); lzhu@zzuli.edu.cn (L.Z.)

² Shangu Cyber Security Technology Co., Ltd., Zhengzhou 450000, China; zuohq2023@163.com

* Correspondence: 13903868786@139.com or shangu@ishangu.com

Abstract: Role-based access control (RBAC) is a widely adopted access control model in various domains for defining security management. Role mining is closely related to role-based access control, as the latter employs role assignments to offer a flexible and scalable approach to managing permissions within an organization. The edge role mining problem (Edge RMP), a variant of the role mining problem (RMP), has long been recognized as an effective strategy for role assignment. Role mining, which groups users with similar access permissions into the same role, bears some resemblance to symmetry. Symmetry categorizes objects or graphics with identical characteristics into one group. Both involve a certain form of “classification” or “induction”. Edge-RMP reduces the associations between users and permissions, thereby lowering the security risks faced by the system. While an algorithm based on Boolean matrix factorization exists for this problem, it fails to further refine the resulting user–role assignment (UA) and role–permission assignment (PA) relationships. Additionally, this algorithm does not address constraint-related issues, such as cardinality constraints, user exclusion constraints, and user capabilities. Furthermore, it demonstrates significant redundancy of roles when handling large datasets, leaving room for further optimization of Edge-RMP results. To address these concerns, this paper proposes the MFC-RMA algorithm based on Boolean matrix factorization. The method achieves significant optimization of Edge-RMP results by handling relationships between roles possessing various permissions. Furthermore, this paper clusters, compresses, modifies, and optimizes the original data based on the similarity between users, ensuring its usability for role mining. Both theoretical and practical considerations are taken into account for different types of constraints, and algorithms are devised to reallocate roles incorporating these constraints, thereby generating UA and PA matrices. The proposed approach yields optimal numbers of generated roles and the sum of the minimum number of generated edges to address the aforementioned issues. Experimental results demonstrate that the algorithm reduces management overhead, provides efficient execution results, and ensures the accuracy of generated roles.

Keywords: role mining; Boolean matrix factorization; Edge RMP; constraints; clustering; compression



Citation: Zhu, F.; Yang, C.; Zhu, L.; Zuo, H.; Gu, J. MFC-RMA (Matrix Factorization and Constraints- Role Mining Algorithm): An Optimized Role Mining Algorithm. *Symmetry* **2024**, *16*, 1008. <https://doi.org/10.3390/sym16081008>

Academic Editor: Sergei D. Odintsov

Received: 7 April 2024

Revised: 26 May 2024

Accepted: 4 June 2024

Published: 7 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

With the swift evolution and widespread adoption of network information technology, there has arisen an increasing demand for significant storage and exchange of information within expansive and intricate management systems [1]. Over the past three decades, more and more enterprises and organizations are opting for RBAC as their principal access control method, as it enhances the flexibility and manageability of security management [2]. With the successful implementation of RBAC systems, the precise formulation of effective role sets and the construction of well-designed RBAC systems that meet practical application requirements have become critical tasks. Bottom-up role engineering techniques aim to transition from non-RBAC systems to RBAC systems [3]. In RBAC, roles signify functional positions within an organization, and users are assigned suitable roles according

to their qualifications. This approach streamlines security management by eliminating the necessity for frequent alterations in fundamental security configurations while ensuring that roles consistently mirror the organization's security policies. RBAC has evolved into a prevalent component of commercial systems, furnishing organizations with more potent security management tools. Meanwhile, it designs the structure and function of the system in combination with the principle of symmetry, and the integration of the two can more effectively protect system security and optimize user experience. This approach significantly mitigates the intricacies associated with directly managing individual user permissions, particularly in large-scale organizations, thereby drastically minimizing the occurrence of errors and oversights. In this article, we discuss role engineering among them.

In the implementation of RBAC, the initial stage entails identifying the necessary roles, followed by executing user-role assignment (UA) and role-permission assignment (PA). Yet, the difficulty lies in establishing a comprehensive, accurate, and efficient set of roles. This process is referred to as role engineering [4]. At its core, role engineering is a process aimed at defining roles and assigning relevant permissions. In the implementation of the RBAC model, role engineering assumes critical importance. Its primary objective is to establish a comprehensive, accurate, and efficient set of roles while associating the appropriate permissions with these roles. Particularly in large-scale systems, transitioning from traditional access control lists (ACLs) to RBAC can incur significant conversion costs, presenting challenges to RBAC adoption. To mitigate this challenge, role mining algorithms, as a specific role engineering approach, help reduce the cost of role definition by automatically generating roles. Therefore, the design of efficient role mining algorithms has become particularly important.

Typically, role mining algorithms aim to discover an RBAC policy that aligns with a given access control list while minimizing policy complexity. The intricacy can be assessed through various factors, including the quantity of roles, relationships between users and roles, and relationships between roles and permissions. The primary goal of these algorithms is to streamline permission management and refine RBAC policies for greater efficiency.

However, during the role mining process, significant challenges arise, particularly in large-scale data systems. These systems often exhibit high complexity and redundancy, necessitating improvements in the efficiency and maintainability of permission management. This challenge is particularly pronounced in large organizations and systems, where permission assignments can become intricate and extensive. With numerous connections between permissions and users, systems face difficulties in effectively controlling and monitoring access to critical resources, thereby increasing the risk of security threats.

In response to some of the above-mentioned issues, certain individuals have preliminarily proposed specific concepts and methods. Vaidya et al. [5] first formally defined and analyzed the role mining problem. Under the assumption that user permissions can be represented as a binary matrix, the basic role mining problem (basic RMP) can be defined as follows: given a $m \times n$ binary matrix UPA , where m represents the number of users, and n represents the number of permissions, this matrix represents user permission assignments. The task of basic RMP is to decompose the UPA matrix into two matrices UA , which is a $m \times k$ matrix representing user–role relationships and PA , which is a $k \times n$ matrix representing role–permission relationships. Here, k represents the number of roles derived by the algorithm based on UPA , and it is essential to ensure that k is minimized. In summary, given the user permission assignments (UPA), the goal of the role mining problem is to find a way to associate users with roles (UA) and roles with permissions (PA) that minimizes the required number of roles.

Vaidya and his colleagues [5,6] have separately proven that both Basic RMP and Edge RMP are NP-complete problems. Since UPA depicts the associations between users and permissions, it is logical to represent it using a bipartite graph, as depicted in Figure 1a. In this bipartite graph, users and permissions are depicted as two sets of vertices, with each edge indicating a user–permission assignment. Figure 1b illustrates the decomposition

of UPA into a tripartite graph, demonstrating the connections between users, roles, and permissions based on the user-permission assignments. The objective of Basic RMP is to identify a decomposition of user permissions that minimizes the required number of roles. Figure 1b presents a role mining outcome with a role set of {R1, R2, R3}.

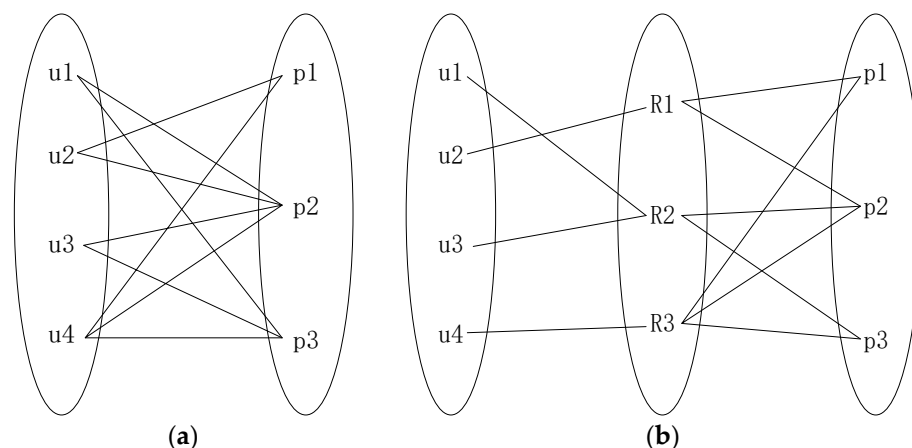


Figure 1. Illustration of the basic role mining process: (a) original user-permission assignment (UPA) matrix and (b) decomposed user-role (UA) and role-permission (PA) matrices.

An alternative approach involves employing role discovery algorithms to reduce the overall count of user-to-role and role-to-permission assignments ($|UA| + |PA|$), as illustrated in Figure 2. From an administrative standpoint, this may be more practically meaningful as it reduces the number of assignments requiring management. This is termed the minimum edge role mining problem, or Edge RMP.

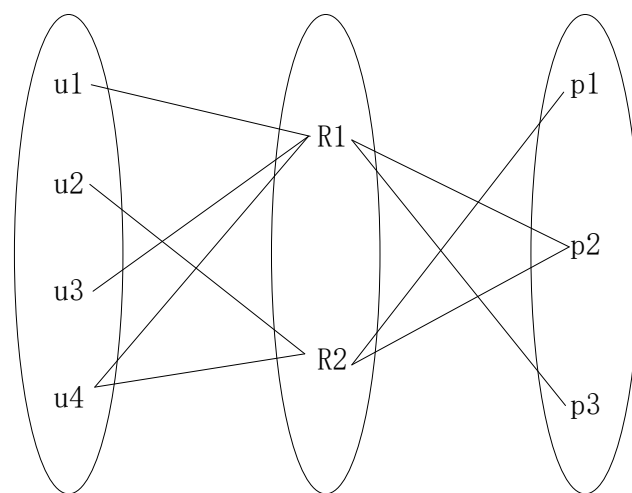


Figure 2. Illustration of Edge RMP processing results.

Role mining plays a crucial role in optimizing databases by aiming to obtain the minimal role set and the minimal set of edges. This optimization can significantly improve database response times, especially when dealing with large datasets. Additionally, solving the Edge RMP problem offers valuable assistance to access control administrators, thereby enhancing the efficiency of permission management.

Meanwhile, as a representative of role-based access control (RBAC) systems, a key unresolved challenge arises during the transition from non-RBAC systems to RBAC systems: how to mine roles based on constraints. In this regard, most existing role engineering methods only partially integrate organizational business rules into access control considerations. They aim to establish effective sets of roles by reflecting these rules in the process of role definition, naming, structuring, and construction. However, despite the pivotal role constraints

play in RBAC, techniques for role mining based on constraints are not widely available. Cardinality constraints encompass four different types [7]: user-role cardinality constraint (UCC), permission-role cardinality constraint (PCC), role-user cardinality constraint (RUC), and role-permission cardinality constraint (RPC). This paper discusses RUC and RPC algorithms. Many existing role optimization methods, which rely on cardinality constraints, frequently overlook the validation of other security constraints within the constructed RBAC system. Aside from cardinality constraints, exclusion constraints and user capability constraints are equally pivotal in enforcing security policies, particularly throughout the role assignment procedure. Static mutual exclusion role (SMER) [8] constraints are one of the most common types of constraints, aiming to limit the role memberships a single user can possess, implying that a user can obtain a restricted number of roles during assignment. Another type of constraint is user capability constraints, which dictate the roles a user can possess based on their abilities or qualifications. This is frequently encountered in real-world scenarios, where users with specific expertise or qualifications may be eligible for certain roles but not others. For instance, users with a computer science degree may be suited for roles in software development or testing but not in procurement or finance. User capability constraints account for these differences in user abilities and qualifications. While these constraints must be taken into consideration during role assignment, it is common for precise information about the system and constraints to be unavailable, necessitating a bottom-up approach for derivation.

To address the aforementioned challenges, we propose a post-processing constraint-based role mining algorithm called MFC-RMA. This algorithm comprises several components. Firstly, the raw data undergo preprocessing through partitioning of the data matrix using the k-means clustering algorithm. This step involves identifying the centroids of each matrix and compressing the matrices based on these centroids. Subsequently, the core algorithm for solving the Edge RMP is introduced, which aims to minimize the associations between user roles and role permissions. The fewer associations between permissions and users, the easier it becomes for the system to control and monitor access to critical resources. Finally, the results generated by the previous algorithm are further refined to meet the administrator's requirements for role accuracy within the system. Given the scarcity of constraint algorithms in this area, we propose role-user cardinality constraints, role-permission cardinality constraints, and multiple constraint algorithms.

The primary contribution of this paper lies in the introduction of a novel MFC-RMA algorithm, which leverages Boolean matrix factorization to enhance the efficiency of Edge RMP through the transformation of relationships between various user roles and role permissions. By reducing the interdependencies between user roles and role permissions, the MFC-RMA algorithm diminishes the complexity of the system. This simplification facilitates the ease of system administration and maintenance, mitigates potential errors and oversights, and consequently enhances system security. The reduction in interdependencies also leads to a more centralized and efficient approach to permission management, providing significant support for RBAC systems. Subsequently, utilizing partitioning and compression techniques for data preprocessing based on similarities, we designed a post-processing cardinality-constrained algorithm, which achieves higher comprehensive performance compared to other algorithms by integrating Boolean matrix representations, role mining for user roles, and permission attributes.

Our work's contributions can be delineated into four facets as follows.

1. In the context of Edge RMP, an optimization algorithm is proposed to address the hierarchy and redundancy among roles, thereby enhancing the efficiency of the final results. This optimization aims to minimize the relationships between users and roles, as well as between roles and permissions, leading to optimized system manageability and security. Experimental results demonstrate the superiority of the proposed algorithm across various parameters.
2. By clustering the dataset based on user similarity and compressing the data according to permission support, role mining can be performed efficiently. This approach

- reduces the computational complexity of role mining and ensures that the generated roles remain consistent with the attributes of the original dataset within an acceptable margin of error, thereby enhancing the accuracy and reliability of role mining.
3. Optimization algorithms for role engineering are proposed, considering constraints on the cardinality of role-user assignments and role-permission assignments. These algorithms ensure that the number of users assigned to each role and the number of permissions assigned to each role fall within specified ranges, preventing roles from being overly concentrated or dispersed. Experimental comparisons demonstrate the advantages of the proposed algorithms over other methods.
 4. A role engineering approach under multiple constraints is introduced, ensuring that role-user assignments comply with constraints on cardinality, user capabilities, and user exclusion. This prevents conflicts among users assigned to different roles, mitigating the risk of permission confusion or misuse. The accuracy of the generated roles is guaranteed, and the approach is practically meaningful. Experimental results illustrate the variation in the preservation rate of role assignments under different parameter settings for the proposed algorithm.

The remainder of this paper is organized as follows: Section 2 presents the related work that serves as the theoretical foundation for the research content of this paper. Section 3 delves into the theoretical underpinnings of the algorithms involved in our research. Section 4 introduces the MFC-RMA algorithm's optimization of Edge-RMP, along with the experimental setup and data preprocessing. It also discusses the post-processing cardinality-constrained algorithms RUC and RPC, as well as the hybrid constraint algorithm within the MFC algorithm. This section concludes with the utilization of real-world datasets and experimental comparisons to validate the feasibility of the proposed enhancements. Section 5 discusses the impact, performance, and necessity of the algorithms presented in this paper. Finally, Section 6 summarizes the paper, addresses its limitations, and outlines directions for future work.

2. Related Work

2.1. Role Engineering

With the introduction of RBAC, the most critical challenge in building an RBAC model is the identification of a suitable set of roles, which is known as role engineering. The concept of role engineering was initially proposed by E.J. Coyne [4]. The goal of role engineering, as explicitly stated in the article, is to establish a comprehensive, accurate, and efficient set of roles, which essentially involves studying the relationship between roles and permissions. There are primarily three methods for achieving role engineering: top-down, bottom-up, and hybrid approaches. The top-down approach emphasizes role construction through the analysis of business processes, while the bottom-up approach typically focuses on consolidating existing permissions into roles. These two methods are considered the fundamental approaches to role engineering, and the hybrid approach combines them for practical usage.

1. Top-down Approach

The top-down approach relies on a comprehensive understanding and analysis of the internal business processes of the system to meet various business requirements. It involves analyzing the permissions required for each functionality, extraction of relevant business information, and the gradual assignment of appropriate roles. Because this approach considers business logic and system functional requirements in role generation, its results are more practical and easier for security administrators to comprehend.

However, the top-down approach faces a significant challenge: scalability issues arise as the volume of business processes, users, and permissions grows substantially. This method is time-consuming and often requires significant manual intervention. Additionally, initiating the analysis from internal business processes may overlook existing user-

permission relationships in the system, potentially resulting in inconsistencies between the resulting roles and corresponding permissions and the original access control data.

To address these challenges, Narouei and Takabi [9] introduced an automated top-down role engineering methodology leveraging natural language processing techniques. Additionally, contemporary access management tools offer workflow and validation functionalities, thereby simplifying the intricacies associated with top-down role modeling.

2. Bottom-up Approach

Thomsen et al. [10] proposed a bottom-up approach, which generates roles from permissions derived from some objects and their corresponding methods. The lifecycle comprises four stages, all within the framework of the role lifecycle: role analysis, role design, role management, and role maintenance. Kuhlmann et al. [11] introduced another bottom-up approach utilizing clustering techniques similar to k-means clustering, requiring a predefined number of clusters. In [12], Schlegelmilch and Steffens proposed an aggregation-based clustering method for role mining (referred to as ORCA), discovering roles by appropriately merging permissions. However, the order of permission merging in ORCA determines the resulting roles, and it does not allow for overlapping roles (i.e., users cannot assume multiple roles), which is a significant limitation. Recently, Vaidya et al. [13] introduced a subset enumeration-based method called Role Miner, which eliminates the aforementioned restrictions. It encompasses an automated process that considers current user-permission assignments to define roles. In [14], a novel role engineering method called REO_CCUMEC was introduced, which transforms the role mining problem into a clustering problem. It utilizes partitioning and compression techniques to eradicate redundancy.

3. Hybrid Approach

Hybrid role engineering techniques have been proposed to amalgamate the benefits of both top-down and bottom-up approaches. Fuchs and Pernul [15] addressed this gap by analyzing existing methods and introducing the Hydro tool-supported approach. This approach facilitates the integration of existing identity information and access permissions tools, while also emphasizing the significance of managerial knowledge regarding their employees. Frank [16] introduced statistical metrics for analyzing the significance of various types of business information in defining roles. He proposed a method that incorporates relevant business information into a probabilistic model, along with an algorithm for hybrid role mining. Molloy [17] investigated role mining problems in scenarios with varying levels of information availability.

2.2. Role Mining

In building role sets that meet the demands of practical scenarios, including a hierarchical structure for role relationships is crucial. This helps to enhance the understanding of roles and alleviates the burden of permission management. Numerous algorithms have emerged to tackle the role mining problem (RMP), employing heuristic techniques such as evolutionary algorithms, permission grouping, graph optimization, and various data-mining strategies. A comprehensive overview of diverse RMP solution methods is offered in [18]. Vaidya et al. [13] proposed two subset enumeration-based mining algorithms: the Complete-Miner algorithm and the Fast-Miner algorithm. Both algorithms start with sets of permissions assigned to users as initial roles and generate new roles by taking intersections of different initial roles until no new roles are generated. This approach helps to build hierarchical relationships among roles. The Complete-Miner algorithm exhaustively enumerates all possibilities of permission sets, but it may suffer from high time complexity and generate redundant roles. The Fast-Miner algorithm significantly reduces time complexity but may not produce high-quality role sets. In this paper, the Edge-RMP algorithm utilizes the Fast-Miner algorithm as a preliminary result. A study was conducted to compare and analyze several classic role mining algorithms, including the GO algorithm, ORCA algorithm, CM algorithm, HM algorithm, HPr algorithm, and HPe algorithm [19]. To address the inefficiency problem of existing role mining algorithms in re-mining when the original

permission data changes, Martin [20] proposed an incremental algorithm. This algorithm significantly improves runtime efficiency and reduces the number of roles while maintaining appropriate policy complexity. It provides an effective solution to the problem caused by changes in permission data. These algorithms offer different approaches to building role sets and managing role hierarchical relationships to meet the needs of different scenarios. Anderer et al. [21] introduced a comprehensive event handling method for hierarchical structural changes prompted by dynamic events in role mining. They seamlessly integrated this method into the framework of evolutionary role mining algorithms.

Vaidya et al. [22] proposed a comprehensive role optimization framework and provided a series of greedy algorithms to address various problems such as Basic RMP, δ -approximate RMP, minimum-noise RMP, and Edge RMP. In their study, a series of greedy algorithms were introduced to solve Basic-RMP and Edge RMP problems, all based on Boolean matrix factorization. The Basic-RMP algorithm generates a set of candidate roles using Fast Miner, creates an initial PA, and then selects roles from the candidate role set using a greedy strategy to generate UA. In each iteration, roles satisfying the maximum number of constraints (basic keys) are selected. The Edge-RMP algorithm uses edge keys instead of basic keys, where an edge key of a role is associated with the total number of users associated with a non-zero element in UPA. However, the results obtained by their proposed algorithms exhibit redundancy in the number of edges and roles. This paper improves the Edge-RMP algorithm by analyzing the logical relationships existing in each row of PA, enabling one row to be represented by other rows. Redundant roles are further eliminated based on rules for redundant role identification to simplify PA, and any rows with common permissions are processed accordingly. Experimental results demonstrate a significant improvement in the efficiency of the optimized Edge-RMP algorithm. Although the HPe algorithm proposed in [23] addresses Edge RMP on graphs, there is still room for improvement in terms of the values assigned to the edges and the number of roles generated. Huang's GA_{edge} algorithm [24], which is based on set covering, represent a significant improvement upon the HPe algorithm in terms of both the number of edges generated and the number of roles produced. Our proposed algorithm builds upon the foundation of [22] and, as evidenced by our experimental results, demonstrates advantages over both of these algorithms.

Although role optimization involves large-scale operations, Colantonio et al. [25] reduced the complexity of problem-solving by partitioning the user-permission assignment dataset into multiple subsets. On the other hand, Verde et al. [26] transformed role mining into a clustering problem to compress the mining scale and extract multiple partitions' similar features to ensure the integrity of mining results. Although these methods did not consider constraint conditions, constraints play a crucial role in the RBAC model.

2.3. Boolean Matrix Factorization

Matrix factorization techniques are widely applied in the current data mining domain [27], including but not limited to recommendation systems, biological data analysis, data dimensionality reduction, noise removal, and community detection, among others [28]. The fundamental concept revolves around decomposing a matrix into two or more submatrices, which is closely associated with the role mining problem discussed in this paper. Specifically, it involves decomposing the user-permission relationship into user-role assignment and role-permission assignment relationships.

The Boolean matrix factorization problem and the role mining problem are formally similar because they both have well-defined objective functions and constraints. The rows and columns of the matrix can correspond to attributes in the context of role mining. Due to this similarity, the Boolean matrix factorization problem can serve as an effective role mining modeling tool, leveraging existing Boolean matrix algorithms to optimize solutions and extract appropriate roles.

2.4. Constraints

Several approaches have been proposed to impose constraints in role mining. Kumar et al. [29] proposed a constrained role mining algorithm that imposes limitations on the number of permissions assigned to roles. Blundo et al. [30] introduced a heuristic method that yields a comprehensive set of roles, adhering to the same cardinality constraint. Hingankar et al. [31] proposed a bipartite graph covering approach to generate roles limiting the maximum number of users associated with roles. Ma et al. [7] suggested a role mining algorithm grounded on a permission cardinality constraint and user cardinality constraint to cap the maximum number of users or permissions linked with roles. Harika and colleagues [32] introduced two role optimization strategies, namely post-processing and parallel processing, to constrain the maximum number of roles allocated to users and associated permissions simultaneously. The post-processing method involves initially mining roles without constraint consideration. Subsequently, during optimization, user-role and role-permission assignments are scrutinized to ensure adherence to constraints. If necessary, appropriate reallocation is performed to maintain compliance. The parallel processing method achieves dual-constraint optimization during the role mining process. The cardinality constraints in MFC-RMA, which we propose, follow a post-processing approach. Additionally, Sarana et al. [33] proposed three role optimization methods, including applying separation of duty constraints during, after, or between the mining processes. To satisfy separation of duty constraints and ensure authorization security, Sun et al. [34] proposed a method called Role Mining Optimization, which combines separation of duty constraints with authorization security checks.

Regarding constraints, Blundo and Cimato [30] formally defined a constrained version of the role mining problem that considers cardinality constraints, limiting the maximum number of permissions that can exist within a role. They referred to this problem as the *t*-constrained role mining problem. By restricting the number of permissions that can be present in a role, this constraint prevents any user from being burdened with an excessive number of operations due to the assignment of a specific role. In order to satisfy the *t*-*t* SMER constraints in RBAC and perform role assignments, Roy et al. [35] presented an approach aiming to determine the minimum number of users that fulfill multiple *t*-*t* SMER constraints. Subsequently, they defined the minimum users problem for cardinality constraint mutual exclusion tasks, which aims to identify the minimum number of users capable of executing tasks under specified security constraints [36]. Additionally, Roy et al. [37] proposed a method incorporating multiple constraints, including cardinality constraints and user capability constraints, based on relevant business instances. The work in [38] effectively captures organizational states in analysis by transforming roles automatically outputted by the mining process, generating a role set suitable for real-world application based on constraints involving the number of roles per user and the number of permissions per role. Blundo et al. [39] proposed a method within a novel model to prevent the generation of roles that share the same set of permissions, taking into account constraints on the cardinality of role–user relationships. However, there is still room for performance improvement in their algorithm. In this paper, we compare and build upon their algorithm in the section focusing on cardinality constraints, introducing enhancements and modifications to further optimize its performance.

Through the analysis above, we have identified three main limitations in existing research. Firstly, the scale of role mining is extensive, often resulting in redundant outcomes. Secondly, as the number of associations between users and permissions increases, controlling the system becomes more challenging. Additionally, most role optimization methods only address a single cardinality constraint and overlook the evaluation of authorization security within the constructed RBAC system. Consequently, role assignments may fail to simultaneously satisfy user capability constraints and mutual exclusion constraints. Lastly, existing role allocation methods assume the existence of an RBAC system, whereas in reality, the system may be unknown, and constraint conditions may be uncertain.

Currently, both the Edge RMP variants proposed in [23,24] and the constraint handling algorithm in [39] exhibit certain limitations. The former still produces redundant results, leaving room for optimization, while the latter, despite making progress in reducing the number of roles generated, still requires improvement in overall performance. Our work addresses these issues, delivering superior outcomes that enhance the efficiency and accuracy of real-world RBAC management.

Therefore, this paper proposes a role engineering method (MFC-RMA), primarily focused on the following four aspects:

1. Employing partitioning and compression techniques for data preprocessing based on similarity.
2. Enhancing the Edge-RMP algorithm to eliminate redundancies and improve results.
3. Introducing a post-processing approach to tackle cardinality constraint-based role optimization problems and corresponding algorithms.
4. Achieving maximum role allocation that satisfies multiple security constraint conditions in the constructed RBAC system.

By conducting experiments with real datasets, we evaluated the performance of this method through experimental results, and its advantages and limitations are discussed.

3. Theoretical Foundation

Through in-depth research into the RBAC model and considering the evolving needs of businesses, four major models have emerged, each uniformly named by the National Institute of Standards and Technology (NIST): the basic model RBAC0, the hierarchical model RBAC1, the constrained model RBAC2, and the consolidated model RBAC3.

In the context of the RBAC model, for the sake of simplicity, this paper does not consider sessions, role hierarchies, or constraints and adopts RBAC0, which provides the minimum security functionality required by RBAC. RBAC0 consists of users, roles, and permissions: users are independent entities entitled to autonomous access to object resources; roles represent various job responsibilities within an organization and are collections of permissions; and permissions, on the other hand, are abstract concepts indicating the right to perform specific actions.

In RBAC, roles play a central role in access control, with users obtaining necessary permissions by activating roles assigned to them. A role may encompass multiple authorized permissions, and each permission can be part of several roles. Similarly, each user may have multiple authorized roles, and conversely, each role can be assigned to multiple users.

3.1. Role-Based Access Control Model

In RBAC, access permissions are managed and controlled based on users' roles. Each user is assigned one or more roles, and roles have specific permissions and access rights. The fundamental idea of the RBAC model is to associate users' access permissions with roles rather than directly granting permissions to users. This approach simplifies permission management, enhances system security, and reduces administrative costs. Through the concept of roles, the RBAC model enables administrators to more easily control users' permissions and allows for easy adjustment and management of access control policies according to organizational needs.

The basic elements contained in an RBAC system can be formalized as follows:

U , a set of users, with a total of m users.

P , a set of permissions, with a total of n permissions.

R , a set of roles, with a total of k roles.

$UA \subseteq U \times R$, representing the user-role assignment relationship, with dimensions $m \times k$.

$PA \subseteq R \times P$, representing the role-permission assignment relationship, with dimensions $k \times n$.

$UPA \subseteq U \times P$, representing the user-permission assignment relationship, with dimensions $m \times n$.

$user_roles(u) = \{r | \exists r \in R : (u, r) \in UA\}$, the mapping of user u onto a set of roles.

$role_users(r) = \{u | \exists u \in U : (u, r) \in UA\}$, the mapping of role r onto a set of users.

$role_permissions(r) = \{p | \exists p \in P : (r, p) \in PA\}$, the assignment of role r to a group of permissions;

$permission_roles(p) = \{r | \exists r \in R : (r, p) \in PA\}$, the assignment of permission p to a group of roles.

$user_permissions(u) = \{p | \exists p \in P, \exists r \in R : ((u, r) \in UA) \wedge ((r, p) \in PA)\}$, the assignment of user u to a group of permissions.

3.2. Constraints

This paper considers different types of constraints in RBAC: role-user cardinality constraint (RUC), role-permission cardinality constraint (RPC), mutual exclusion constraint, and user capability constraint.

(1) RUC and RPC

RUC [7] specifies that for a given user set U , role set R and threshold MUC_{role} , the number of users assigned to any role should not exceed MUC_{role} , as shown in Equation (1):

$$\forall r \in R : |role_users(r)| \leq MUC_{role} \leq |U| \quad (1)$$

RPC [7] stipulates that for a given permission set P , role set R , and threshold MPC_{role} , the number of roles assigned to any permission should not exceed MPC_{role} , as shown in Equation (2):

$$\forall r \in R : |role_permissions(r)| \leq MPC_{role} \leq |P| \quad (2)$$

(2) SMER

For a given set of m roles $r_1, r_2, \dots, r_m$, no user is allowed to possess t or more of these m roles. This constraint is represented as $SMER < \{r_1, r_2, \dots, r_m\}, t >$, where m and t are integers satisfying $2 \leq t \leq m$ [8], as shown in Equation (3).

$$\forall u \in U : |\{r_1, r_2, \dots, r_m\} \cap user_roles(u)| < t \quad (3)$$

In this study, we confine SMER constraint to encompass only the canonical form, specifically t - t SMER constraint. The canonical SMER constraint pertains to the scenario when t equals m , commonly known as t - t SMER constraint. Furthermore, prior studies have demonstrated that any t - m SMER constraint can be accurately depicted as an aggregation of t - t SMER constraints. Consequently, our investigation in this study solely concentrates on the canonical form of SMER constraint.

(3) User capability constraint [37]

The user capability (UC) constraint is represented by a matrix UC of size $|U| \times |R|$, where each element $UC[i, j]$ in the matrix indicates whether user u_i can handle role r_j . If $UC[i, j]$ is 1, it means that user u_i can handle role r_j ; if $UC[i, j]$ is 0, it means that user u_i cannot handle role r_j . The satisfaction condition of the UC constraint is that for each element in the matrix, only when $UC[i, j]$ is 0, the corresponding element $UA[i, j]$ in the UA matrix can be 0.

In this paper, the knowledge definitions applied to access control system constraints include the following:

1. The degree of constraint on a role.

With t - t SMER constraint set $C = \{c_1, c_2, \dots, c_i, \dots\}$, where $c_i = SMER < \{r_1, r_2, \dots, r_{ti}\}, ti >$, the degree of constraint on role r in C refers to the percentage of constraints in C that include r .

We use the degree to represent the number of occurrences of a role in the SMER constraints. The higher its value, the lower the priority of the role.

$$smer_c(r) = \frac{|\{c_k | \exists c_k \in C : r \text{ is included in } c_k\}|}{|C|} \quad (4)$$

2. User retention ratio

The user retention ratio (URR) denotes the ratio between the user-role assignment relationships generated by the algorithm under multiple constraints and the initial user-role assignment relationships. A higher value indicates better role performance generated under constraints.

$$URR = \frac{|UA'|}{|UA|} \times 100\% \quad (5)$$

3.3. Matrix Decomposition

Boolean Matrix Factorization

The multiplication of a $m \times k$ order Boolean matrix UA and a $k \times n$ order Boolean matrix PA can be represented as $UA \otimes PA = UPA$, where UPA is a $m \times n$ order Boolean matrix that satisfies Equation (6). $UA \otimes PA$ is referred to as the factorization of UPA .

$$x_{ij} = \bigvee_{l=1}^k (c_{il} \wedge r_{lj}) \quad (6)$$

In this context, “ x ” represents a unit within UPA , “ c ” denotes a unit in UA , and “ r ” signifies a unit within PA .

3.4. Role Mining Problem

The role mining problem refers to the process of identifying and discovering roles from user-permission assignment data. In the domain of access control, roles represent sets of permissions that can be conveniently assigned to users. The objective of role mining is to identify groups of users with similar access patterns and permission requirements by analyzing their access behaviors. These groups are then categorized into roles, simplifying access control management and enhancing system security and efficiency.

The definition of the concept related to the role mining problem is as follows.

1. Basic RMP

Given a user-permission assignment (UPA), find the smallest set of roles that satisfies $UA \otimes PA = UPA$.

2. Edge RMP

The Basic RMP aims to find the smallest set of roles, while the Edge RMP seeks to minimize the sum of the sizes of the user-role and role-permission assignment relationships, i.e., $\text{Min}|UA| + |PA|$.

3. Similarity

In statistics, the Jaccard coefficient [40] is commonly employed to assess the similarity or dissimilarity between distinct sets of samples, with the objective of discerning sample clusters. Given a set $U = \{U_1, U_2, \dots, U_i, \dots, U_j, \dots\}$, the formula for calculating the similarity between U_i and U_j is as follows:

$$\text{Sim}(U_i, U_j) = \frac{|U_i \cap U_j|}{|U_i \cup U_j|} \quad (7)$$

$$\text{Dissim}(U_i, U_j) = 1 - \frac{|U_i \cap U_j|}{|U_i \cup U_j|} \quad (8)$$

In this paper, the calculation of similarity and dissimilarity between user u_i and user u_j is as follows:

$$\text{Sim}(u_i, u_j) = \frac{|\text{user_permission}(u_i) \cap \text{user_permission}(u_j)|}{|\text{user_permission}(u_i) \cup \text{user_permission}(u_j)|} \quad (9)$$

$$Dissim(u_i, u_j) = 1 - sim(ui, uj) \quad (10)$$

$|user_permission(u_i) \cap user_permission(u_j)|$ represents the intersection of permissions between user i and user j in the Boolean matrix, while $|user_permission(u_i) \cup user_permission(u_j)|$ represents the union of permissions between user i and user j in the Boolean matrix.

4. Support degree of a permission

Given the $CU = \{U_1, U_2, \dots, U_i, \dots\}$, the percentage of different users in CU who possess permission p is referred to as the support of permission p relative to CU . This percentage is denoted as follows:

$$support_{CU(p)} = \frac{|\{u_k | \exists u_k \in CU : p \in user_permissions(u_k)\}|}{|CU|} \quad (11)$$

5. Compression point

The term “compression point” refers to a user u_i in the given user cluster $CU = \{U_1, U_2, \dots, U_i, \dots\}$ and a threshold t that satisfies the following condition:

$$\exists u_i \in CU, \forall p_{ij} \in u_i : support_{cu(p_{ij})} \geq t \quad (12)$$

6. Weighted structural complexity

For $w_r, w_u, w_p, w_d \in Q^+ \cup \{\infty\}$, given the weight vector $W = \langle w_r, w_u, w_p, w_d \rangle$, the weighted structural complexity (WSC) of a state $\gamma = \langle R, UA, PA, DUPA \rangle$, denoted by $wsc(\gamma, W)$, is computed as follows:

$$wsc(\gamma, W) = w_r \cdot |R| + w_u \cdot |UA| + w_p \cdot |PA| + w_d \cdot |DUPA| \quad (13)$$

To limit the RBAC states to be considered, different values can be set for the weights w_r, w_u, w_p , and w_d . As a result, different weight vectors encode different mining objectives and minimization goals. In this work, we are interested in comparing heuristic algorithms based on their performance in terms of overall state complexity. Therefore, we follow the common practice in the literature and set $w_r = w_u = w_p = w_d = 1$.

4. Role Mining Algorithm

This section introduces the MFC-RMA method, which consists of the following components:

1. Optimization of the Edge-RMP algorithm.
2. Role optimization to satisfy cardinality constraints.
3. Role assignment that meets multiple constraints.

4.1. Edge-RMP Optimization

For role mining, a variety of algorithms serve different purposes. Edge-RMP optimization algorithm seeks to minimize the total number of user-to-role and role-to-permission assignments, thereby reducing the management workload for administrators. In this paper, we have optimized the Edge-RMP algorithm proposed by Lu et al. [22]. We propose an optimized variant of the Edge-RMP algorithm based on Boolean matrix factorization to enhance role mining. Although the Edge RMP algorithm minimizes $|UA| + |PA|$, hierarchical relationships and data redundancies among roles persist. Our algorithm further refines the results based on its model, as evidenced by experiments comparing various parameters.

In $UA \otimes PA = UPA$, if a user possesses a specific permission, at least one role with that permission must be assigned to them. Conversely, if the user lacks a specific permission, none of the roles with that permission can be assigned to them.

To perform matrix decomposition accurately, it is crucial to ensure that no user gains additional permissions or loses any. In essence, this can be achieved by establishing

constraints for each “1” or “0” present in the original *UPA*. A detailed breakdown of this follows below.

For each user, represented by index i , the user’s permission set X_i can be denoted by a binary vector $\{x_{i1}, \dots, x_{ij}, \dots, x_{in}\}$, where “ $x_{ij} = 1$ ” signifies that user i has permission j , and “0” indicates otherwise. Similarly, the permission set R_k of a candidate role k can be illustrated by $\{r_{i1}, \dots, r_{ij}, \dots, r_{in}\}$, where “ $r_{ij} = 1$ ” denotes that the role includes permission j , while “0” implies its absence.

The binary value c_{ik} , either “1” or “0”, indicates whether user i possesses role k . If “ $c_k = 1$ ”, it signifies that user i has role k ; otherwise, it denotes the absence of such a role assignment.

In order to calculate the required number of roles and identify the existence of roles, a set of new indicator variables $\{d_1, \dots, d_i, \dots, d_k\}$ has been defined, where $d_i = 0$ indicates the absence of role i , and $d_i = 1$ signifies the presence of role i . Consequently, the following constraints hold:

$$\left\{ \begin{array}{l} \sum_{j=1}^k c_{ij} r_{jt} \geq 1, \text{ if } x_{it} = 1, 1 \leq i \leq m, 1 \leq t \leq n \\ \sum_{j=1}^k c_{ij} r_{jt} = 0, \text{ if } x_{it} = 0, 1 \leq i \leq m, 1 \leq t \leq n \\ d_j \geq c_{ij}, 1 \leq i \leq m, 1 \leq j \leq k \\ d_j = 0 \text{ or } 1, 1 \leq j \leq k \\ c_{ij} = 0 \text{ or } 1, 1 \leq i \leq m, 1 \leq j \leq k \end{array} \right. \quad (14)$$

For *UPA*, there may be instances where multiple users share the same set of permissions. In such scenarios, redundant users should be removed, and the frequency of each unique user permission set should be calculated, denoted as $\{u_1, u_2, \dots, u_n\}$, where u_i represents the frequency count of the i th unique permission set. Therefore, the Edge-RMP entails identifying the following:

$$\text{Min} \sum_{i=1}^m \sum_{j=1}^k u_i c_{ij} + \sum_{i=1}^k \left(d_i \sum_{j=1}^n r_{ij} \right) \quad (15)$$

4.1.1. Algorithm for Initial Role Generation

In this work, the Fast Miner algorithm is utilized as the initial role generation algorithm. Compared to other similar algorithms such as Complete-Miner, the Fast Miner algorithm has relatively lower complexity and is suitable for handling moderately sized user-permission assignment datasets. Its core idea lies in constructing a role for each user’s permission set initially, followed by generating roles by taking intersections of permission sets between any two roles, ultimately completing the generation of initial roles.

4.1.2. Optimized Edge-RMP Algorithm

The Edge-RMP algorithm utilizes the Fast-Miner algorithm to generate initial roles. In the initial phase of role generation in the Fast-Miner algorithm based on *UPA*, before the generated roles are intersected, if there are duplicate permissions in the generated roles, the role with duplicate permissions is removed, and the count of users owning this role is incremented by 1. This forms the basis for computing the final Edge-RMP results.

The Edge-RMP algorithm aims to keep *PA* unchanged while simplifying *UA* as much as possible. Even though *PA* remains unchanged, utilizing d_i in its original form facilitates achieving the best solution for Edge-RMP. This is because, even if multiple columns in the resulting *UA* are all zeros, corresponding d_i values would be set to 0. Consequently, the entire rows in *PA* for these columns, even if not all zeros, would not contribute to the final result, leading to the optimal solution.

The Edge-RMP algorithm adeptly circumvents the issue of redundant rows in *PA*. However, *PA* itself remains in a non-minimal form. The optimized algorithm presented in this paper simplifies both *PA* and the columns in *UA* obtained through the algorithm.

This results in changes in the number of rows in PA as well as columns in UA , providing a better foundation for subsequent optimization.

For each column in UA , representing a specific role assigned to various users, if the entire column consists of zeros, it indicates that the role has not been assigned to any user. In such cases, these roles, along with their corresponding rows in PA , can be identified as redundant and removed from UA . This process leads to a minimal form of UA , and the corresponding rows in PA become meaningless and can be safely eliminated. The resulting UPA remains unchanged. This aspect of the improvement operation aligns with Edge-RMP algorithm's experimental results, with changes occurring in both PA and UA relative to their original matrices.

Regarding the PA and UA generated by Edge-RMP algorithm, there might be redundant roles within the matrices. A role is considered redundant if it does not introduce any new users or permissions within the role hierarchy. In other words, users can obtain the same permissions through other roles, or the permissions can be obtained through other roles. This paper addresses the removal of such redundant roles. This algorithm does not necessitate the construction of a role hierarchy. For each row in PA corresponding to a role, if it satisfies $Pr = \cap P_{R_i}$ and $Ur = \cup U_{R_i}$, where Pr represents the permission set of role r , and Ur represents the users who possess role r , it is deleted. This implies that the permission set Pr of role r is the intersection of permission sets of other roles (where P_{R_i} represents the permission sets of other roles), and the user set Ur is the union of these roles' user sets (where U_{R_i} represents the user sets of other roles). In this scenario, role r might be deemed redundant. Likewise, when aiming to minimize the number of edges, a role containing permissions forming a subset S of all permission sets is included in the solution only if a user is authorized for a superset of S . Additionally, unless S precisely matches a user's permission set, S must be a subset of permissions held by at least two users. Otherwise, it is feasible to merge multiple roles into one, substantially reducing the number of edges [41].

According to the literature [23], for a specific row in PA , if it encompasses all permissions of another row, it can be represented by adding a vector row to it, as depicted in Figure 3. Consequently, users assigned this role can also be assigned its subset roles. It is worth noting that these subset roles may contain further subsets, leading to modifications in multiple cells of a UA row. Additionally, it is essential to ensure that the reduction in "1" cells in PA exceeds or equals the increase in "1" cells in UA , particularly for larger datasets.

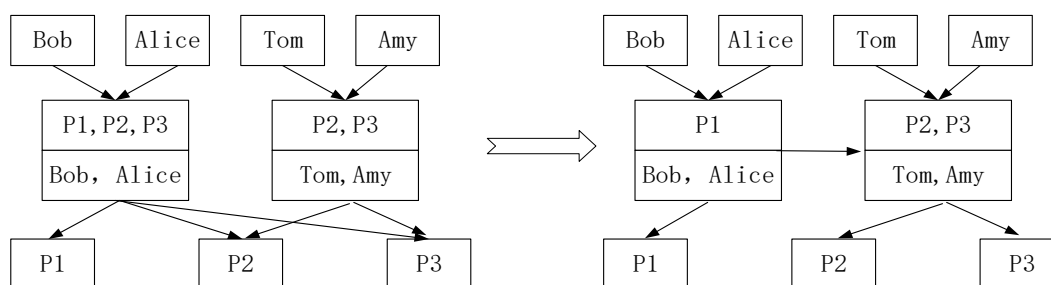


Figure 3. A role permission set is a superset of another role permission set.

In UA , columns consisting solely of "0" are removed, denoted by $\{c1, c2, \dots, ck\}$, along with their corresponding rows in PA , denoted as $\{c1, c2, \dots, ck\}$. In the newly generated PA , if a row "a" is a superset of another row "k", it can be represented as the subtraction of its specific subset vector plus the subset vector itself. Accordingly, for each UA row containing role "a", cells in columns corresponding to "k" are updated to "1". If subset roles "k" further contain other subsets "l", cells corresponding to "k" and "l" in this row are also updated to "1".

Similarly, if any two rows in PA share common permissions, these rows are simultaneously removed from PA . Subsequently, a new row is added to PA to represent a single role that encompasses the common permissions. In UA , a new column is added with the corresponding values. However, it is crucial to note that these row and column addition

operations will increase the number of generated roles. Therefore, it is advisable to make these decisions based on computational metrics. The specific code implementation is not included here, as depicted in Figure 4.

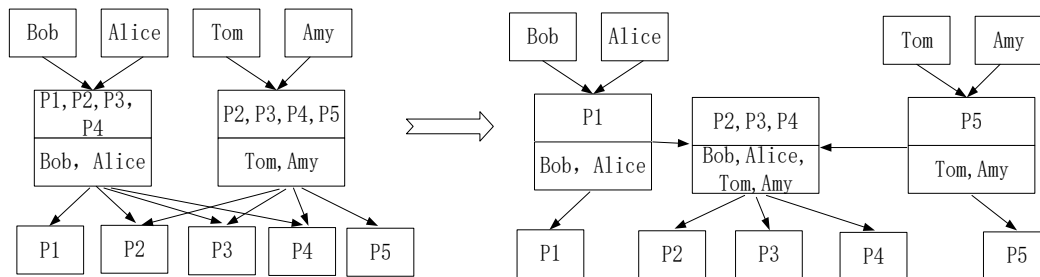


Figure 4. Two roles have a common permission set.

In Algorithm 1, Steps 1 and 2 aim to remove obvious redundant roles from UA and PA . In Step 3, if a role in the role hierarchy introduces neither new users nor new permissions, it is deleted. Step 4 handles the case where a role set in PA is a superset of several others. The pseudocode does not elaborate on the situation where two roles share a common permission set.

The time complexity of the proposed algorithm is $O(r \times m/2)$, which iterates over each role in PA , excluding subsets that have been identified previously, and finally traverses the users to update their states. Clearly, the system's performance is directly influenced by the number of users and roles in the system. To enhance the scalability of the RBAC system, the following strategies can be considered:

Optimization of Data Structures: Employing more efficient data structures to store and query user–role relationships can significantly accelerate permission verification and role assignment. Hash tables, index trees, and similar data structures offer viable options to boost these operations.

Caching Strategies: For frequently accessed data, such as user–role mappings and permission lists, caching mechanisms can significantly reduce the number of database accesses and improve overall system performance.

Distributed System Architecture: In large-scale systems, adopting a distributed architecture can distribute the user and role data across multiple nodes for processing. This approach enhances the system's parallel processing capabilities and fault tolerance.

Limitation of User and Role Numbers: In certain scenarios, limiting the number of users and roles can effectively enhance system scalability. Merging similar roles or eliminating unnecessary ones are practical methods for reducing the role count.

These measures can be implemented to facilitate the application of the algorithm in large-scale databases, ensuring its scalability and performance in real-world scenarios.

For illustration, consider the example shown in Figure 5a.

Where UA and PA are derived from the Edge-RMP, and according to Equation (15), their value is 19.

$$\begin{pmatrix} 0011001 \\ 0001000 \\ 1010000 \\ 0110000 \end{pmatrix} \begin{pmatrix} 00010 \\ 10000 \\ 00100 \\ 11010 \\ 00110 \\ 10100 \\ 11111 \end{pmatrix} = \begin{pmatrix} 11111 \\ 11010 \\ 00110 \\ 10100 \end{pmatrix} \quad (a)$$

$$\begin{pmatrix} 00111 \\ 00010 \\ 10100 \\ 01100 \end{pmatrix} \begin{pmatrix} 00010 \\ 10000 \\ 00100 \\ 11010 \\ 11111 \end{pmatrix} = \begin{pmatrix} 11111 \\ 11010 \\ 00110 \\ 10100 \end{pmatrix} \quad (b)$$

Figure 5. Illustration of preliminary experimental results: (a) Edge-RMP results and (b) the results of removing redundant rows and columns.

Algorithm 1. Edge-RMP Optimized Algorithm

Input: User–role relationships UA , role–permission relationships PA , and user–permission relationships UPA generated by the Edge-RMP algorithm [22].

Output: New UA and PA generated by the optimized algorithm.

Step 1: Remove columns in UA that are all zeros.
All listed as $\{c1, c2, \dots, ck\}$.

Step 2: Remove corresponding rows in PA for row in range $\{c1, c2, \dots, ck\}$

Step 3: For each row $r \in PA$ do
 If $\exists R' \subseteq PA, Pr = \cap P_{R'} \& \& Ur = \cup U_{R'}$, then
 $PA = PA \setminus \{r\}$
 End if
 End for

Step 4: For each row a in PA do
 If a is a superset of other rows in PA with the maximum number of permissions, then
 Let k be the subset corresponding to row a
 If k has other subsets $\{l\}$, then
 Let the subsets be $\{k, l\}$
 End if
 For each row in UA do
 If the value for a in row is 1,
 //To mitigate the potential increase in the combined value of $|UA|$ and $|PA|$ after modifications to PA , we implement a counting mechanism to ensure that such changes do not exceed the original count.
 Then count the number of zeros in columns $\{k, l\}$ of row a in PA and record it.
 $Numi+ = \text{the numbers}$
 End if
 End for
 //The product of the repetition count of unique permission sets for users and the number of changes in UA is less than the number of units replaced in PA represented by subsets.
 If $\sum_{i=1}^m u_i * num_i \leq \text{the numbers of 1 in subset } k$ then
 //Make changes to role a in PA
 Row $a = a - k$
 //Make changes to users in UA who have role a
 Set the recorded cell to 1.
 End if
 End if
 End for

 //When two roles in PA share a common permission, r_c , the respective permission sets of these two roles are modified by subtracting r_c from their respective vectors. Concurrently, r_c is established as a new role within PA . In UA , a new column is appended with corresponding values to reflect this change. Additionally, it is ensured that the product of the repetition count of unique permission sets for users and the number of changes in UA remains less than the number of units replaced in PA , represented by subsets. For the calculation of commonalities in this scenario, all rows in PA are sorted in descending order based on the number of permissions in each row.

Following Step 1, the 5th and 6th columns are removed from UA .

In Step 2, the 5th and 6th rows are removed from PA . This point in the procedure is shown in Figure 5b.

Following Step 3, the repetition counts of unique permission sets possessed by users in UPA are $\{1, 1, 1, 1\}$. In PA , the fourth row $r4$ is a superset of the first row $r1$, and $r1$ is not a superset of any other rows. Therefore, in PA , $r4$ can be represented as $\{11000\} + r1$. In UA , there are 2 cells that contain $r4$ and not $r1$, which correspond to $u1 * 1 + u2 * 1 = 2$. However, since 2 is not less than or equal to 1, the change in PA is canceled. Similarly, it does not hold for $r4$ with $r2$.

The fifth row $r5$ in PA , which is a superset of $r4$ and where $r4$ has the maximum number of permissions among all rows in PA except $r5$, can be represented as $\{00101\} + r4$. In UA , there are 0 cells containing $r5$ and not containing $r4$, corresponding to $u1 * 0 = 0$. In this

case, 0 is less than or equal to 2, so the change is retained. Since the first row in UA already contains $r4$ by itself, $\{00111\}$ in UA remains unchanged. In summary, the result is shown in Figure 6.

$$\begin{pmatrix} 00111 \\ 00010 \\ 10100 \\ 01100 \end{pmatrix} \begin{pmatrix} 00010 \\ 10000 \\ 00100 \\ 11010 \\ 00101 \end{pmatrix} = \begin{pmatrix} 11111 \\ 11010 \\ 00110 \\ 10100 \end{pmatrix}$$

Figure 6. Optimized Edge-RMP algorithm results.

According to Equation (15), the value is determined to be 17. This represents an improvement compared to the original result.

4.2. Preprocessing of UPA

This paper clusters data based on the similarity between users as determined by the permissions they possess. Users with similar similarity are grouped into one partition, and their centroids are identified. Finally, compression is applied to ensure that the roles generated from the compressed matrix have an error within an acceptable range compared to those generated before compression. This ensures data quality and consistency while simplifying the dataset. The partitioning is based on the similarity of users to permissions, as demonstrated by experiments [14], proving its feasibility. However, due to differences in the constraint algorithm and the data involved in the optimized Edge-RMP algorithm presented in the previous section, the algorithmic operations also vary. Consequently, this approach is not applicable to the optimized Edge-RMP algorithm because preprocessing UPA does not ensure that the roles generated by the optimized Edge-RMP algorithm fall within the acceptable error range compared to those generated without preprocessing.

4.2.1. Clustering

First, each user in UPA is clustered based on the similarity between users using the k-means clustering algorithm, dividing them into clusters $\{CU1, CU2, \dots\}$. Then, partitioning and compression techniques are applied independently to each cluster. Here, the partitioning around medoids (PAM) algorithm, similar to the k-means clustering algorithm but using dissimilarity instead of distance, is employed [42].

In Algorithm, k initial center points are randomly selected, and the users are partitioned accordingly. During the algorithm's execution, if there exists a non-center point closer to another point (here, in terms of similarity) than its current center, the two points are swapped, and the partitioning is recalculated. The rest of the approach follows a methodology similar to the k-means clustering algorithm.

4.2.2. Compression

After partitioning the user clusters based on user similarity, we further simplify UPA using support and compression points. The compression process is outlined in Algorithm 2.

Algorithm 2. Compression of partitions

Input: role–permission relationships PA , the partition with point u_i and threshold t

Output: compressed matrix $UPA_{compressed}$

Initialize $UPA_{compressed}$;

For each p in PA **do**

$$support_{associate(u_i) \cup \{u_i\}}(p) = \frac{|\{u \mid \exists u \in associate(u_i) \cup \{u_i\}; p \in user_permissions(u)\}|}{|associate(u_i) \cup \{u_i\}|};$$

If $support_{associate(u_i) \cup \{u_i\}}(p) \geq t$, **then**

$UPA_{compress} = UPA_{compress} \cup \{(u_i, p)\}$;

End if

End for

If the support of a permission in the user set is greater than or equal to the threshold, then this permission and its corresponding users can be regarded as compression points.

When addressing variations or constraints in data quality, particularly during the initial clustering and data preprocessing stages, the existence of missing values in datasets can potentially undermine the accuracy of clustering and data analysis. Moreover, noise data poses another challenge, as it may introduce patterns that are unrelated to the genuine data distribution, thereby compromising clustering accuracy.

To address these concerns, various techniques can be applied. For instance, statistical values such as mean, median, or mode can be utilized to fill in missing values, providing a reasonable approximation based on the distribution of available data.

Furthermore, to mitigate the impact of noise data, it is advisable to employ clustering algorithms that exhibit robustness against noise. In this context, clustering algorithms that rely on user similarity as a metric are particularly suitable. These algorithms assess the degree of similarity between users based on their attributes or behaviors and subsequently group them into clusters.

By incorporating these strategies, we can enhance the overall quality of the data and improve the precision of clustering and subsequent data analysis. Specifically, imputing missing values using statistical methods ensures data completeness, while utilizing noise-robust clustering algorithms mitigates the adverse effects of noise data. In this paper, we adopt a clustering algorithm that measures user similarity, aiming to achieve more reliable insights from our clustering and analysis efforts.

4.3. Role Optimization for Satisfying Cardinality Constraints

To optimize the results of the Optimized Edge-RMP algorithm and make them more practically meaningful, we restrict the number of users allowed per role to reduce the risks of permission misuse and leakage. The post-processing method proposed in [14] introduced algorithms for UCC and PCC, but it lacks handling methods for RUC and RPC. Additionally, according to the algorithm proposed by Ma et al. [7] for satisfying RUC and RPC, there is room for optimization in the obtained results. Thus, this paper proposes post-processing algorithms for role-user cardinality (RUC) and role-permission cardinality (RPC), demonstrating their effectiveness through experiments. Given the UA and PA data processed by the algorithm, we assess the number of users assigned to each role and the number of permissions associated with each role. If the given constraints are violated, we proceed with the following constraint optimization approach.

4.3.1. Role-User Assignment Optimization for Role-User Cardinality

Regarding RUC, it is defined as follows: Given a user-permission assignment matrix $UPA_{m \times n}$, preprocessed UA and PA matrices, and a specific threshold MUC_{role} , find a set of roles R that optimally assign user sets, keep the number of users assigned to any role less than or equal to MUC_{role} , and minimize the number of users associated with each role. The user-permission assignment derived from UA and PA matrices must align precisely with UPA matrix, such that $UA \otimes PA = UPA$. Additionally, the number of users associated with each role should be less than or equal to a predefined threshold MUC_{role} , where MUC_{role} is a maximum user count for a role and is less than the total number of users in the system. Mathematically, this condition can be expressed as $\sum_{i=1}^m UA[i][j] \leq MUC_{role} \leq |U|$ for all roles j , where $|U|$ denotes the total number of users in the system.

The constraints are as follows:

$$\begin{cases} \min |User| \\ UA \otimes PA = UPA \\ \sum_{i=1}^m UA[i][j] \leq MUC_{role} \leq |U| \end{cases} \quad (16)$$

The process of optimizing users under the constraints of RUC is illustrated in Algorithm 3.

Algorithm 3. Role-user assignment optimization for role-user cardinality (RUC)

Input: Preprocessed matrices UA and PA , initial role set CR , and threshold MUC_{role}
Output: The matrices UA and PA satisfying cardinality constraints
Define $count_role_users(r)$ as the number of users assigned to role r and compute it accordingly;
Define $count_user_roles(u)$ as the number of roles held by user u and compute it accordingly;
For each role in CR **do**
 If $count_role_users(r) > MUC_{role}$ **then**
 $k = count_role_users(r) - MUC_{role} + 1$;
 Select the top k users from the user set of $count_role_users(r)$, denoted as set S , where these users have the highest $count_user_roles(u)$ compared to others;
 Take the intersection of these k user sets, denoted as U_k , $U_k = \{U_1 \cap U_2 \dots \cap U_k\}$;
 Take the union of role permissions corresponding to the set of these intersected users, denoted as Ps ;
 Create new roles r_n ,
 let $role_permission(r_n) = Ps$;
 For each p_r in CR **do**
 If $p_r \in Ps$ **then**
 $PA[nr][r] = 1$; // If a particular permission belongs to the intersection of the permission sets of two users, it is advisable to assign that permission to a newly defined role.
 else
 $PA[nr][r] = 0$; // If a specific permission does not belong to the intersection of the permission sets of two users, it should remain unchanged.
 End if
 End for
 For each u_i in UA **do**
 For each j in U_k **do**
 // For the top k users, all the values that are set to 1 in their respective intersections should be altered to 0, and subsequently, a new role r_n should be assigned to these users.
 If $\forall u_i \in S: UA[i][k] = 1$ **then**
 $UA[i][k] = 0$;
 $UA[i][n] = 1$;
 else
 $UA[i][n] = 0$;
 End if
 End for
 End for
 Update $count_user_roles(u)$ and $count_role_users(r)$;
 End for

In Algorithm 3, we first define the number of users a certain role possesses and the number of roles a certain user possesses. We iterate over each column in UA , and if the number of users a certain role possesses exceeds the threshold MUC_{role} , we identify the user with the highest number of roles ($count_user_roles(u)$), which is equal to $count_role_users(r) - MUC_{role} + 1$. Then, we determine the intersection of these users. Subsequently, using the permission set associated with this intersection set, we introduce it as a new role into PA . The users implicated in the intersection can relinquish their previous roles and adopt the new roles, preserving their permissions. Consequently, UA and PA are adjusted accordingly. Thus, the constraints are fulfilled, and the count of users assigned to it is diminished.

In Algorithm 3, the initial phase involves defining the number of users associated with each role and the number of roles assigned to each user. Subsequently, we iterate through each column in UA matrix. If the count of users assigned to a specific role surpasses the predefined threshold MUC_{role} , our algorithm pinpoints the user(s) who possess the highest number of roles. This number is calculated by subtracting MUC_{role} from the total user count assigned to that role ($count_role_users(r) - MUC_{role} + 1$).

Next, we determine the intersection of users who meet this criterion. Leveraging the permission set associated with this intersection, we introduce a novel role into the PA

matrix. The users encompassed within this intersection can then relinquish their previous roles and adopt the newly introduced role, maintaining their original permissions.

Consequently, UA and PA matrices are updated to reflect these changes. This ensures that the constraint of limiting the number of users assigned to any role to MUC_{role} is upheld and reduces the overall count of users associated with the original role. Through this optimization process, the algorithm enhances system security and management efficiency while preserving user permissions.

To illustrate this concept, let us consider an example where the threshold is set to 3, as depicted in Figure 7.

$$\begin{pmatrix} 1000 \\ 1100 \\ 1110 \\ 1011 \end{pmatrix} \otimes \begin{pmatrix} 1001 \\ 1101 \\ 0100 \\ 1111 \end{pmatrix} = \begin{pmatrix} 1001 \\ 1101 \\ 1101 \\ 1111 \end{pmatrix}$$

Figure 7. Initial RUC data.

After application of the algorithmic steps, it is observed that role r_1 is associated with users represented by the vector $\{1, 1, 1, 1\}^T$, indicating that four users are assigned to this role. However, the number of users associated with role r_1 exceeds the threshold of 3, where $(k=2, S = \{u_3, u_4\}, U_k = \{U_3 \cap U_4\} = \{1, 0, 1, 0\}, P_s = \{r_1 \cup r_3\} = \{1, 1, 0, 1\})$.

The subsequent outcome, obtained through the prescribed steps, is illustrated in Figure 8.

$$\begin{pmatrix} 10000 \\ 11000 \\ 01001 \\ 00011 \end{pmatrix} \otimes \begin{pmatrix} 1001 \\ 1101 \\ 0100 \\ 1111 \\ 1101 \end{pmatrix} = \begin{pmatrix} 1001 \\ 1101 \\ 1101 \\ 1111 \end{pmatrix}$$

Figure 8. The refined result of the RUC algorithm.

4.3.2. Role-Permission Assignment Optimization for Role-Permission Cardinality

Similar to RUC, the optimization problem for roles containing permissions under RPC is defined as follows. Given a user-permission assignment matrix $UPA_{m \times n}$, preprocessed UA and PA matrices and a specific threshold MPC_{role} , find an optimal set of permissions P for a group of roles R , keep number of permissions assigned to any role less than or equal to MPC_{role} , and minimize the number of permissions while ensuring that $UA \otimes PA = UPA$. The number of users associated with each role should be less than or equal to MPC_{role} , and the threshold should be less than the total number of permissions; that is, the permission j of role i is $\sum_{j=1}^n PA[i][j] \leq MPC_{role} \leq |P|$.

The constraints are as follows.

$$\begin{cases} \min |Permission| \\ UA \otimes PA = UPA \\ \sum_{j=1}^n PA[i][j] \leq MPC_{role} \leq |P| \end{cases} \quad (17)$$

The optimization of role permissions under the constraints of RPC is illustrated in Algorithm 4.

Algorithm 4. Role-permission assignment optimization for role-permission cardinality (RPC)

Input: Preprocessed matrices UA and PA , initial role set CR , and threshold value MPC_{role}
Output: The matrices UA and PA satisfying the cardinality constraints
Define $count_role_permissions(r)$ as the number of permissions assigned to role, and calculate it accordingly;
Define $count_permission_roles(p)$ as the number of roles that have permission p and compute it accordingly;
For each role in CR **do**
 If $count_role_permissions(r) > MPC_{role}$ **then**
 $k = count_user_roles(u) - MPC_{role} + 1$;
 Select the top k permission sets with the highest $count_permission_roles(p)$ from role r to form set S ;
 Form a role, denoted as P_s , from set S ;
 Take the intersection of the k permission sets to create set P_k , $P_k = \{P_1 \cap P_2 \dots \cap P_k\}$
 Create a new role r_n , $role_permission(r_n) = P_s$;
 For each p_r in CR **do**
 For each p_t in P_k **do**
 If $p_r \in S, p_r \supseteq P_k$ **then**
 $PA[t][r] = 0$; // Roles are no longer granted permissions that are intersections of their previous privileges, and these privileges are instead assigned to new roles.
 $PA[n][r] = 1$;
 else
 $PA[n][r] = 0$; // New roles, being defined independently of permission intersections, are not granted such privileges.
 End if
 End for
 End for
 For each u_i in UA **do**
 If $UA[i][k] = 1$ **then**
 $UA[i][n] = 1$; // If users in UA possess roles that are intersections of permissions, they will be assigned the new role instead.
 else
 $UA[i][n] = 0$;
 End if
 End for
 Update $count_role_permissions(r)$ and $count_permission_roles(p)$;
 End for

In Algorithm 4, we initially determine the count of permissions associated with each role and the count of roles associated with each permission. Then, for each role, if its permission count surpasses MPC_{role} , we identify the k permissions with the highest $count_permission_roles(p)$ within the role. Subsequently, we compute the intersection of these k permissions and set the corresponding role indexes in PA and the permission indexes corresponding to the k permission sets to 0, effectively eliminating these permissions from their original roles. The k permission sets are amalgamated to create a new role. Concurrently, if a user in UA is linked to any of the role indexes in the intersection, the new role is allocated to this user. This approach ensures that permissions assigned to roles adhere to the constraints while minimizing the number of permissions in PA associated with the roles.

In Algorithm 4, we commence by assessing the number of permissions associated with each role and the frequency of roles assigned to each permission. When a role's permission count exceeds the threshold MPC_{role} , we identify the k permissions with the highest occurrence among that role's associated permissions based on the $count_permission_roles(p)$ function. Next, we determine the intersection of these k permissions and proceed to update PA matrix. Specifically, we set the role indexes corresponding to the intersection to 0 in the PA matrix, effectively disassociating these permissions from their original roles. Addition-

ally, we set the permission indexes corresponding to the k permission sets to 0, reflecting the removal of these permissions from the system.

Subsequently, we consolidate the k permission sets into a new role. Concurrently, if any user in UA is associated with any of the role indexes that were part of the intersection, we allocate the newly created role to that user. This approach ensures that the permissions assigned to roles adhere to the MPC_{role} constraint while minimizing the overall number of permissions in the PA matrix associated with the roles.

To illustrate this concept, let us consider an example where the threshold is set to 3, as depicted in Figure 9.

$$\begin{pmatrix} 1000 \\ 1100 \\ 1110 \\ 1011 \end{pmatrix} \otimes \begin{pmatrix} 1001 \\ 1101 \\ 0100 \\ 1111 \end{pmatrix} = \begin{pmatrix} 1001 \\ 1101 \\ 1101 \\ 1111 \end{pmatrix}$$

Figure 9. Initial RPC data.

After the algorithmic steps are executed, r_4 possesses the permissions set $(\{1, 1, 1, 1\})$, indicating that it holds more permissions than does the threshold value of 3, where $(k = 2, S = \{p_1, p_2\}, P_k = \{p_1 \cap p_2\} = \{0, 1, 0, 1\})$, and new role $r_n = P_S = \{p_1, p_2\} = \{1, 1, 0, 0\}$.

The subsequent outcome, obtained through the prescribed steps, is illustrated in Figure 10.

$$\begin{pmatrix} 10000 \\ 11001 \\ 11101 \\ 10111 \end{pmatrix} \otimes \begin{pmatrix} 1001 \\ 0001 \\ 0100 \\ 0011 \\ 1100 \end{pmatrix} = \begin{pmatrix} 1001 \\ 1101 \\ 1101 \\ 1111 \end{pmatrix}$$

Figure 10. The refined result of the RPC algorithm.

4.4. Role Assignments Satisfying Multiple Constraints

In this subsection, we integrate cardinality constraints, user exclusion constraints, and user capability constraints to derive a user-role assignment from the algorithm results that adheres to all constraints. This ensures that each user is allocated appropriate roles while maximizing the number of user-role assignments. Although another role assignment algorithm with mixed constraints has been proposed by other authors [14], they fix the number of roles in the user capability constraint UC , and their experimental results are discussed based on $\frac{|UA'|}{|UC|} \times 100\%$. Additionally, the algorithm does not guarantee that each role in the resulting user-role assignment matrix contains at least one user. If a role has zero associated users, the generated role lacks practical significance. Hence, we perceive certain limitations, such as roles generated in the dataset not participating in constraints and some roles lacking practical significance.

To address these limitations, we incorporate all roles generated in the dataset into the user capability constraint UC and randomly generate a user capability constraint UC with the same number of roles as the dataset. The experimental results are subsequently discussed based on variations in $\frac{|UA'|}{|UA|} \times 100\%$. Furthermore, while their algorithm considers user-role assignments that meet all constraints as positive values, we consider assignments that do not meet the constraints as 0. Consequently, the remaining user-role assignments are deemed accurate assignments generated by the algorithm, thereby maximizing the user retention ratio (URR). The algorithm is outlined as follows.

A role assignment problem with multiple constraints arises in the optimization process of role mining. Given the user-permission set UA , the role-user cardinality constraint MUC_{role} , a set C of user exclusion constraints in the form of t - t SMER, and the user capability constraints UC , the objective is to find a matrix UA' that maximizes the number of user-role

assignments while ensuring compliance with all constraints. The constraints to be satisfied include the following:

$$\begin{cases} \max |UA'| \\ UA' [i][j] = 0, \forall UC [i][j] = 0 \\ user_role \text{ satisfy } C, \forall u \in UA' \\ count_role_users(r) \leq MUC_{role}, \forall u \in UA' \end{cases} \quad (18)$$

In Algorithm 5, first, UC is generated. The UC with the same number of roles is randomly generated based on the UA . Then, a copy of UA is made into UA' , and UA' is initialized based on the constraint matrix UC . Subsequently, the role-user constraints are applied to UA' , where values that do not meet the constraints are set to 0. Next, the compliance of each role in UA' with the constraint set C is verified. If a role does not comply, the corresponding values in UA' belonging to the $SMER < \{r_1, r_2, \dots, r_t\}, t >$ with the lowest priority role r_y are set to 0. Finally, the UA' obtained is the matrix for maximizing role assignments satisfying multiple constraints.

Algorithm 5. Role assignments satisfying multiple constraints (RMC)

Input: threshold MUC_{role} , matrix UC , t - t $SMER$ constraints with set C , user-permission set UA

output: user-role assignment matrix UA'

$UA' = UA$;

//User capability constraint

For each $UC [i][j]$ in UC **do**

If $UC [i][j] == 0$ **then**

$UA' [i][j] = 0$;

end if

End for

Sort the roles r in UA' based on their priority, where roles with lower $SMER_c(r)$ values have higher priority. Add the sorted roles to an array RC ;

For each r_j in RC **do**

 //If the number of users in the role exceeds the threshold, perform the RUC operation again.

If $count_role_users(r) > MUC_{role}$ **then**

 Identify the set S consisting of $count_role_users(r) - MUC_{role} + 1$ users with the highest $count_user_roles(u)$;

For each s in S **do**

$UA' [s][r] = 0$;

End for

End if

For each u_i in UA' **do**

If $UA' [i][j] != 0$ **then**

For each $SMER < \{r_1, r_2, \dots, r_t\}, t >$ in C **do**

 //The assignment relationship with the lowest priority must satisfy the mutually exclusive user constraint.

If $\left| \left\{ user_role(u_i) \cup \{r_j\} \right\} \cap \{r_1, r_2, \dots, r_t\} \right| \geq t$ **then**

 Find the role r_y in $SMER < \{r_1, r_2, \dots, r_t\}, t >$ with the lowest priority.

$UA' [i][y] = 0$;

End if

End for

End if

End for

End for

5. Experimental Results and Analysis

The real datasets used in this study consist of a total of six datasets, which were sourced from the Hewlett-Packard Laboratories and have been previously used in [24]. In this study, we compared and analyze the experimental results of the original Edge-RMP

algorithm with the optimized results based on these datasets. The pertinent data for the real datasets are presented in Table 1.

Table 1. Real datasets.

Dataset	Users	Permissions	UPA
Americas_small	3477	1587	105,205
APJ	2044	1164	6841
EMEA	35	3046	7220
Healthcare	46	46	1486
Firewall1	365	709	31,951
Firewall2	325	590	36,428

Among these, $|users|$ represents the number of users, $|permissions|$ denotes the number of permissions, and $|UPA|$ indicates the relationships established for user-permission assignments. The datasets Americas_small, APJ, and EMEA were extracted from Cisco's firewall, describing the access of authorized external users to internal corporate resources. The Healthcare dataset originated from the U.S. Department of Veterans Affairs, while the Firewall1 and Firewall2 datasets were obtained as results from monitoring algorithms running at firewall monitoring points.

The software utilized in this paper was developed in Java and executed on a personal computer equipped with an AMD Ryzen 5 4600 H processor with Radeon Graphics running at a processing speed of 3.0 GHz.

5.1. Optimized Edge-RMP

The experiments conducted in this paper involved applying both the Edge-RMP algorithm and the optimized Edge-RMP algorithm to various datasets, and their performance was compared. Table 2 presents the experimental results across the six different datasets. These results confirm the effectiveness of the optimized Edge-RMP algorithm, which demonstrated the ability to handle problems more efficiently and exhibit outstanding performance on large datasets.

Table 2. Comparison between the Edge-RMP algorithm and the Optimized algorithm.

Dataset	Edge-RMP Algorithm UA + PA	Optimized Edge-RMP Algorithm UA + PA	Efficiency Improvement
Americas_small	14,732	8978	39.0%
APJ	5149	4228	17.9%
EMEA	8748	4147	52.6%
Healthcare	546	235	57%
Firewall1	3755	2129	43.3%
Firewall2	1378	1196	13.2%

First, it is worth noting that the optimized Edge-RMP algorithm significantly enhances efficiency across multiple datasets, as evidenced by the reduction in $|UA|+|PA|$ (the sum of user access frequency and permission assignment times). This demonstrates the effectiveness of the optimization algorithm in reducing system management and maintenance complexity while upholding system security.

Next, we can posit that MFC-RMA (if designed based on similar optimization principles) may also yield comparable efficiency improvements on analogous datasets. Specifically, MFC-RMA could potentially offer advantages in the following areas:

Streamlined administrative complexity: Similar to the optimized Edge-RMP algorithm, MFC-RMA has the potential to streamline system administration complexity by minimizing unnecessary user access and permission assignments. This would facilitate administrators' comprehension and maintenance of system permission settings.

Enhanced security: Through optimizing role and permission allocation, MFC-RMA may mitigate potential security risks. For instance, by curtailing superfluous permission assignments, it could reduce the likelihood of permission leakage or misuse.

Improved scalability: Optimization algorithms typically exhibit superior scalability due to their ability to efficiently handle large datasets and complex permission structures. With a similar optimization strategy, MFC-RMA might also demonstrate enhanced performance when dealing with expansive systems.

Elevated user satisfaction: By diminishing unnecessary access restrictions and refining precision in permission assignments, MFC-RMA has the potential to elevate user satisfaction levels and productivity.

Next, a comparative analysis was conducted between the HPe algorithm [23], GA_{edge} [24], and the optimized Edge-RMP algorithm in various aspects, including the number of generated roles, generated user–role relationships, the sum of roles and permissions edges, and the runtime of the algorithms. The HPe algorithm utilizes a method of rapid graph reduction based on graphs smaller than the original input to recover optimal solutions from problem instances in smaller graphs. The GA_{edge} algorithm transforms the role mining problem into the corresponding set cover problem, applies a greedy strategy to solve the set cover problem, and subsequently translates the solution of the set cover problem into the solution for the role mining problem. In contrast, the optimized Edge-RMP algorithm operates based on Boolean matrix factorization.

Table 3 presents the performance of the HPe, GA_{edge}, Edge-RMP, and optimized Edge-RMP algorithm in terms of role generation across different datasets. The experimental results indicate that the optimized Edge-RMP algorithm consistently generates fewer roles than do the other algorithms across all datasets. In some datasets, the number of roles generated by the optimized Edge-RMP algorithm is even less than half of what the other algorithms produce. Figure 11 illustrates the effectiveness of the optimized Edge-RMP algorithm in role generation. It can be observed that our algorithm excels primarily in role generation, even outperforming the HPe and GA_{edge} algorithms, particularly on the healthcare dataset where the Edge-RMP algorithm falls short. Our algorithm addresses this deficiency effectively. Table 4 displays the results for the sum of edges generated by each algorithm, and Figure 12 provides a corresponding line graph of these results.

Table 3. Role count generated by the algorithms.

Dataset	HPe	GAedge	Edge-RMP	Optimized Edge-RMP
Americas_small	258	275	196	193
APJ	471	479	465	461
EMEA	104	115	43	35
Healthcare	15	16	17	14
Firewall1	75	79	66	66
Firewall2	10	10	10	10

Table 4. Sum of edges generated by the algorithms.

Dataset	HPe	GAedge	Edge-RMP	Optimized Edge-RMP		
				UA	PA	UA + PA
Americas_small	8071	7635	14,732	5735	3243	8978
APJ	3959	3916	5149	2730	1498	4228
EMEA	3772	3722	8748	475	3672	4147
Healthcare	211	193	546	120	115	235
Firewall1	1873	1745	3755	986	1143	2129
Firewall2	1076	1046	1378	505	691	1196

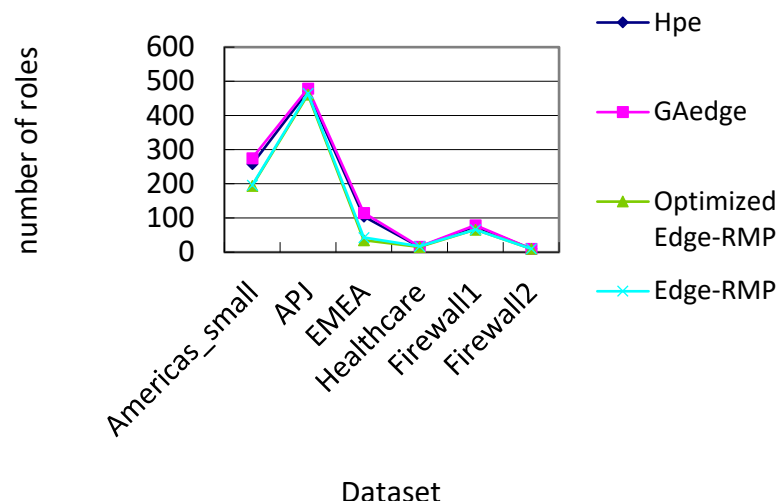


Figure 11. Diagram illustrating the role generation algorithm.

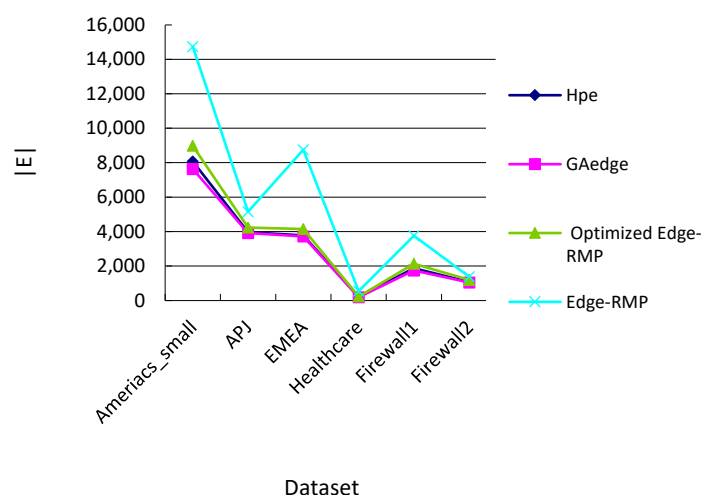


Figure 12. Graph illustrating the summation of edges.

In finding the sum of its sides, note that the Fast-Miner algorithm removes the initial repetition role, and the repetition times need to be recorded in order to more accurately find the minimum sum of $|UA| + |PA|$.

It can be observed that the optimized Edge-RMP algorithm yielded experimental results comparable to those of other algorithms across most datasets. However, compared to the pre-optimized Edge-RMP algorithm, the optimized Edge-RMP algorithm demonstrates a significantly higher efficiency improvement across various datasets. Additionally, the optimized Edge-RMP algorithm holds a significant advantage in the number of roles generated compared to other algorithms within the datasets. While the runtime of the Edge-RMP algorithm was notably shorter than that of the HPe and GA_{edge} algorithms, the optimized Edge-RMP algorithm, being an optimization of this algorithm, also exhibits significantly shorter runtime compared to the HPe and GA_{edge} algorithms. In practical applications, role mining should not only focus on the quantity of generated user-role and role-permission assignment relationships but also consider the number of roles generated to save mining costs. In comparison, the optimized Edge-RMP algorithm proposed in this paper achieves quite satisfactory results.

In summary, the optimized Edge-RMP algorithm, while showing minor variations in edge generation compared to other algorithms, offers a significant advantage in role generation and algorithm runtime. Role generation plays a crucial role in Edge-RMP problems. The optimized Edge-RMP algorithm enhances role generation, consequently reducing the

number of generated edges. However, an excessive number of roles may introduce data redundancy within the system, posing challenges for system security administrators in access control. Thus, the algorithm demonstrates a significant advantage in addressing the $\lambda_r|R| + \lambda_e(|UA| + |PA|)$ problem.

The efficiency enhancements of the enhanced algorithm exhibit variations across different datasets, influenced by the diverse distribution structures of the datasets. Certain variables may characterize the overall distribution of user-role assignment relationships within each dataset. Particularly notable is the significant improvement observed on the Healthcare dataset, attributable to its balanced distribution of users and permissions. Evidently, the algorithm performs admirably across datasets of various sizes, encompassing large, medium, and small datasets.

These experimental results underscore the disparate efficiency levels observed across different datasets, stemming from structural factors that warrant further investigation. The algorithm's outcomes may encompass low-priority roles with infrequent usage. To further refine the algorithm's results, role-permission cardinality constraints could be integrated. Moreover, RBAC confronts real-world challenges, wherein dynamic changes occur in business environments, and interactions between users and the role mining process give rise to events. This underscores the need for incremental algorithms capable of handling real-time scenarios while ensuring system stability. These insights lay the groundwork for future endeavors aimed at enhancing the efficiency of this optimized algorithm.

5.2. Experiment on Data Preprocessing

In the data preprocessing stage, the clustering compression algorithm from [14] was employed, and the similarity between roles in the compressed result and the initial roles, as well as the similarity of permissions within the same group in matrix PA ($\text{simP}(PA_{\text{compressed}}, PA_{\text{initial}})$), was verified in the study. The experimental results demonstrated that the similarity of permissions within the same group in matrix PA consistently remained high at 0.95. This indicates that for identical sets of permissions, the roles in the compressed result closely resemble the initial roles. While there may still be occasional variations in access resource permissions within large-scale application systems, differences below 0.05 can be considered acceptable. Hence, from the perspective of simP , the utilization of preprocessing methods proves to be accurate.

5.3. Cardinality Constraint Experiment

We investigated the optimization of the number of users and permissions for roles under the cardinality constraint using Algorithms 3 and 4. Subsequently, experiments were conducted on the healthcare dataset using the preprocessed matrices UA and PA , along with the initial role set CR as inputs, where the thresholds MUC_{role} and MPC_{role} were greater than 1. The experimental results were compared with those in UPRCM [7], as shown in Figures 7 and 8. Here, the x-axis represents the values of thresholds MUC_{role} and MPC_{role} , while the y-axis represents the number of roles generated under the constraints.

As observed in Figure 13, it is evident that both the UPRCM and RUC (role-user assignment optimization for role-user cardinality) algorithms display a decreasing trend in the number of roles as MUC_{role} increases, eventually stabilizing after reaching a certain threshold. Specifically, beyond a threshold of 8 for MUC_{role} , the experimental results for both algorithms become relatively consistent. At this threshold, the UPRCM algorithm generates 21 roles, while the RUC algorithm generates 18 roles. Hence, under no constraints, the maximum number of users per role is eight. Furthermore, a closer examination reveals that the UPRCM algorithm experiences drastic changes in the number of roles between thresholds 4 and 8, indicating strong constraints within this range. As the threshold increases, the number of users per role gradually rises, resulting in fewer assignable roles. In contrast, our RUC algorithm demonstrates relatively stable changes with varying thresholds, indicating fewer fluctuations in the number of roles and higher stability compared to the former. Beyond threshold 8, the number of roles stabilizes, indicating saturation

in the number of users per role. Even significant changes in the threshold thereafter do not noticeably affect the number of roles. Additionally, the figure illustrates significant variations in the number of generated roles with smaller MUC_{role} thresholds, where smaller MUC_{role} values imply stronger constraints. Consequently, more permissions are allocated to rarely used irregular roles, leading to a significant increase in the number of roles for smaller thresholds.

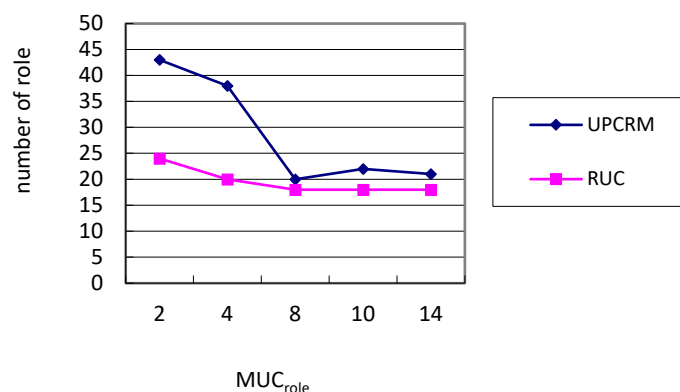


Figure 13. Role-user assignment optimization for role-user cardinality (RUC).

In summary, the RUC algorithm consistently generates fewer roles compared to the UPRCM algorithm, indicating its superiority in the healthcare dataset.

From Figure 14, it is evident that both the UPRCM and RPC (role-permission assignment optimization for role-permission cardinality) algorithms exhibit a decrease in the number of generated roles as the threshold MPC_{role} increases, eventually stabilizing after reaching a certain threshold. The UPRCM algorithm shows no change in results beyond a threshold of 26 for MPC_{role} , generating 21 roles. In contrast, the RPC algorithm approaches stability around threshold 26 for MPC_{role} , with gradual changes until reaching a stable state after a threshold of 40, generating 17 roles. The stability in the number of roles after reaching different values for threshold MPC_{role} is because some roles generated by the RPC algorithm contain multiple permissions. However, it is evident that the RPC algorithm generates fewer roles after stability. Moreover, the figure illustrates significant differences in experimental results between the two algorithms at thresholds 2–3 for MPC_{role} . However, the RPC algorithm quickly catches up with the UPRCM algorithm in generating the same number of roles for subsequent changes in threshold MPC_{role} , eventually producing a smaller number of experimental results than the UPRCM algorithm. In conclusion, the RPC algorithm demonstrates overall superiority over the UPRCM algorithm in the healthcare dataset.

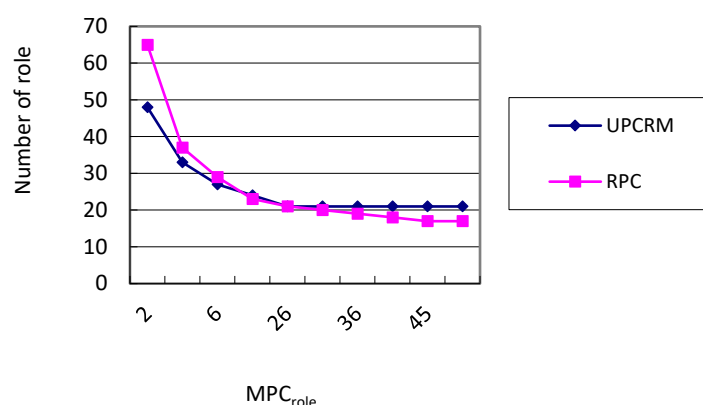


Figure 14. Role-permission assignment optimization for role-permission cardinality (RPC).

Given the scarcity of constraint experiments conducted on the basis of Edge-RMP optimization results, this study aimed to validate the experimental effectiveness through the adoption of the WSC comparison methodology. As shown in Table 5, our results were compared with the DuplicateUDCC algorithm [39] on the APJ dataset. Notably, the WSC value reported for the DuplicateUDCC algorithm represents the optimal outcome among various algorithms presented in its original paper. In computing the WSC value, it is worth mentioning that the $|DUPA|$ value is also taken into consideration. However, since its value remains constant at 0 after a threshold of 28, it is omitted from further discussion. Notably, the RUC algorithm exhibits superior overall efficiency in role mining under various thresholds compared to the DuplicateUDCC algorithm. Moreover, both algorithms share a common trait: the WSC value remains unchanged after a threshold of 112. In summary, due to the different foundations of the constraint experiments, a direct comparison based on individual variable attributes is not feasible. However, in terms of the overall performance of the constraint algorithms measured by WSC, the proposed RUC algorithm outperforms the DuplicateUDCC algorithm.

Table 5. WSC values for the APJ dataset.

MUC_{role}	Duplicate UDCC	RUC		
		Role	$ UA + PA $	WSC
28	4945	470	4349	4845
56	4881	467	4253	4720
84	4872	466	4253	4719
112	4857	465	4228	4693
139	4852	465	4228	4693
167	4852	465	4228	4693

5.4. Multiple Constraints Experiment

In this section, we describe the analysis of the impact of different parameters on the user retention ratio (URR) using various inputs such as the threshold MUC_{role} , the density ρ of the user capability matrix (percentage of 1s in the matrix), the t - t SMER constraint set C , and the preprocessed data results UA . Specifically, we randomly generated the t - t SMER constraints based on specified parameters including the minimum and maximum constraint numbers and the number of constraints in set C . Experiments were conducted on both the healthcare and APJ datasets to evaluate the URR, and the results were compared. For the healthcare dataset, which is relatively small, the number of constraints in the t - t SMER constraint set C was set to 100, 200, and 300, respectively. Additionally, for the APJ dataset, an additional set with $|C| = 400$ constraints was included to provide more comprehensive experimental results. The experimental findings are depicted in Figures 9 and 10, where the vertical axis represents the change in URR, and the horizontal axis represents variations in MUC_{role} , C , and ρ . These results provide insights into how different parameters affect the efficiency and performance of the algorithm.

In this approach, we explored the impact of varying threshold MUC_{role} values, the density ρ of the user capability matrix, and the set of user exclusion constraints C on the user retention ratio (URR). Specifically, we set the cardinality constraint value to be greater than 1.

Figure 15 illustrates the line graph of URR as the threshold MUC_{role} varies on the healthcare dataset, with a user capability matrix density of $\rho = 0.7$. The graph shows a gradual increase in URR with an increase in the threshold MUC_{role} . Notably, URR stabilizes when the threshold reaches eight, which corroborates findings from previous experiments on cardinality constraints. At $MUC_{role} = 2$, the UUR values remain nearly identical regardless of the size of the set of user exclusion constraints ($|C| = 100$, $|C| = 200$, or $|C| = 300$), indicating a minimal impact of t - t SMER constraints at this threshold. However, variations in $|C|$ consistently affect the URR values, especially in the range of 100–200 constraints. Overall, the results demonstrate that URR tends to increase with higher threshold MUC_{role} values until reaching a stable value. The influence of t - t SMER

constraint sets on URR highlights their significance in affecting the results, particularly with varying constraint sizes.

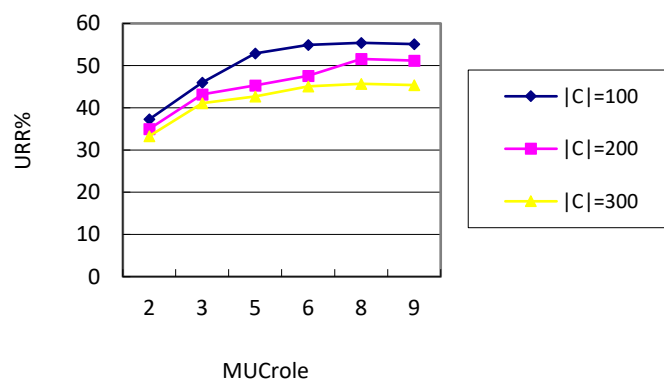


Figure 15. Performance of RMC in the healthcare dataset with a different MUC_{role} .

Figure 16 illustrates the line graph of the user retention ratio (URR) in the healthcare dataset as the density ρ of the user capability matrix UC varied. In this case, the threshold MUC_{role} was fixed at four. It is evident that URR increases linearly with increasing ρ . This is because as ρ increases, the algorithm results in more user-role assignment relationships, which is expected. Moreover, URR continues to increase with increasing ρ and does not reach a plateau until $\rho = 1$.

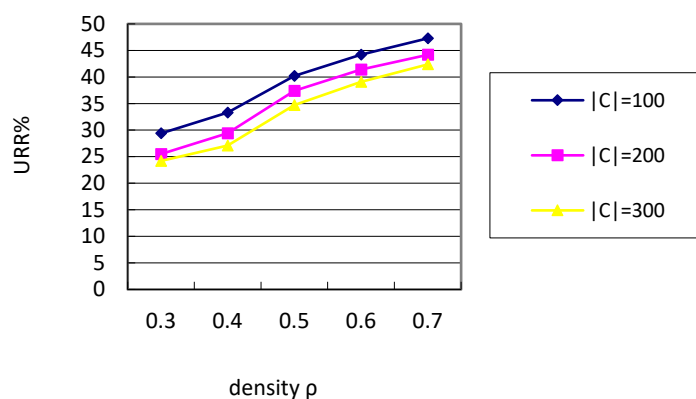


Figure 16. Performance of RMC in the healthcare dataset with a different ρ .

Figures 17 and 18 illustrate the variation of the user retention ratio (URR) on the large dataset APJ concerning changes in the threshold MUC_{role} and density ρ . In Figure 11, the density of the user capability constraint matrix is fixed at 0.7, while in Figure 12, the threshold MUC_{role} is fixed at 5. It can be observed that when the threshold MUC_{role} is two and three, the URR values for $|C| = 100$, $|C| = 200$, $|C| = 300$, and $|C| = 400$ are almost identical, indicating minimal variation, especially in large datasets where the resulting UA' matrix is sparse. This is due to the significant number of zeros in the UA matrix of large datasets, making it difficult for the user exclusion constraints to have a pronounced effect. Additionally, Figures 13 and 14 show that for $|C| = 300$ and $|C| = 400$, under different parameters of threshold MUC_{role} and density ρ , they achieve nearly identical URR values. This suggests that the impact of the number of $t-t$ SMER constraints on URR is smaller in large datasets compared to small datasets. Overall, in sparse large datasets, user-role assignment relationships are less influenced by the quantity of user exclusion cons

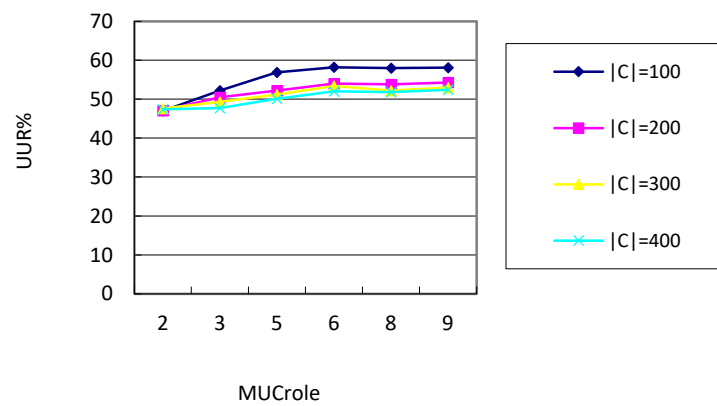


Figure 17. Performance of RMC in the APJ dataset with a different MUC_{role} .

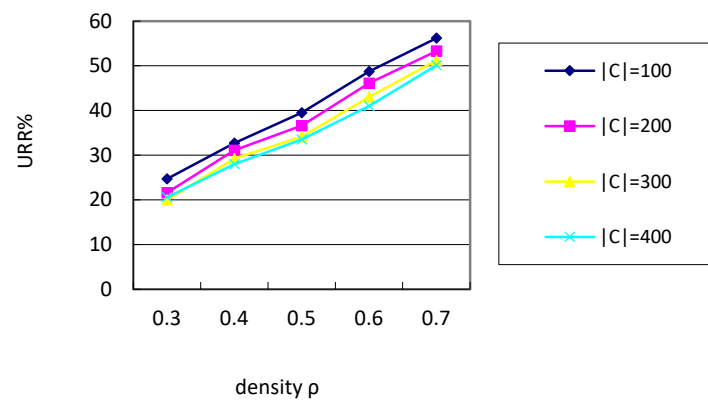


Figure 18. Performance of RMC in the APJ dataset with a different p .

The methodologies introduced by Kumar et al. [29] and Blundo et al. [30] exclusively addressed the cardinality constraint RPC, while Hingankar et al.'s approach [31] solely tackled the RUC constraint. On the other hand, John et al.'s CPA and RPA [43] were designed to fulfill the UCC constraint. Ma et al.'s method [7] was capable of satisfying either the RUC or RPC constraint. Sarana et al.'s techniques [33] did not adhere to any cardinality constraints but met the SMER constraints. Conversely, Harika et al.'s methodologies [32] achieved simultaneous compliance with UCC and PCC. It is worth noting that the system's status remained unknown with the utilization of these methodologies. Although Roy et al.'s method [37] concurrently addressed UCC, SMER, and user-capability constraints, it was built upon an existing RBAC system. For comparison with other existing studies, we present a table where methods are marked with a check mark (✓) if they exist. Based on the Table 6, it can be seen that our algorithm exhibits superiority.

Table 6. Related work.

Characteristic	Kumar et al. [29] Blundo et al. [30]	Hingankar et al. [31]	John et al. [43]	Ma et al. [7]	Sarana et al. [33]	Harika et al. [32]	Roy et al. [37]	Proposed Method
UCC		✓	✓			✓	✓	
PCC						✓		
RUC				✓				✓
RPC	✓			✓				✓
SMER					✓		✓	✓
User Capability							✓	✓
System Status	✓	✓	✓	✓	✓			✓

6. Discussion

By optimizing the Edge-RMP algorithm, we have successfully reduced the mining scale and computational complexity while satisfying dual cardinality constraints in role optimization and multiple security constraints in role assignment. This enhancement not only improves system efficiency but also ensures heightened security and reliability. As enterprises and organizations' demands for access control continue to grow, our research findings are pivotal in enhancing the overall performance of RBAC systems.

The MFC-RMA algorithm has excelled in numerous aspects. Through comparisons with the original Edge-RMP algorithm, the graph-based HPe algorithm, and the set-cover-based GA_{edge} algorithm, we have demonstrated that the MFC-RMA algorithm possesses significant advantages in terms of the number of edges and roles generated. This superiority is not merely quantitative; the quality of the generated roles aligns more closely with real-world needs, better satisfying user capabilities and mutual exclusion constraints.

We have introduced a constraint-based role optimization problem and designed corresponding algorithms to tackle it. This approach addresses the limitations of traditional role mining algorithms when dealing with large-scale datasets while also enhancing the flexibility and scalability of our algorithms. Furthermore, our constraint-based algorithm was compared with the recent DuplicateUDCC algorithm, exhibiting its superior comprehensive performance.

Additionally, we delved into role mining under multiple constraints, providing novel insights and methodologies for future research. Our research findings are of utmost importance in improving and optimizing RBAC systems. As enterprises and organizations' requirements for information security and data protection increase, traditional RBAC systems are becoming insufficient. By introducing constraints and optimizing role mining algorithms, we provide valuable support for the further development and refinement of RBAC systems. Furthermore, our research offers new inspirations and references for studies in related fields.

7. Conclusions

The paper begins by introducing RBAC and elaborates on the fundamental and variant problems of role mining. It discusses the concepts of role engineering and Boolean matrix decomposition, leading to the introduction of the Edge-RMP algorithm, which is relevant to the paper. The shortcomings of the Edge-RMP algorithm in terms of edge and role generation are pointed out, prompting the proposed improvements in this study. Through in-depth analysis and optimization of the Edge-RMP algorithm, our research has introduced the MFC-RMA algorithm, which significantly enhances the performance and efficiency of role mining. By introducing constraint conditions, the MFC-RMA algorithm optimizes the role mining process, achieving performance gains over the DuplicateUDCC algorithm. It effectively reduces the mining scale and computational complexity, satisfying the cardinality constraints in role optimization and multiple security constraints in role assignment, ultimately generating the desired roles. Through comparisons between the original and enhanced algorithms on real datasets, the study demonstrates the superior performance of the MFC-RMA algorithm in improving the Edge-RMP algorithm. Furthermore, comparisons with graph-based HPe algorithm and set-covering-based GA_{edge} algorithm demonstrate significant advantages of the Edge-RMP optimized algorithm in terms of both the number of generated edges and the number of generated roles. Experimental results also validate the superiority of the cardinality constraint design algorithm in the MFC-RMA algorithm compared to other algorithms. Lastly, this paper explores role mining under multiple constraints and discusses the role outcomes under different parameters.

In practical applications, the algorithm first generates initial roles through the Fast Miner algorithm, which is followed by preliminary processing with the Edge-RMP algorithm. After that, it is integrated into our MFC-RMA algorithm to generate user roles and role-permission assignment relationships that satisfy Edge RMP, as well as further processed constraint roles to meet specific needs. Experiments proved that the algorithm

exhibits high individual and comprehensive performance in generating the number of roles, user-role, and role-permission relationships in RBAC, both under constrained and unconstrained conditions. As an effective role assignment strategy, it significantly reduces the direct association between users and permissions, thus minimizing security risks faced by RBAC systems, which holds significant practical importance.

However, there are some limitations in the current research.

(1) Randomness of constraints. In the algorithmic work presented here, there is a significant randomness in generating the user capability constraint matrix (UC) and the user exclusion constraint set (C). While the algorithm takes the average of the generated results, it still leads to a certain degree of error in the generated roles. In the future, additional parameters or rules can be introduced during the generation of UC and C to increase stability and consistency. For example, fixed rules or weights can be defined to reduce randomness while maintaining flexibility.

(2) Role interpretability. The interpretability of roles makes permission assignment more transparent and understandable. Users and administrators can clearly understand the permission scope and access rules of each role, thus gaining a better understanding of the system's security policies and access restrictions.

(3) Adaptive access control systems. Future role mining and role engineering techniques may incorporate intelligent algorithms and machine learning to achieve adaptive access control systems. Through continuous learning and adjustment, the system can automatically adapt to changing environments and needs, providing more personalized and precise access control services.

In summary, the MFC-RMA algorithm is an effective solution for Edge RMP. By comprehensively considering the relationships between users, roles, and permissions as well as different types of constraints, it significantly optimizes the results of Edge RMP. For modern organizations that require efficient management of permissions within the organization, the MFC-RMA algorithm provides powerful technical support.

Author Contributions: Conceptualization, F.Z.; methodology, F.Z.; formal analysis, J.G. and H.Z.; implementation of algorithms, C.Y.; data management, J.G. and H.Z.; writing, C.Y. and L.Z.; reviewing and editing of original drafts, F.Z. and L.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (NSFC) under grant no. 61902361, in part by the Key Research and Development Special Project of Henan Province (221111210500), and by the Henan Postgraduate Joint Training Base Project (no. YJS2022JD08).

Data Availability Statement: The data used to support the findings of the study are available within the article.

Conflicts of Interest: Author Hongqiang Zuo and Jingzhong Gu were employed by the company Shangu Cyber Security Technology Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Sun, W.; Su, H.; Xie, H. Policy-Engineering Optimization with Visual Representation and Separation-of-Duty Constraints in Attribute-Based Access Control. *Future Internet* **2020**, *12*, 164. [\[CrossRef\]](#)
2. Batra, G.; Atluri, V.; Vaidya, J.; Sural, S. Deploying ABAC policies using RBAC systems. *J. Comput. Secur.* **2019**, *27*, 483–506. [\[CrossRef\]](#) [\[PubMed\]](#)
3. Ghafoorian, M.; Abbasinezhad-Mood, D.; Shakeri, H. A Thorough Trust and Reputation Based RBAC Model for Secure Data Storage in the Cloud. *IEEE Trans. Parallel Distrib. Syst.* **2019**, *30*, 778–788. [\[CrossRef\]](#)
4. Coyne, E.J. Role engineering. In Proceedings of the First ACM Workshop on Role-Based Access Control, Gaithersburg, MD, USA, 30 November–2 December 1996; p. 4.
5. Vaidya, J.; Atluri, V.; Guo, Q. The role mining problem: Finding a minimal descriptive set of roles. In Proceedings of the 12th ACM Symposium on Access Control Models and Technologies, Sophia Antipolis, France, 20–22 June 2007; pp. 175–184.
6. Vaidya, J.; Atluri, V.; Guo, Q.; Lu, H. Edge-RMP: Minimizing administrative assignments for role-based access control. *J. Comput. Secur.* **2009**, *17*, 211–235. [\[CrossRef\]](#)

7. Ma, X.; Li, R.; Wang, H.; Li, H. Role mining based on permission cardinality constraint and user cardinality constraint. *Secur. Commun. Netw.* **2015**, *8*, 2317–2328. [[CrossRef](#)]
8. Li, N.; Tripunitara, M.V.; Bizri, Z. On mutually exclusive roles and separation-of-duty. *ACM Trans. Inf. Syst. Secur.* **2007**, *10*, 5. [[CrossRef](#)]
9. Narouei, M.; Takabi, H. Towards an Automatic Top-down Role Engineering Approach Using Natural Language Processing Techniques. In Proceedings of the 20th ACM Symposium on Access Control Models and Technologies, Vienna, Austria, 1–3 June 2015; pp. 157–160.
10. Thomsen, D.; Brien, D.O.; Bogle, J. Role based access control framework for network enterprises. In Proceedings of the 14th Annual Computer Security Applications Conference (Cat. No.98EX217), Phoenix, AZ, USA, 7–11 December 1998; pp. 50–58.
11. Kuhlmann, M.; Shohat, D.; Schimpf, G. Role mining—Revealing business roles for security administration using data mining technology. In Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies, Como, Italy, 2–3 June 2003; pp. 179–186.
12. Schlegelmilch, J.; Steffens, U. Role mining with ORCA. In Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies, Stockholm, Sweden, 1–3 June 2005; pp. 168–176.
13. Vaidya, J.; Atluri, V.; Warner, J. RoleMiner: Mining roles using subset enumeration. In Proceedings of the 13th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 30 October–3 November 2006; pp. 144–153.
14. Sun, W.; Su, H.; And Liu, H.B. Role-Engineering Optimization with Cardinality Constraints and User-Oriented Mutually Exclusive Constraints. *Information* **2019**, *10*, 11. [[CrossRef](#)]
15. Fuchs, L.; Pernul, G. HyDRo—Hybrid Development of Roles. In Proceedings of the 4th International Conference on Information Systems Security, Hyderabad, India, 16–20 December 2008.
16. Frank, M.; Streich, A.P.; Basin, D.; Buhmann, J.M. A probabilistic approach to hybrid role mining. In Proceedings of the 16th ACM Conference on Computer and Communications Security, Chicago, IL, USA, 9–13 November 2009; pp. 101–111.
17. Molloy, I.; Chen, H.; Li, T.; Wang, Q.; Li, N.; Bertino, E.; Calo, S.; Lobo, J. Mining Roles with Multiple Objectives. *ACM Trans. Inf. Syst. Secur.* **2010**, *13*, 36. [[CrossRef](#)]
18. Mitra, B.; Sural, S.; Vaidya, J.; Atluri, V. A Survey of Role Mining. *ACM Comput. Surv.* **2016**, *48*, 50. [[CrossRef](#)]
19. Jiang, J.; Yuan, X.; Mao, R. Research on Role Mining Algorithms in RBAC. In Proceedings of the 2018 2nd High Performance Computing and Cluster Technologies Conference, Beijing, China, 22–24 June 2018; pp. 1–5.
20. Trnecka, M.; Trneckova, M. An incremental algorithm for the role mining problem. *Comput. Secur.* **2020**, *94*, 101830. [[CrossRef](#)]
21. Anderer, S.; Kempter, T.; Scheuermann, B.; Mostaghim, S. Dynamic Optimization of Role Concepts for Role-Based Access Control Using Evolutionary Algorithms. *SN Comput. Sci.* **2023**, *4*, 416. [[CrossRef](#)]
22. Lu, H.; Vaidya, J.; Atluri, V. Optimal Boolean Matrix Decomposition: Application to Role Engineering. In Proceedings of the 2008 IEEE 24th International Conference on Data Engineering, Cancun, Mexico, 7–12 April 2008; pp. 297–306.
23. Ene, A.; Horne, W.; Milosavljevic, N.; Rao, P.; Schreiber, R.; Tarjan, R.E. Fast exact and heuristic methods for role minimization problems. In Proceedings of the 13th ACM Symposium on Access Control Models and Technologies, Estes Park, CO, USA, 11–13 June 2008; pp. 1–10.
24. Huang, H.J.; Shang, F.; Liu, J.L.; Du, H. Handling least privilege problem and role mining in RBAC. *J. Comb. Optim.* **2015**, *30*, 63–86. [[CrossRef](#)]
25. Colantonio, A. Visual Role Mining: A Picture Is Worth a Thousand Roles. *IEEE Trans. Knowl. Data Eng.* **2012**, *24*, 1120–1133. [[CrossRef](#)]
26. Verde, N.V.; Vaidya, J.S.; Atluri, V.; Colantonio, A. Role engineering: From theory to practice. In Proceedings of the 2nd ACM Conference on Data and Application Security and Privacy, San Antonio, TX, USA, 7–9 February 2012.
27. Rajaraman, A.; Ullman, J.D. *Mining of Massive Datasets*; Cambridge University Press: Cambridge, UK, 2011.
28. Belohlavek, R.; Trnecka, M. A new algorithm for Boolean matrix factorization which admits overcovering. *Discret. Appl. Math.* **2018**, *249*, 36–52. [[CrossRef](#)]
29. Kumar, R.; Sural, S.; Gupta, A. Mining RBAC Roles under Cardinality Constraint. In Proceedings of the 6th International Conference on Information Systems Security, Gandhinagar, India, 17–19 December 2010; pp. 171–185.
30. Blundo, C.; Cimato, S. Constrained Role Mining. In Proceedings of the Security and Trust Management 8th International Workshop, Pisa, Italy, 13–14 September 2012; pp. 289–304.
31. Hingankar, M.; Sural, S. Towards role mining with restricted user-role assignment. In Proceedings of the 2011 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE), Chennai, India, 28 February–3 March 2011.
32. Harika, P.; Nagajyothi, M.; John, J.C.; Sural, S.; Vaidya, J.; Atluri, V. Meeting Cardinality Constraints in Role Mining. *IEEE Trans. Dependable Secur. Comput.* **2015**, *12*, 71–84. [[CrossRef](#)]
33. Sarana, P.; Roy, A.; Sural, S.; Vaidya, J.; Atluri, V. Role Mining in the Presence of Separation of Duty Constraints. In Proceedings of the 11th International Conference on Information Systems Security, Kolkata, India, 16–20 December 2014.
34. Sun, W.; Wei, S.; Guo, H.; Liu, H. Role-Mining Optimization with Separation-of-Duty Constraints and Security Detections for Authorizations. *Future Internet* **2019**, *11*, 201. [[CrossRef](#)]

35. Roy, A.; Sural, S.; Majumdar, A.K. Impact of Multiple t - t SMER Constraints on Minimum User Requirement in RBAC. In Proceedings of the 10th International Conference on Information Systems Security, Hyderabad, India, 16–20 December 2014.
36. Roy, A.; Sural, S.; Majumdar, A.K.; Vaidya, J.; Atluri, V. Minimizing Organizational User Requirement while Meeting Security Constraints. *ACM Trans. Manag. Inf. Syst.* **2015**, *6*, 1–25. [[CrossRef](#)]
37. Roy, A.; Sural, S.; Majumdar, A.K.; Vaidya, J.; Atluri, V. On Optimal Employee Assignment in Constrained Role-Based Access Control Systems. *ACM Trans. Manag. Inf. Syst.* **2016**, *7*, 10. [[CrossRef](#)]
38. Blundo, C.; Cimato, S.; Siniscalchi, L. Role Mining Heuristics for Permission-Role-Usage Cardinality Constraints. *Comput. J.* **2021**, *65*, 1386–1411. [[CrossRef](#)]
39. Blundo, C.; Cimato, S. Role mining under User-Distribution cardinality constraint. *J. Inf. Secur. Appl.* **2023**, *78*, 103611. [[CrossRef](#)]
40. Valsesia, D.; Fosson, S.M.; Ravazzi, C.; Bianchi, T.; Magli, E. Analysis of SparseHash: An efficient embedding of set-similarity via sparse projections. *Pattern Recognit. Lett.* **2019**, *128*, 93–99. [[CrossRef](#)]
41. Guo, Q.; Tripunitara, M. The secrecy resilience of access control policies and its application to role mining. In Proceedings of the 27th ACM on Symposium on Access Control Models and Technologies, New York, NY, USA, 8–10 June 2022; pp. 115–126.
42. Li, Z.; Wang, G.; He, G. Milling tool wear state recognition based on partitioning around medoids (PAM) clustering. *Int. J. Adv. Manuf. Technol.* **2017**, *88*, 1203–1213. [[CrossRef](#)]
43. John, J.C.; Sural, S.; Atluri, V.; Vaidya, J.S. Role Mining under Role-Usage Cardinality Constraint. In Proceedings of the IFIP TC 11 Information Security and Privacy Conference, Heraklion, Greece, 4–6 June 2012.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.