International Journal of Innovative Computing, Information and Control
Volume **18**, Number **4**, August **2022**

ICIC International ©2022 ISSN 1349-4198
pp. **1037–1053**

CONSTRAINED ROLE-ENGINEERING OPTIMIZATION USING BOOLEAN MATRIX DECOMPOSITION AND INTEGER LINEAR PROGRAMMING TECHNIQUES

Wei Sun

School of Computer and Information Technology
Xinyang Normal University
No. 237, Nanhu Road, Xinyang 464000, P. R. China
sunny810715@xynu.edu.cn

Received December 2021; revised April 2022

Abstract. *Role-based access control (RBAC) is a widely popular access control mechanism because of its convenience for authorization administration, as well as various security policies, such as the separation-of-duty constraint and cardinality constraint. In recent few years, role-engineering technology has emerged as an efficient approach to constructing optimal RBAC systems. However, the bottom-up approaches lack flexibility and extendibility as the organizational requirements change dynamically. Furthermore, most conventional methods do not consider multiple cardinality constraints. To address these issues, this paper proposes a novel role-engineering method. First, to flexibly meet diverse organizational requirements while enhancing the security of role-engineering processes, according to different evaluation measures or optimization objectives, we define several variants of the optimization problem with the cardinality constraint using Boolean matrix decomposition technique. Second, we present a unified modelling framework for these variants using integer linear programming technique, and propose heuristic optimization algorithms in the bottom-up way, in order to verify whether the constraints can be satisfied in the constructed access control model. The experimental evaluations demonstrate the effectiveness and efficiency of the proposed method.*
Keywords: Role-based access control, Role engineering, Cardinality constraint, Boolean matrix decomposition, Integer linear programming

1. **Introduction.** With the rapid development and comprehensive application of network information technology, there are large amounts of information storages as well as frequent data exchanges in large-scale and complex management information systems [1]. Role-based access control (RBAC) mechanism, which is featured by its convenience for authorization administration and various security policies, is widely accepted and adopted in organizations of different sizes [2-7]. With the successful deployments of RBAC systems, accomplishing the task of devising a complete, correct and effective role set and constructing a good RBAC system become more and more crucial. As a solution to facilitate the process of migrating from a non-RBAC system to an RBAC system, role-engineering technology has been proposed in the literature, which consists of two main approaches: The top-down [8], and bottom-up [9].

The top-down role-engineering approach defines particular roles for job responsibilities, and decomposes them into smaller units by analyzing the business processes in detail. Once the required privileges for performing specific tasks are identified, they are grouped into appropriate functional roles [10]. This process is repeated until all the job functions

DOI: 10.24507/ijicic.18.04.1037

are covered. Such technique discovers roles that can well reflect the functional requirements of organizations. However, it is very complex and tedious to identify the access privileges from a large number of business processes, when there are thousands of users and millions of permissions. On the other hand, the bottom-up approach, also known as role mining, starts from the existing user-permission assignments, and then aggregates them into roles by applying data mining techniques. It has received considerable attention in recent years, and numerous role mining approaches have been developed subsequently [11-14]. The role-mining problem can be viewed as an instantiation of Boolean matrix decomposition [15,16]. Although a Boolean matrix can be decomposed in various ways, different optimal decomposition solutions may match various semantics with respect to different criteria. Different RMP variant can help pick and choose the mining results that perfectly suit a particular organizational need. However, most existing role mining methods lack flexibility and extendibility, and cannot handle all these RMP variants in a unified way. System engineers need to deal with each problem variant individually, and these methods cannot mine valuable or meaningful results when the organizational needs change dynamically.

A key characteristic of the RBAC mechanism is that it allows the specification and enforcement of various security policies [17], which can reflect the constraint requirements of organizations while ensuring system security. The commonly used constraint in actual scenarios is referred to as the cardinality constraint. It involves the user-to-role cardinality constraint (URCC), permission-to-role cardinality constraint (PRCC), role-to-user cardinality constraint (RUCC) and role-to-permission cardinality constraint (RPCC), which restrict the maximum number of roles assigned to a user, the maximum number of roles to which a permission can be assigned, the maximum number of permissions assigned to a role, and the maximum number of users to which a role can be assigned, respectively [18]. For example, the general-manager role in a company is only assigned to one person; ordinary users cannot possess too many roles; otherwise there is the possibility of the abuse of privileges. Moreover, meeting one cardinality constraint should not violate another one in the constrained role-engineering process. Specifically, if any role r meets the URCC constraint in the optimization process, then any permission assigned to r should not violate the PRCC constraint. Similarly, if any role r meets the PRCC constraint, then any user possessing r should not violate the URCC constraint. In the approaches for role engineering with cardinality constraints, however, most existing methods do not consider the URCC and PRCC constraints simultaneously.

To address the above-mentioned issues, this paper proposes a novel method, called constrained role-engineering optimization using Boolean matrix decomposition and integer linear programming techniques (CREO_BMD&ILP). In summary, the main contributions of this work are as follows.

1) To flexibly suit the organizational needs, while enhancing the security of role engineering, we convert the role-engineering problem into the Boolean matrix decomposition (BMD) problem, and define several variants of the optimization problem with multiple cardinality constraints, according to different optimization objectives.
2) We present a unified modelling representation for all these variants using integer linear programming technique (ILP). To verify whether the cardinality constraints can be satisfied in the constructed RBAC model, we propose heuristic algorithms to uniformly solve the optimization problems, and evaluate its performance using synthetic datasets.

The rest of the paper is organized as follows. In Section 2, we discuss the related work and present some necessary preliminaries. In Section 3, we propose a novel constrained role-engineering method, and present heuristic algorithms. We implement the experiments

and present the performance evaluations and comparisons in Section 4. Section 5 concludes the paper and discusses future work.

2. Related Work and Preliminaries.

2.1. Methods of role engineering.

According to whether or not constraints are taken into account, the state-of-the-art studies are further divided into two categories: The unconstrained role engineering, and role engineering with constraints.

Vaidya et al. [15] converted the role mining problem into a Boolean matrix decomposition problem, and presented a definition for the basic role mining problem (basic RMP), which aims to discover a minimal set of roles that fully cover all the permission assignments. Lu et al. [16] used the methods of Boolean matrix decomposition and integer linear programming to propose a unified framework for several variants of role mining, including the basic RMP, δ-approx RMP, edge-RMP and min-noise RMP, and presented the heuristic algorithms for solving these problems. However, the mining processes are complex and the mining scales are very large. To reduce the complexity of solving problems, Colantonio et al. [19] separated the dataset of user-permission assignments into several subsets. To reduce the mining scale, Verde et al. [20] converted the role mining problem into a clustering problem, which compresses each single partition into a cluster, extracts similar features from different clusters, and ensures the integrity of the mining results. The constraints play a crucial role for ensuring system security. However, none of these methods take consideration of constraints in the role engineering processes.

In order to limit the maximum number of users or that of permissions related to a role, Ma et al. [18] proposed a role mining algorithm to generate roles using the permission cardinality constraints and user cardinality constraints. In order to limit the maximum number of roles assigned to a user and a related permission simultaneously, Harika et al. proposed two optimization methods for role mining: The post processing and concurrent processing. The former mines roles without considering the cardinality constraints, and constructs the initial user-role and role-permission assignments. Then these assignment relationships are checked for constraint violation in the optimization phase and appropriately re-assigned, if necessary [21]. The latter implements the optimization with the double cardinality constraints, during the process of role mining. Blundo et al. [22] comprehensively examined different types of cardinality constraints, defined the constrained mining problem for each constraint type, and presented efficient heuristics for solving these problems. Subsequently, to restrict the number of roles assigned to a user and the number of permissions assigned to a role simultaneously, they presented two heuristics to produce roles compliant with both the constraints [23]. In addition, other constraints are also considered in the literature. Sarana et al. [24] proposed three methods for role optimization with the separation-of-duty constraints, during or after the mining process. To eliminate the redundancy of the mining roles, while satisfying single or double cardinality constraints, Sun et al. [25] used the cluster partitioning and compressing technologies, and proposed a novel role-mining optimization method.

Obviously, there are two main drawbacks in the existing studies. The first drawback is that, the bottom-up approaches lack flexibility and extendibility, and they cannot generate valuable or meaningful roles when the organizational requirements change dynamically. The second is that most existing studies do not consider the URCC and PRCC constraints simultaneously. In this work, we propose a novel role-engineering optimization method, called CREO_BMD&ILP, in order to flexibly satisfy different optimization objectives for organizational requirements, while ensuring the system security. We also evaluate the performance of the proposed method using the synthetic datasets.

2.2. **Preliminaries.** Before actually proposing our novel method, we present some preliminaries that are discussed in this paper, including the basic components of role engineering, several representative RMP variants, and cardinality constraints.

2.2.1. *Basic components of role engineering.* According to the NIST standard of RBAC, conventional role engineering consists of the following basic components:

1) U, P and R are the basic set elements of RBAC, which represent the sets of users, permissions, and roles, respectively;
2) $UPA \subseteq U \times P$, which represents a many-to-many mapping relationship of user-permission assignments;
3) $URA \subseteq U \times R$, which represents a many-to-many mapping relationship of user-role assignments;
4) $RPA \subseteq R \times P$, which represents a many-to-many mapping relationship of role-permission assignments;
5) $user\_roles(u) = \{r | \exists r \in R : (u, r) \in URA\}$, which represents the set of roles assigned to user u;
6) $role\_users(r) = \{u | \exists u \in U : (u, r) \in URA\}$, which represents the set of users associated with role r;
7) $perm\_roles(p) = \{r | \exists r \in R : (r, p) \in RPA\}$, which represents the set of roles associated with permission p;
8) $role\_perms(r) = \{p | \exists p \in P : (r, p) \in RPA\}$, which represents the set of permissions assigned to role r;
9) $user\_perms(u) = \{p | \exists p \in P, \exists r \in R : ((u, r) \in URA) \wedge ((r, p) \in RPA)\}$, which represents the set of permissions assigned to user u in RBAC;
10) $perm\_users(p) = \{u | \exists u \in U, \exists r \in R : ((u, r) \in URA) \wedge ((r, p) \in RPA)\}$, which represents the set of users associated with permission p in RBAC.

For convenience, the UPA, URA, and RPA are also used to represent Boolean matrices corresponding to their respective assignment relationships.

2.2.2. *Several representative RMP variants.* According to different constraint conditions and objectives in the mining process, we consider four different RMP variants with no constraints, including the basic RMP, δ-approx RMP, edge-RMP, and min-noise RMP. They can be converted into the BMD problem as follows [16].

The basic RMP states that, given a set U of users, a set P of permissions, and a matrix UPA of user-permission assignments, find a set R of roles, a matrix URA of user-role assignments, and a matrix RPA of role-permission assignments, such that the UPA can be precisely Boolean decomposed as the URA and RPA, and the number of the mining roles is minimum. This process is formalized as follows:

$$\begin{cases} \min |R| \\ URA \otimes RPA = UPA \end{cases} \tag{1}$$

Different from the basic RMP, the δ-approx RMP allows the matrix reconstruction with an error value δ when the UPA is approximately Boolean decomposed as the URA and RPA, which is formalized as follows:

$$\begin{cases} \min |R| \\ ||URA \otimes RPA - UPA||_1 \leq \delta \end{cases} \tag{2}$$

Different from the basic RMP, the edge-RMP aims to minimize the summing of the number of the user-role assignments and that of the role-permission assignments, which is formalized as follows:

$$\begin{cases} \min(|URA| + |RPA|) \\ URA \otimes RPA = UPA \end{cases} \tag{3}$$

At last, the min-noise RMP aims to minimize the reconstruction error, while considering that the number of the mining roles is less than a fixed value k, which is formalized as follows:

$$\begin{cases} \min ||URA \otimes RPA - UPA||_1 \\ |R| \leq k \end{cases} \tag{4}$$

Although there are similarities and differences towards the constraint conditions and mining objectives for all these RMP variants, the basic RMP is the primary RMP variant. The Fast Miner method, which is used to solve the basic RMP, mainly consists of the following steps:

1) According to the hash mapping rule, a given access control matrix is converted into the user-permission assignment relationship;
2) To reduce the role-engineering scale, different users who have the same permissions in the permission assignments are grouped together, and an initial set of roles is constructed.

All the potentially interesting roles are identified by implementing intersections between any pair of the initial roles, and they are regarded as new candidate roles.

2.2.3. *Cardinality constraint.* Two typical cardinality constraints, URCC and PRCC, which have been proven to be mutually exclusive [21], are taken into consideration.

The URCC states that, given a set U of users, a set R of roles, and threshold MRC_{user}, the maximum number of roles assigned to any user should not exceed MRC_{user}. This can be formalized as follows:

$$\forall u \in U : |user\_roles(u) \cap R| \leq MRC_{user} \tag{5}$$

The PRCC states that, given a set P of permissions, a set R of roles, and threshold MRC_{perm}, the maximum number of roles to which any permission can be assigned should not exceed MRC_{perm}. This can be formalized as follows:

$$\forall p \in P : |perm\_roles(p) \cap R| \leq MRC_{perm} \tag{6}$$

3. **Proposed Method.** In this section, we propose a novel research method, named as CREO_BMD&ILP, which is twofold: 1) Definitions for different constrained RMP variants via Boolean matrix decomposition, and 2) modelling representation for all these RMP variants via integer linear programming in a unified form. Specifically, we first construct an unconstrained role engineering system according to the initial access control matrix. Subsequently, we convert the role-engineering problem with the URCC and PRCC into the BMD problem, and use the 0-1 programming method to uniformly model different constrained RMP variants. Last, we implement the role-engineering optimization, and present heuristic algorithms to further verify whether the cardinality constraints can be satisfied in the constructed model. An overall view of the proposed framework is shown in Figure 1.

3.1. **Problem definitions.** Based on the basic RMP, δ-approx RMP, edge-RMP and min-noise RMP, as well as the cardinality constraints URCC and PRCC, we present the definitions for the optimization problems using the BMD method as follows.

Definition 3.1.
(Basic role-engineering optimization problem, basic $REOP_{URCC\&PRCC}$). Given a matrix $UPA_{n\times m}$ of user-permission assignments, the initial Boolean decomposed matrices for the $UPA_{n\times m}$, and two particular constraint thresholds MRC_{user} and MRC_{perm}, find two

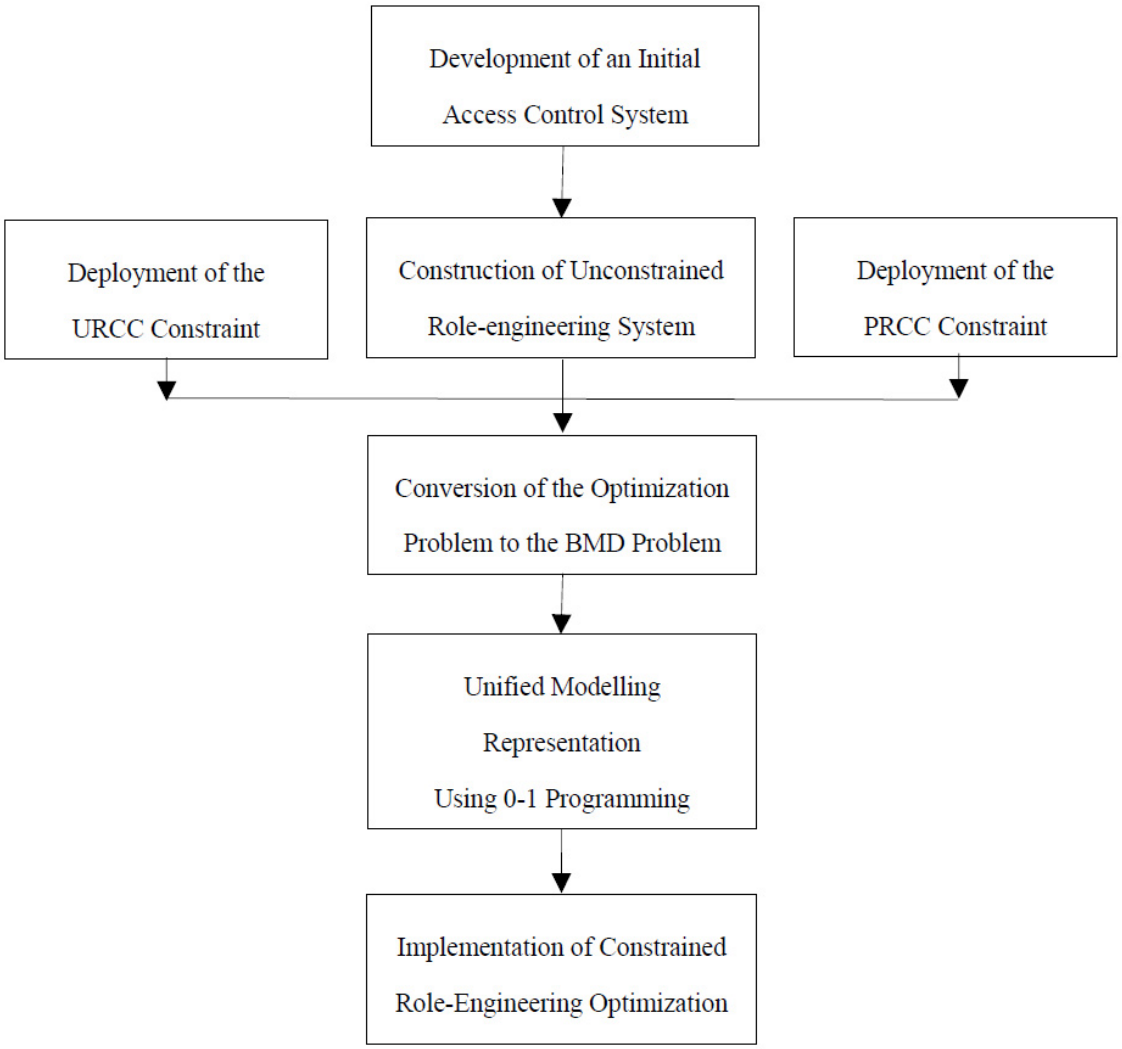


FIGURE 1. Overview of the proposed role-engineering optimization framework

matrices $URA_{n\times k}$ *and* $RPA_{k\times m}$ *such that, 1) the number* k *of the optimal roles, which is regarded as the optimization objective, should be minimum; 2) the* $UPA_{n\times m}$ *is precisely Boolean decomposed as the* $URA_{n\times k}$ *and* $RPA_{k\times m}$ *to ensure the integrity of the matrix reconstruction, the number of roles assigned to any user is less than or equal to* MRC_{user}*, and/or the number of roles to which any permission can be assigned is less than or equal to* MRC_{perm}*, which are together regarded as the constraint conditions. This process can be formalized as follows:*

$$\begin{cases} \min k \\ URA_{n\times k} \otimes RPA_{k\times m} = UPA_{n\times m} \\ \sum\limits_{j} URA[i][j] \leq MRC_{user} \leq k,\ \forall i \in [1,n] \\ \sum\limits_{j} RPA[j][t] \leq MRC_{perm} \leq k,\ \forall t \in [1,m] \end{cases} \tag{7}$$

Definition 3.2.
(δ*-approx role-engineering optimization problem,* δ*-approx* $REOP_{URCC\&PRCC}$*). Compared to Definition 3.1, the integrity constraint becomes that, the* $UPA_{n\times m}$ *can be approximately reconstructed by the* $URA_{n\times k}$ *and* $RPA_{k\times m}$ *with a reconstruction error* δ*. This process can be formalized as follows:*

$$\begin{cases} \min k \\ ||URA_{n\times k} \otimes RPA_{k\times m} - UPA_{n\times m}||_1 \leq \delta \\ \sum\limits_j URA[i][j] \leq MRC_{user} \leq k,\ \forall i \in [1,n] \\ \sum\limits_j RPA[j][t] \leq MRC_{perm} \leq k,\ \forall t \in [1,m] \end{cases} \tag{8}$$

Definition 3.3.
(Edge-role-engineering optimization problem, edge-REOP$_{URCC\&PRCC}$). Compared to Definition 3.1, the optimization objective becomes that, the summing of the number of the user-role assignments and that of the role-permission assignments should be minimum. This process can be formalized as follows:

$$\begin{cases} \min(|URA_{n\times k}| + |RPA_{k\times m}|) \\ URA_{n\times k} \otimes RPA_{k\times m} = UPA_{n\times m} \\ \sum\limits_j URA[i][j] \leq MRC_{user} \leq k,\ \forall i \in [1,n] \\ \sum\limits_j RPA[j][t] \leq MRC_{perm} \leq k,\ \forall t \in [1,m] \end{cases} \tag{9}$$

Definition 3.4.
(Min-noise role-engineering optimization problem, min-noise REOP$_{URCC\&PRCC}$). Compared to the above definitions, the optimization objective becomes that, the reconstruction error should be minimum when the number k of the mining roles is no more than k'. This process can be formalized as follows:

$$\begin{cases} \min ||URA_{n\times k} \otimes RPA_{k\times m} - UPA_{n\times m}||_1 \\ k \leq k' \\ \sum\limits_j URA[i][j] \leq MRC_{user} \leq k,\ \forall i \in [1,n] \\ \sum\limits_j RPA[j][t] \leq MRC_{perm} \leq k,\ \forall t \in [1,m] \end{cases} \tag{10}$$

3.2. **Complexity analysis.** Now we analyze the computational complexity of the above four optimization problems.

Statement 1. *The basic REOP$_{URCC\&PRCC}$, δ-approx REOP$_{URCC\&PRCC}$, edge-REOP$_{URCC\&PRCC}$, and min-noise REOP$_{URCC\&PRCC}$ are all NP-hard.*

The basic RMP is known to be *NP*-hard, as it can be converted into the set cover problem that is *NP*-hard. Notice that, the basic RMP can be viewed as a special case of the basic REOP$_{\text{URCC\&PRCC}}$ when both the constraint thresholds take values of the number of mining roles. Since the decision version of the basic RMP has been proven to be *NP*-complete [16], so is the basic REOP$_{\text{URCC\&PRCC}}$. Similarly, the basic REOP$_{\text{URCC\&PRCC}}$ can be viewed as a special case of the δ-approx REOP$_{\text{URCC\&PRCC}}$ when the reconstruction error takes a value of 0, and then the δ-approx REOP$_{\text{URCC\&PRCC}}$ is also *NP*-hard. Further, since the difference between the basic REOP$_{\text{URCC\&PRCC}}$ and edge-REOP$_{\text{URCC\&PRCC}}$ is only in terms of the optimization objective, the edge-REOP$_{\text{URCC\&PRCC}}$ is *NP*-hard. Last, the δ-approx REOP$_{\text{URCC\&PRCC}}$ can be viewed as a special case of the min-noise REOP$_{\text{URCC\&PRCC}}$ when the reconstruction error is less than δ, while minimizing the number of mining roles, then the min-noise REOP$_{\text{URCC\&PRCC}}$ is also *NP*-hard.

3.3. **Unified modelling framework.** The differences among all the variants lie in the constraint conditions and optimization objectives. The basic REOP$_{\text{URCC\&PRCC}}$ is the most basic and important variant, and the others can be easily extended once the basic REOP$_{\text{URCC\&PRCC}}$ is modelled. Here, for the convenience of discussion, Boolean matrices

$X_{n\times m}$, $C_{n\times k}$, and $R_{k\times m}$ are used to represent matrices $UPA_{n\times m}$, $URA_{n\times k}$, and $RPA_{k\times m}$, respectively, and the variants x_{it}, c_{ij} and r_{jt} are used to denote the cells of matrices $X_{n\times m}$, $C_{n\times k}$, and $R_{k\times m}$, respectively.

The ILP problem is an important branch of operational research, and it is widely used in economic analysis, operation management and engineering technology [16]. The 0-1 programming method in the ILP is often used to solve the practical problems such as the assignment, location, and delivery. According to Definitions 3.1-3.4, the unified modelling representations for all the RMP variants via 0-1 programming are presented as follows.

Step 1. Suppose there are q candidate roles: $r_1, r_2, \ldots, r_q$, and the set of all these roles can be represented as the matrix R. Next, define a new set $\{d_1, d_2, \ldots, d_q\}$ of identifier variables, in which role r_j is present when $d_j = 1$; otherwise, r_j is absent. Thus, the number of roles can be denoted as $k = \sum_{j=1}^{q} d_j$. Furthermore, several observations can be concluded as follows.

1) To ensure the integrity of the optimization process, if one user has a permission, which is represented as $x_{it} = 1$, then at least one role containing permission t should be assigned to user i, which can be denoted as $\sum_{j=1}^{q} c_{ij} \cdot r_{jt} \geq 1$, $\forall x_{it} = 1$; otherwise, any role containing permission t should not be assigned to user i, which can be denoted as $\sum_{j=1}^{q} c_{ij} \cdot r_{jt} = 0$, $\forall x_{it} = 0$.
2) At least one user has role r_j when $d_j = 1$, that is, at least one variable in $\{c_{1j}, c_{2j}, \ldots, c_{nj}\}$ has a value of 1, which can be denoted as $\forall i : d_j \geq c_{ij}$. Similarly, at least one variable in $\{r_{j1}, r_{j2}, \ldots, r_{jm}\}$ has a value of 1, which can be denoted as $\forall t : d_j \geq r_{jt}$.
3) The cardinality constraints URCC and PRCC should be taken into consideration, which can be denoted as $\sum_{j=1}^{q} c_{ij} \leq MRC_{user}$, and $\sum_{j=1}^{q} r_{jt} \leq MRC_{perm}$, respectively.

Based on these analyses, we model the basic $\text{REOP}_{\text{URCC\&PRCC}}$ in Equation (11) as follows, where $\min \sum_{j=1}^{q} d_j$ represents the optimization objective, and the other parts of the equation represent the constraint conditions.

$$\begin{aligned} &\min \sum_{j=1}^{q} d_j \\ &\text{s.t.} \begin{cases} \sum_{j=1}^{q} c_{ij} \cdot r_{jt} \geq 1, \forall x_{it} = 1 \\ \sum_{j=1}^{q} c_{ij} \cdot r_{jt} = 0, \forall x_{it} = 0 \\ \sum_{j=1}^{q} c_{ij} \leq MRC_{user} \\ \sum_{j=1}^{q} r_{jt} \leq MRC_{perm} \\ d_j \geq c_{ij},\ d_j \geq r_{jt},\ d_j \in \{0,1\},\ c_{ij} \in \{0,1\},\ r_{jt} \in \{0,1\} \\ 1 \leq j \leq q,\ 1 \leq i \leq n,\ 1 \leq t \leq m \end{cases} \end{aligned} \tag{11}$$

Step 2. Compared with the basic $\text{REOP}_{\text{URCC\&PRCC}}$, the optimization objective of the δ-approx $\text{REOP}_{\text{URCC\&PRCC}}$ remains unchanged, while the constraint conditions vary and tolerate a reconstruction error. Thus, we only need to update the constraints and follow the same target function as shown in Step 1. We model the δ-approx $\text{REOP}_{\text{URCC\&PRCC}}$ in Equation (12), where $\min \sum_{j=1}^{q} d_j$ represents the optimization objective, and the other parts of the equation represent the updated constraint conditions. The detailed descriptions of the constraints are as follows.

1) In the first two constraint conditions, an auxiliary variable x'_{it} is employed to relax the completeness constraint and make sure the reconstruction matrix is acceptable.

Without x'_{it}, the constraint is to enforce the exact permission coverage, as shown in Equation (11).

2) The third constraint indicates that the reconstruction error does not exceed a specific value δ.
3) The other constraints remain consistent with those in Equation (11).

$$\min \sum_{j=1}^{q} d_j$$
$$\text{s.t.} \begin{cases} \sum_{j=1}^{q} c_{ij} \cdot r_{jt} + x'_{it} \geq 1, \forall x_{it} = 1 \\ \sum_{j=1}^{q} c_{ij} \cdot r_{jt} - x'_{it} = 0, \forall x_{it} = 0 \\ \sum_{i=1}^{n} \sum_{t=1}^{m} x'_{it} \leq \delta \\ \sum_{j=1}^{q} c_{ij} \leq MRC_{user} \\ \sum_{j=1}^{q} r_{jt} \leq MRC_{perm} \\ d_j \geq c_{ij},\ d_j \geq r_{jt},\ d_j \in \{0,1\},\ c_{ij} \in \{0,1\},\ r_{jt} \in \{0,1\},\ x'_{it} \in \{0,1\} \\ 1 \leq j \leq q,\ 1 \leq i \leq n,\ 1 \leq t \leq m \end{cases} \tag{12}$$

Step 3. Compared with the basic $\text{REOP}_{\text{URCC\&PRCC}}$, the constraint conditions of the edge-$\text{REOP}_{\text{URCC\&PRCC}}$ remain unchanged, while the optimization objective varies. Thus, we only need to update the target function and follow the same constraints as shown in Step 1. To reduce the space cost of the modelling, users with the same permission set are clustered for a given set R of candidate roles. Assume that there are n' different permission sets, the numbers of users corresponding to these sets are $u_1, u_2, \ldots, u_{n'}$, respectively. Then, we model the edge-$\text{REOP}_{\text{URCC\&PRCC}}$ in Equation (13) as follows, where $\min\left(\sum_{i=1}^{n'}\left(u_i \cdot \sum_{j=1}^{q} c_{ij}\right) + \sum_{j=1}^{q}\left(d_j \cdot \sum_{t=1}^{m} r_{jt}\right)\right)$ represents the updated optimization objective, and the other parts of the equation represent the constraint conditions.

$$\min \left(\sum_{i=1}^{n'}\left(u_i \cdot \sum_{j=1}^{q} c_{ij}\right) + \sum_{j=1}^{q}\left(d_j \cdot \sum_{t=1}^{m} r_{jt}\right)\right)$$
$$\text{s.t.} \begin{cases} \sum_{j=1}^{q} c_{ij} \cdot r_{jt} \geq 1, \forall x_{it} = 1 \\ \sum_{j=1}^{q} c_{ij} \cdot r_{jt} = 0, \forall x_{it} = 0 \\ \sum_{j=1}^{q} c_{ij} \leq MRC_{user} \\ \sum_{j=1}^{q} r_{jt} \leq MRC_{perm} \\ d_j \geq c_{ij},\ d_j \geq r_{jt},\ d_j \in \{0,1\},\ c_{ij} \in \{0,1\},\ r_{jt} \in \{0,1\} \\ 1 \leq j \leq q,\ 1 \leq i \leq n',\ 1 \leq t \leq m \end{cases} \tag{13}$$

Step 4. Compared with the δ-approx $\text{REOP}_{\text{URCC\&PRCC}}$, both the optimization objective and constraint conditions vary for the min-noise $\text{REOP}_{\text{URCC\&PRCC}}$. To address these issues, we need to make some improvements: Remove the expressions with the slack variables from the constraint conditions, construct a new objective function using these slack variables, and add a new expression of the upper limit number of roles into the constraint conditions. We model the min-noise $\text{REOP}_{\text{URCC\&PRCC}}$ in Equation (14) as follows,

where $\min \sum_{i=1}^{n} \sum_{t=1}^{m} x'_{it}$ represents the optimization objective, and the other parts of the equation are the constraint conditions.

$$\min \sum_{i=1}^{n} \sum_{t=1}^{m} x'_{it}$$
$$\text{s.t.} \begin{cases} \sum_{j=1}^{q} d_j \leq k' \\ \sum_{j=1}^{q} c_{ij} \leq MRC_{user} \\ \sum_{j=1}^{q} r_{jt} \leq MRC_{perm} \\ d_j \geq c_{ij},\ d_j \geq r_{jt},\ d_j \in \{0,1\},\ c_{ij} \in \{0,1\},\ r_{jt} \in \{0,1\},\ x'_{it} \in \{0,1\} \\ 1 \leq j \leq q,\ 1 \leq i \leq n,\ 1 \leq t \leq m \end{cases} \tag{14}$$

3.4. **Heuristic algorithms.** First, to construct an initial RBAC system, the Fast Miner and BMD methods are used to mine various types of role sets according to the unconstrained RMP variants, as shown in Algorithm 1.

Algorithm 1. Initial construction of role engineering

Input: the original access control matrix $X_{n\times m}$

Output: the initial role set *Init_Roles*, and decomposed matrices $C_{n\times k}$ and $R_{k\times m}$

The Fast Miner and BMD methods are adopted to derive *Init_Roles* and configure an initial RBAC system, in order to satisfy different RMP variants, such as the basic RMP, δ-approx RMP, edge-RMP, and min-noise RMP.

Next, it is necessary to study how to implement both the URCC and PRCC in the optimization process. For this purpose, a role set *RU*, which would not cause any new violations to *RPA*, needs to be identified. Another role set *RI*, which would not cause any new violations to *URA*, also needs to be identified. The optimization process with the double cardinality constraints is presented in Algorithm 2.

In Algorithm 2, we first identify *RU*, *RI*, and determine the violating users or permissions using a heuristic strategy in lines 2-4. Then, if user u is chosen, the first l roles are chosen from *RU*; if permission p is chosen, the first l roles are chosen from *RI*. Similar to updating the *URA* and *RPA* using Algorithm 4 and Algorithm 5 in [25], the detailed specifications of the algorithm are omitted owing to the limited space.

4. **Experimental Analysis.** To validate the effectiveness and efficiency of the CREO _BMD&ILP, we conduct experiments using synthetic datasets and evaluate its performance in this section.

4.1. **Performance evaluations.** Taking the basic $\text{REOP}_{\text{URCC\&PRCC}}$, edge-$\text{REOP}_{\text{URCC\&PRCC}}$ and δ-approx $\text{REOP}_{\text{URCC\&PRCC}}$ as the research objects, we randomly construct synthetic matrices *UPA*, *URA* and *RPA* via Algorithm 1 with 80 number of users, 250 number of permissions, and 20 number of initial roles, such that matrix *UPA* is Boolean decomposed as matrices *URA* and *RPA* according to different optimized variants.

First, we carry out Algorithm 2 in order to study how the constraint URCC and other factors affect the optimized results of our method in the optimization phase, when the value of the cardinality constraint varies from 2 to 8 with a step of 1. We take account of the number of the constrained roles, and the size of the optimized assignment relationships *URA* and *RPA*, as evaluation measures. The experimental results are shown in Figures

Algorithm 2. Constrained role-engineering optimization

Input: the preprocessed results $C_{n\times k}$ and $R_{k\times m}$, candidate role set *Init_Roles*, and thresholds MRC_{user} and MRC_{perm}

Output: the optimized role set *Cons_Roles*, and updated matrices $C_{n\times k}$ and $R_{k\times m}$

1. Create and identify functions $count\_user\_roles(u)$, $count\_role\_users(r)$, $count\_perm\_roles(p)$, and $count\_role\_perms(r)$, respectively;
2. Identify RU, RI, such that
   $\forall r \in RU, \forall p \in role\_perms(r) : count\_perm\_roles(p) \leq MRC_{perm} - 1$;
   $\forall r \in RI, \forall u \in role\_users(r) : count\_user\_roles(u) \leq MRC_{user} - 1$;
3. **while** $((\exists u \in U : count\_user\_roles(u) > MRC_{user})$ or $(\exists p \in P : count\_perm\_roles(p) > MRC_{perm}))$ **do**
4. Choose violating users or violating permissions based on a heuristic strategy;
5. **if** user u is chosen **then**
6. $l = count\_user\_roles(u) - (MRC_{user} - 1)$;
7. Choose the first l roles of u from RU with the greatest values of $count\_role\_users(r)$ to constitute set S;
8. Merge the permissions of all the l roles and denote the union as set P_S;
9. Create a new role r_{nr} such that $role\_perms(r_{nr}) = P_S$;
10. Update the RPA and URA via Algorithm 4 in [25];
11. **else**
12. $l = count\_perm\_roles(p) - (MRC_{perm} - 1)$;
13. Choose the first l roles of p from RI with the greatest values of $count\_role\_perms(r)$ to constitute set S;
14. Intersect the permissions of all the l roles and denote the intersection as set P_S;
15. Create a new role r_{nr} such that $role\_perms(r_{nr}) = P_S$;
16. Update the URA and RPA with r_{nr} via Algorithm 5 in [25];
17. **end if**
18. **end while**

2-4. In Figure 2, the lateral axis represents the constraint threshold MRC_{user}, and the vertical axis represents the number of roles. In Figure 3, the lateral axis represents MRC_{user}, and the vertical axis represents the size of the assignments relationships URA and RPA. In Figure 4, the lateral axis represents the approximation rate, simply denoted as δ, which is calculated by $|URA \otimes RPA|/|UPA|$, and the vertical axis represents the number of roles.

Figure 2 demonstrates that the number of roles tends to decrease slightly as the value of MRC_{user} increases. When the value of MRC_{user} increases to a certain value, the number of roles tends to be stable and becomes unchanged. Specifically, the number of roles does not obviously vary and remains close to 15 when the value of MRC_{user} exceeds 6. However, the number of roles increases as MRC_{user} decreases. When the value of MRC_{user} equals 2, the number of roles becomes 23. Figure 3 demonstrates the effects of the constraint MRC_{user} on the size of the relationships between URA and RPA, which is also regarded as the administrative cost of the system. Obviously, it tends to decrease as the value of MRC_{user} increases, which is similar to the variation tendency of Figure 2. This is because the greater the value of MRC_{user} is, the weaker the constraint will be. More general roles are assigned to any user, in other words, when MRC_{user} takes a greater value, general roles that do not include many permissions are more practical and can be utilized frequently. Thus, fewer irregular roles need to be generated, and the number of roles and size of assignments do not vary considerably. On the contrary, the

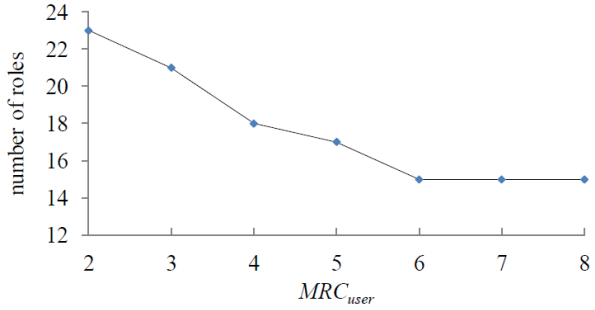


FIGURE 2. Optimized results for the basic $\text{REOP}_{\text{URCC\&PRCC}}$

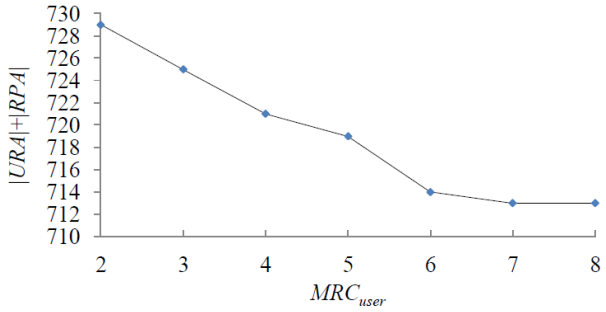


FIGURE 3. Optimized results for the edge-$\text{REOP}_{\text{URCC\&PRCC}}$

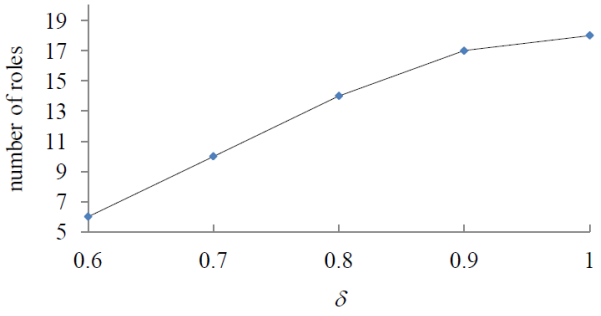


FIGURE 4. Optimized results for the δ-approx $\text{REOP}_{\text{URCC\&PRCC}}$

less the value of MRC_{user}, the stricter the constraint. When the constraint becomes strict and takes a small value, each user is allowed to associate with few roles. Thus, these roles need to possess more access permissions, in order to ensure that each user can still get enough access permissions. As a result, the number of roles and size of the assignment relationships increase remarkably. For the same dataset, in order to evaluate the effects of the approximation ratio on the mining roles, the value of MRC_{user} is fixed at 4, and the value of δ varies from 0.6 to 1. From Figure 4 it is observed that, the number of roles increases as δ increases. This is because the greater value of the approximation rate, the better the integrity of the matrix reconstruction. Thus, more roles are needed for covering the original permission assignments.

Similarly, we evaluate the effects of the constraint PRCC and other factors on the optimized results, with respect to the same evaluation measures. The experimental results are shown in Figures 5-7. Figure 5 shows that, when $MRC_{perm} \geq 52$, the number of roles first decreases slightly and then grows to be stable as the value of MRC_{perm} increases. Specifically, when MRC_{perm} equals 1, 231 roles are developed, and each role includes

only one permission, which is not practical; when $2 \leq MRC_{perm} < 52$, the number of roles decreases remarkably from 231 to 35 with the increasing value of MRC_{perm}; when $52 \leq MRC_{perm} < 112$, the number of roles decreases slightly from 35 to 18; when $MRC_{perm} \geq 112$, it grows to be stable. Figure 6 demonstrates the effects of the constraint MRC_{perm} on the size of assignments URA and RPA, which tends to decrease as the value of MRC_{perm} increases and is similar to the variation tendency of Figure 5. This is because, the less the value of MRC_{perm} is, the stricter the constraints and the more roles and role-permission assignments that satisfy the constraint requirements will be. As a result, the value of $(|URA| + |RPA|)$ increases accordingly as MRC_{perm} decreases. For the same dataset, to evaluate the effects of the approximation ratio on the mining roles, the value of MRC_{perm} is fixed at 80, and the value of δ varies from 0.6 to 1. From Figure 7 it is observed that, the number of roles increases as δ increases. This is attributed to the fact that, more roles are needed, in order to ensure the better integrity of the matrix reconstruction.

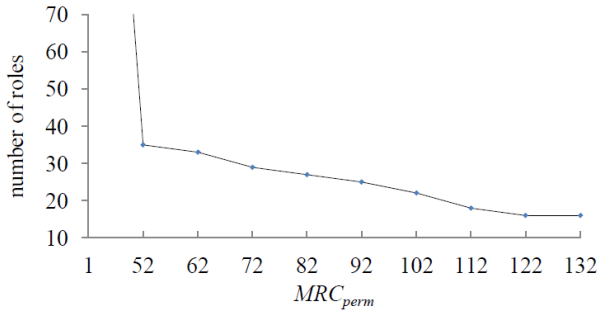


FIGURE 5. Optimized results for the basic $\text{REOP}_{\text{URCC\&PRCC}}$

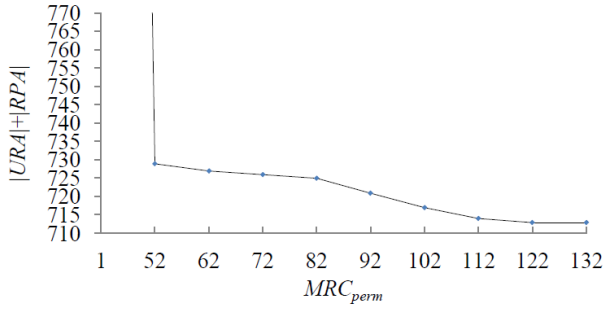


FIGURE 6. Optimized results for the edge-$\text{REOP}_{\text{URCC\&PRCC}}$

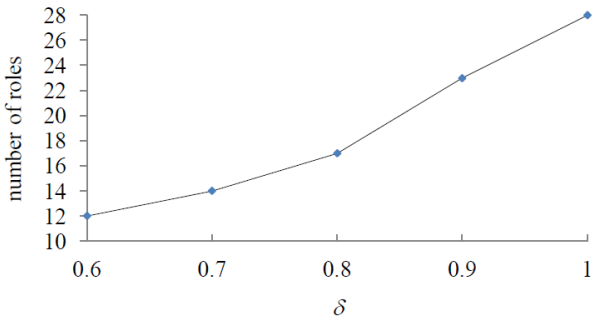


FIGURE 7. Optimized results for the δ-approx $\text{REOP}_{\text{URCC\&PRCC}}$

4.2. **Performance comparisons.** Next, we study the role selection strategy as shown in Line 4 of Algorithm 2. Taking the basic $\text{REOP}_{\text{URCC\&PRCC}}$ as the evaluation object, we randomly construct a synthetic matrix *UPA* consisting of 50 number of users and 50 number of permissions. We adopt the parameter settings as follows: The value of MRC_{user} is fixed at 3; the number of initial roles varies from 2 to 10 with a step of 2, denoted as q; the density value of matrix *UPA* varies from 0.05 to 0.25 with a step of 0.05, denoted as ρ. Furthermore, we utilize four different types of role selection strategies: 1) Greedy, which identifies the role that can cover the most remaining permission assignments, has been widely used for solving different role engineering problems on large-scale datasets; 2) fewest, which identifies the role that contains the fewest permissions; 3) most, which identifies a role that contains the most permissions; 4) random, which optionally chooses a role from the initial set.

To study the effects of different combinations of parameters on the mining results, we generate 5 synthetic matrices of permission assignments and take consideration of the number of roles and execution time as measures. The results are shown in Figures 8-11. In Figures 8 and 9, the lateral axes represent the varying values of matrix density, and the vertical axes respectively represent the number of roles and execution time. In Figures 10 and 11, the lateral axes represent the varying number of initial roles, and the vertical axes respectively represent the number of optimal roles and execution time. From the viewpoint of number of roles, the fewest and greedy strategies have the comparable performance and perform better than the other two. From the viewpoint of execution time, the random performs the best, and the greedy performs the worst. This is because the greedy method needs to consider and check all of the candidate roles at any iteration, in order to discover the best one, which costs much computational time. However, the fewest and most methods only need to consider the candidate role that includes either the minimum or the maximum number of permissions. The random method runs fast and costs the least execution time, as any candidate role may be selected at any iteration. It is also observed that, in terms of the number of roles and execution time, the fewest always outperforms the greedy with a fewer values of parameter settings, though the greedy method is a classic searching strategy that attempts to find a global optimal solution for a specific optimization problem. For the other variants such as the δ-approx $\text{REOP}_{\text{URCC\&PRCC}}$, edge-$\text{REOP}_{\text{URCC\&PRCC}}$ and min-noise $\text{REOP}_{\text{URCC\&PRCC}}$, the fewest has the similar performance to that of the basic $\text{REOP}_{\text{URCC\&PRCC}}$. Therefore, the fewest is more applicable than the greedy method for solving the different role-optimization problems on small-scale datasets.

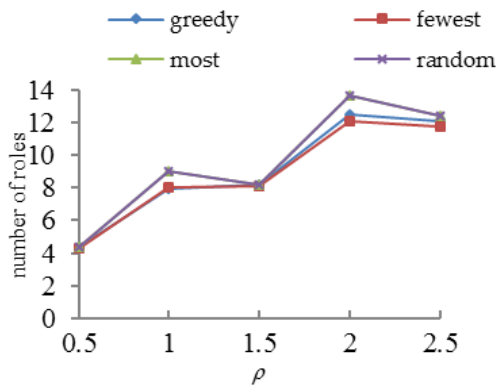


FIGURE 8. Number of roles using different strategies

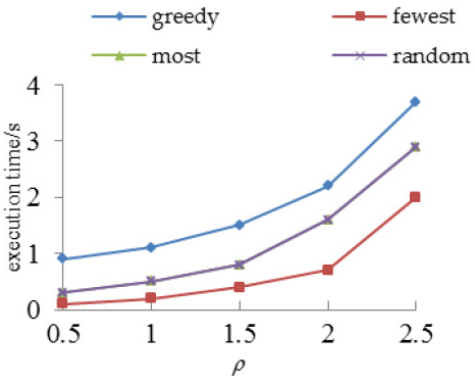


FIGURE 9. Execution time using different strategies

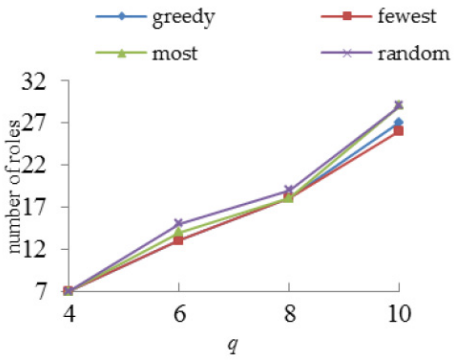


FIGURE 10. Number of roles using different strategies

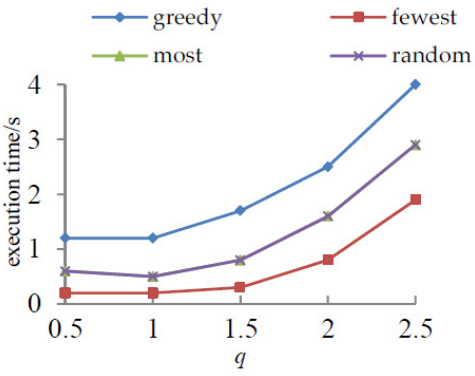


FIGURE 11. Execution time using different strategies

4.3. **Advantages of the CREO_BMD&ILP.** From the above experimental analysis, we find the CREO_BMD&ILP has the following main advantages.

1) It can flexibly satisfy different optimization objectives for organizational requirements in a unified modelling framework by using the BMD and ILP techniques.
2) It can verify whether the cardinality constraints URCC and PRCC can be simultaneously satisfied in the constructed model by using the proposed heuristic algorithms.

Compared with the existing studies, the characteristics of our method are shown in Table 1, where a tick $\surd$ indicates that the characteristic is available.

TABLE 1. Comparison of characteristics

Characteristic	Ma et al. [18]	Harika et al. [21]	Blundo et al. [22]	Blundo et al. [23]	Sarana et al. [24]	Sun et al. [25]	Our method
URCC	√	√	√			√	√
PRCC		√	√	√		√	√
Other constraints	√			√	√	√	
Flexibility and extendibility							√

5. **Conclusions.** A novel role-engineering method, called CREO_BMD&ILP, was proposed in this paper. We first converted the role-engineering problem into the BMD problem, defined several variants of the optimization problem with multiple cardinality constraints, and proposed a unified modelling framework using the ILP technique. Then, we presented the heuristic algorithms for solving all the optimization problems in the constructed system model. The experiments on synthetic datasets demonstrated that the proposed method is efficient and effective. However, a few interesting issues remain to be resolved. To further enhance the interpretability of mining results, one issue is how to implement the other cardinality constraints or other types of constraints for role-engineering optimization in future work.

Acknowledgment. This work is partially supported by the National Natural Science Foundation of China (61501393), the Natural Science Foundation of Henan Province of China (182300410145, 182102210132), and the Foundation of Henan Educational Committee under Contract No. 20B520031.

REFERENCES

[1] W. Sun, H. Su and H. Xie, Policy-engineering optimization with visual representation and separation-of-duty constraints in attribute-based access control, *Future Internet*, vol.12, no.10, p.164, 2020.

[2] G. Batra, V. Atluri, J. Vaidya and S. Sural, Deploying ABAC policies using RBAC systems, *Journal of Computer Security*, vol.27, no.4, pp.483-506, 2019.

[3] M. Ghafoorian, D. Abbasinezhad-Mood and H. Shakeri, A thorough trust and reputation based RBAC model for secure data storage in the cloud, *IEEE Transactions on Parallel and Distributed Systems*, vol.30, no.4, pp.778-788, 2018.

[4] J. P. Cruz, Y. Kaji and N. Yanai, RBAC-SC: Role-based access control using smart contract, *IEEE Access*, no.6, pp.12240-12251, 2018.

[5] W. Sun and H. Su, Role-engineering optimization with mutually exclusive permissions constraints and permission-to-role cardinality constraints, *International Journal of Innovative Computing, Information and Control*, vol.17, no.4, pp.1373-1390, 2021.

[6] N. Pan, Z. Zhu, L. He and L. Sun, An efficiency approach for RBAC reconfiguration with minimal roles and perturbation, *Concurrency and Computation: Practice and Experience*, vol.30, no.11, p.e4399, 2018.

[7] G. Seannery, Yacob, N. Chandra and D. David, Optimization of hospital patient management in hospitals with Android-based applications, *ICIC Express Letters*, vol.14, no.3, pp.211-217, 2020.

[8] M. Narouei and H. Takabi, Towards an automatic top-down role engineering approach using natural language processing techniques, *Proc. of the 20th ACM Symposium on Access Control Models and Technologies*, pp.157-160, 2015.

[9] B. Mitra, S. Sural, J. Vaidya and V. Atluri, A survey of role mining, *ACM Computing Surveys*, vol.30, no.11, pp.1-37, 2016.

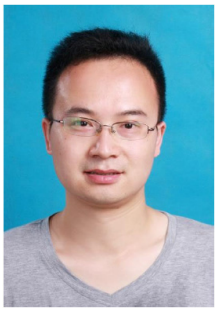
[10] S. Schefer-Wenzl and M. Strembeck, Modeling support for role-based delegation in process-aware information systems, *Business & Information Systems Engineering*, vol.6, no.4, pp.215-237, 2014.

[11] W. Bai, Z. Pan, S. Guo and Z. Chen, RMMDI: A novel framework for role mining based on the multi-domain information, *Security and Communication Network*, 2019.

[12] S. D. Stoller and T. Bui, Mining hierarchical temporal roles with multiple metrics, *Journal of Computer Security*, vol.26, no.1, pp.121-142, 2018.

[13] N. Gal-Oz, Y. Gonen and E. Gudes, Mining meaningful and rare roles from web application usage patterns, *Computers & Security*, vol.82, pp.296-313, 2019.
[14] B. Mitra, S. Sural, J. Vaidya and V. Atluri, Mining temporal roles using many-valued concepts, *Computers & Security*, vol.60, pp.79-94, 2016.
[15] J. Vaidya, V. Atluri and Q. Guo, The role mining problem: Finding a minimal descriptive set of roles, *Proc. of the 12th ACM Symposium on Access Control Models and Technologies*, pp.175-184, 2007.
[16] H. Lu, J. Vaidya and V. Atluri, An optimization framework for role mining, *Journal of Computer Security*, vol.22, no.1, pp.1-31, 2014.
[17] F. Nazerian, H. Motameni and H. Nematzadeh, Emergency role-based access control (E-RBAC) and analysis of model specifications with alloy, *Journal of Information Security and Applications*, vol.45, pp.131-142, 2019.
[18] X. Ma, R. Li, H. Wang and H. Li, Role mining based on permission cardinality constraint and user cardinality constraint, *Security and Communication Networks*, vol.8, no.13, pp.2317-2328, 2015.
[19] A. Colantonio, R. D. Pietro, A. Ocello and N. V. Verde, Visual role mining: A picture is worth a thousand roles, *IEEE Transactions on Knowledge and Data Engineering*, vol.24, no.6, pp.1120-1133, 2012.
[20] N. V. Verde, J. Vaidya, V. Atluri and A. Colantonio, Role engineering: From theory to practice, *Proc. of the 2nd ACM Conference on Data and Application Security and Privacy*, pp.181-192, 2012.
[21] P. Harika, M. Nagajyothi, J. C. John, S. Sural, J. Vaidya and V. Atluri, Meeting cardinality constraints in role mining, *IEEE Transactions on Dependable and Secure Computing*, vol.12, no.1, pp.71-84, 2014.
[22] C. Blundo, S. Cimato and L. Siniscalchi, Managing constraints in role based access control, *IEEE Access*, no.8, pp.140497-140511, 2020.
[23] C. Blundo, S. Cimato and L. Siniscalchi, Role mining heuristics for permission-role-usage cardinality constraints, *The Computer Journal*, no.2, pp.1-26, 2021.
[24] P. Sarana, A. Roy, S. Sural, J. Vaidya and V. Atluri, Role mining in the presence of separation of duty constraints, *International Conference on Information Systems Security*, pp.98-117, 2015.
[25] W. Sun, H. Su and H. Liu, Role-engineering optimization with cardinality constraints and user-oriented mutually exclusive constraints, *Information*, vol.10, no.11, p.342, 2019.

Author Biography



Wei Sun received his B.S. and M.S. degrees from the School of Information Engineering, Zhengzhou University, China, in 2003 and 2008, respectively. He is currently working in the School of Computer and Information Technology, Xinyang Normal University. His current research interests include access control and system security.