Journal of Computer Security 00 (20xx) 1–24 1
DOI 10.3233/JCS-140519
IOS Press

Towards user-oriented RBAC model

Haibing Lu [a,*], Yuan Hong [b], Yanjiang Yang [c], Lian Duan [d] and Nazia Badar [e]
[a] *Santa Clara University*
[b] *University at Albany – SUNY*
[c] *I2R Singapore*
[d] *New Jersey Institute of Technology*
[e] *Rutgers University*

Abstract. Role mining is to define a role set to implement the role-based access control (RBAC) system and regarded as one of the most important and costliest implementation phases. While various role mining models have been proposed, we find that user experience/perception – one ultimate goal for any information system – is surprisingly ignored by the existing works. One advantage of RBAC is to support multiple role assignments and allow a user to activate the necessary role to perform the tasks at each session. However, frequent role activating and deactivating can be a tendinous thing from the user perspective. A user-friendly RBAC system is expected to assign few roles to every user. So in this paper we propose to incorporate to the role mining process a user-role assignment constraint that mandates the maximum number of roles each user can have. Under this rationale, we formulate user-oriented role mining as the user role mining problem, where all users have the same maximal role assignments, the personalized role mining problem, where users can have different maximal role assignments, and the approximate versions of the two problems, which tolerate a certain amount of deviation from the complete reconstruction. The extra constraint on the maximal role assignments poses a great challenge to role mining, which in general is already a hard problem. We examine some typical existing role mining methods to see their applicability to our problems. In light of their insufficiency, we present a new algorithm, which is based on a novel dynamic candidate role generation strategy, tailored to our problems. Experiments on benchmark data sets demonstrate the effectiveness of our proposed algorithm.

Keywords: **???**

1. Introduction

As opposed to the permission-based access control mechanism, which assigns permissions required to perform tasks to users directly, the role-based access control (RBAC) mechanism defines a role set by associating permissions to roles and then assigns roles, rather than a number of permissions, to users. RBAC has been regarded as a de facto access control model. Its many advantages include the convenience of authorization allocation and the reduction of the system administrative workload. To benefit from those advantages, enterprises still employing their old access control systems need to migrate to RBAC. To accomplish the migration, the first phase is to define a good role set. While the role defining problem is seemingly straightforward, it has been recognized as one of the costliest phases in the implementation of RBAC and poses a great challenge to the system engineers. The difficulty comes from the fact that a RBAC system engineer usually has little knowledge on the semantic meanings of user responsibilities and business processes within an enterprise.

Role mining has proven to be an effective (machine-operated) means of discovering a good role set. Its key idea is to utilize data mining technologies to extract patterns from existing permission assignments

* Corresponding author. E-mail: hlu@scu.edu.

of the old access control system, which are then used to establish roles. This greatly facilitates the implementation of RBAC (by migrating from the old access control system). In the literature, role mining has been extensively studied. In a nutshell, the existing literature investigates role mining with different objectives, including minimization of the number of roles, minimization of the administration cost, minimization of the complexity of the role hierarchy structure, and others. However, we find that few existing works considered to improve end-user experience (of the underlying RBAC system), which should be one important goal for any practical information system. Needless to say, users' experience/perception of a system represents system usability and directly affects the eventual success of the system in practice. As such, we argue that user-friendliness should be an essential criterion for evaluating the quality of role mining.

In this paper, we study user-oriented role mining, being the first to explore the role mining problem from the end-user perspective. RBAC allows a user to assume multiple roles. A user can switch roles at different sessions to activate necessary permissions. However, frequent role switching can be a tedious thing for users. Our daily experiences tell us that end users often prefer fewer role assignments; as long as a user acquires all the needed permissions, the fewer roles she has to bear, the better usability she may feel upon the system. That is, from the end-user perspective, a good RBAC system should have as few user-role assignments as possible. This in fact coincides with an advantage of RBAC: recall that one reason accounting for the wide acceptance of RBAC is that it allows users to carry very few roles while enjoying their (potentially many) access rights. However, on the flip side, if we create a unique role for every user, in which case user-role assignments are trivially the most sparse, then the resultant RBAC system would contain too many roles. This would contradict to a premise of RBAC, which is to map permission roles with functional roles within an organization. As such, a user-oriented RBAC solution should not compromise other advantages of RBAC. To this end, we propose to limit the maximum number of roles a user can take on top of regular role mining. Such a strategy would well balance user friendliness and other system quality factors such as administrative overhead. While the idea is clear, the added constraint poses extra challenges to role mining, considering that role mining in general has already been a hard problem. Towards tackling the obstacle, we make the following contributions: (1) we formulate user-oriented role mining as four specific problems that are the user RMP, the approximate user RMP, the personalized RMP, and the approximate personalized RMP; (2) we study the theoretical properties of the four problems and formulate them as standard optimization problems; (3) in view of the weaknesses of the existing role mining algorithms, we present efficient algorithms, tailored to the user-oriented role mining problems; (4) to investigate the effectiveness of our algorithm, we conduct experiments on benchmark data sets and obtain promising experimental results.

This paper is an extension of a paper with the same title accepted in The Proceeding of the 27th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy [13]. The extended version significantly extends the original conference paper. In particular, it addresses two new problems: the personalized RMP and the approximate personalized RMP. The properties of these two problems are presented. Variants of original algorithms are also expanded. Four different methods for candidate role generation are investigated and compared with respect to performances. Several new experiments are conducted on new examples of configuration.

The remainder of the paper is organized as follows. Section 2 reviews existing role mining works in the literature. Section 3 presents the user-oriented role mining problem. Section 4 presents optimization models. The heuristic algorithm is provided in Section 5. Experimental results on benchmark access control data sets are reported in Section 6. Section 7 concludes the paper.

2. Related work

The concept of role engineering was introduced in 1995 by Coyne [2]. It aims to define an architectural structure that is complete, correct and efficient to specify the organization's security policies and business functions. Coyne's approach is a top-down process oriented strategy for role definition. With the top-down approach, one generally starts from requirements and successively refines the definitions to reflect the business functions. Fernandez and Hawkins [5] used cases to determine the needed permissions of roles. Brooks [1] developed a graphical user interface to migrating to a role-based environment. Roeckle et al. [25] proposed to deduce roles by analyzing business processes. Shin et al. [27] conducted a top-down role engineering by examining backward and forward information flows and employing unified modeling language. Neumann and Strembeck [23] discovered roles by aggregating permissions derived from analyzing usage scenarios. Kern et al. [11] proposed an iterative-incremental approach, that considerers different stages of the role life-cycle including analysis, design, management and maintenance.

Top-down approaches are not practical for enterprises of medium or large size, because it is difficult for an access control engineer to understand the semantic meanings of a large number of operational processes and user duties in a short time. For large-scale problems, bottom-up approaches, that discover roles in an automated fashion, are preferred. Bottom-up approaches typically utilize existing user-permission assignments to formulate roles. In particular, data mining techniques are often employed in bottom-up approaches to identify promising roles. Kuhlmann et al. [12] coined the concept of role mining using data mining. In [26], an algorithm ORCA was proposed to build a hierarchy of permission clusters using data mining technologies. However, overlap between roles is not allowed in ORCA, which contradicts to normal practice in real applications. Vaidya et al. [32] proposed a subset enumeration approach, which can effectively overcome this limitation. Lu et al. [14,15] formulated the role mining problem as binary linear integer program, that enables role engineers to utilize optimization software packages to discover roles. Ene et al. [4] provided some fast exact and heuristic algorithms for discovering the minimum role set.

An inherent issue with various bottom-up role discovering approaches is that there is no unanimous notion for goodness of a role set. Vaidya et al. [29] proposed to use the number of roles to evaluate the goodness of a role set. They [30] also introduced to use the administrative task as an evaluative criterion. Role hierarchy is another important evaluative criterion, as it is closely related to the semantic meanings of roles. Related works on role hierarchy include [3,9,20]. Molloy et al. [21] further introduced a weighted structural measure, which could describe most of studied evaluative criteria. Ma et al. [19] also used weights to combine multiple objectives. Our work in this paper strengthens this line of research by incorporating user experience/perception as an extra evaluative criterion in role mining.

Beside the above works on finding a role set with respect to different criteria, there are other interesting works with different flavors. Lu et al. [14] presented an optimization framework for role engineering. They even extended their work to incorporate negative authorizations in [17,18] so that the discovered RBAC solutions support either negative role assignments or negative permissions in roles. The role mining problem can be related to many problems outside the access control field, such as the discrete basis problem [24], the database tiling problem [8], the overlapping clustering problem [28], among others. So the approaches designed for those problems could be utilized for the role mining purpose. For example, Frank et al. [6] provided a probabilistic model to analyze the relevance of different kinds of business information for defining roles that can both explain the given user-permission assignments and describe the meanings from the business perspective. They [7] also introduced a model to take the attributes of users into account. Some research works were concerned about data cleanliness, as collected

user-permission assignments data may be contaminated. Studies on role mining in the presence of noisy data are presented in [22,31].

3. Problem definitions

In this section, we study and formulate user-oriented role mining. As we have discussed in the Introduction, from users' perspective a user-friendly RBAC system should assign as few roles as possible to each user; no user is happy with being overwhelmed by assuming too many roles (titles). Ideally, a user would wish to carry only one role given that the role provides with him all the necessary access privileges for him to work and function smoothly. Indeed, in reality most organization's systems are designed that way. For example, in a school system, the majority of people carry only one role among STUDENT, FACULTY, STAFF and VISITOR. In a software company, most employees are either ACCOUNTANT, ENGINEER or MANAGER. Thus, user-oriented role mining is characterized with the fact that the maximal role assignment for each user should be constrained. This gives rise to the definition of user-oriented role mining, as stated below.

Definition 1 (User-Oriented Role Mining). User-Oriented Role Mining is to discover a RBAC solution with respect to some evaluation criterion, such that after being assigned to roles, every user gets the same permissions as before, and the user-role assignments satisfy the predefined policy on the maximal role assignments.

3.1. Existing evaluation criteria

To concretize user-oriented role mining, we have one question to answer: what evaluation criterion should be used to evaluate the goodness of a RBAC solution? In the literature, role mining is typically formulated as certain optimization problems with objectives and constraints. As summarized by Molloy et al. at [20], there are five main factors which can be used to evaluate the goodness of a RBAC solution. They are the number of roles $|R|$, the number of user-role assignments $|UA|$, the number of role-permission assignments $|PA|$, the number of direct user-permission assignments $|DUPA|$, and the number of edges in the reduced role hierarchy $|t\_reduce(RH)|$. Among them, $|R|$, $|UA|$ and $|PA|$ are routine notations in RBAC, and no further exposition is needed on them. Direct user-permission assignments *DUPA* imply that roles of one single user are treated as special roles. $|DUPA|$ is the amount of such direct user-permission assignments in a deployed RBAC solution. Role hierarchy $RH \subseteq R \times R$ represents the partial order over roles R. $t\_reduce(RH)$ denotes the transitive reduction of the role hierarchy.

Almost all role mining evaluative criteria can be generally described with the weighted structural complexity measure introduced in [20], which sums up the above five factors, with possibly different weights for each factor.

Definition 2 (Weighted structural complexity measure). Given a weight vector $W = \langle w_r, w_u, w_p, w_d, w_h \rangle$, where $w_r, w_u, w_p, w_d, w_h \in \mathcal{Q}^+ \cup \{0\}$,[1] the weighted structural complexity of an RBAC state γ, which is denoted as $\text{wsc}(\gamma, W)$ is computed as:

$$\text{wsc}(\gamma, W) = w_r * |R| + w_u * |UA| + w_p * |PA| + w_d * |DUPA| + w_h * \left|t\_reduce(RH)\right|.$$

[1] $\mathcal{Q}^+$ is the set of all non-negative rational numbers.

Role mining in general involves minimizing wsc(γ, W). However, a minimization implicating all factors not only is too complex, but may not lead to a good RBAC system, as they may counteract with each other in the minimization. Depending on the objective to achieve, a specific role mining task often chooses to minimize a subset of factors relevant to the underlying objective. In particular, minimizing the number of roles $|R|$ might be the most studied role mining problem, where in the weighted structural complexity measure, w_r is a positive number while others are all 0. Such a specific role mining problem is referred to as basic RMP (Role Mining Problem), and it has been proven equivalent to the classic set cover problem.

The sum of user-role assignments and role-permission assignments of $|UA|+|PA|$ is commonly viewed as the representation of the system administrative cost. The minimization of $|UA| + |PA|$ is called edge RMP [30]. However, as far as an end-user is concerned, $|UA|$ is the only part that she can experience of a RBAC system. A user would not care how complex *PA* is. For example, a student would not care about how many permissions her STUDENT role actually contains, and all she cares is the simplicity of executing her role (or roles).

Other evaluative criteria, such as minimizing the complexity of the resultant role hierarchy [20], are also more from the system administrator perspective, rather than the end-user perspective.

By examining existing evaluative criterion, we found that the only factor in the Weighted Structural Complexity Measure that matters to end-users is the size of user-role assignments $|UA|$ and the number of roles $|R|$. If the user-oriented RMP is defined as the minimization of $|UA|$, the trivially optimized solution is to create a unique role for every user. That absolutely contradicts to the premise of role mining, which is to map permission roles with functional roles. As such, a user-oriented RMP solution should balance the user-friendliness and the overall quality of the system. Among five factors in the weighted structural complexity measure, the number of roles $|R|$ would be the best representative of the succinctness and goodness of a RBAC solution and is also the most studied criterion. So we propose to define the user-oriented RMP as the minimization of the number of roles $|R|$.

3.2. Definitions

Given the evaluation criteria, we study four variants of the user-oriented role mining. They are user RMP, approximate RMP, personalized RMP, and approximate personalized RMP. The notations to be used in their definitions are provided in Table 1. Note that we represent all data sets, i.e. user-permission assignments, user-role assignments, and role-permission assignments, in the binary matrix form. It helps to build a connection with the existing work of Boolean matrix decomposition. In the following, we will define the four user-oriented role mining variants.

RMP commonly refers to discovering roles from existing user-permission assignments without much consideration of the semantics of operations within an institution/organization. One basic requirement of role mining is that the discovered roles should well reconstruct the original user-permission assignments.

Table 1
Notations

UPA	User-permission assignment matrix
$UPA_{i.}$	The ith row of *UPA* (permissions assigned to user i)
UA	User-role assignment matrix
$UA_{i.}$	The ith row of *UA* (roles assigned to user i)
PA	Role-permission assignment matrix
$PA_{i.}$	The ith row of *PA* (the permissions associated with role i)

The strictest case is completeness; in other words, for every user there exists an assignment of roles such that the user gets the same permissions (no more no less) as being assigned before. As user-permission assignments, user-role assignments, role-permission assignments are represented by binary matrices *UPA*, *UA* and *PA*, respectively, the completeness requirement can be described as $UPA = UA \otimes PA$, where $\otimes$ denotes the Boolean matrix product. It mandates that (i) if a user is assigned to a permission originally, then the user has to be assigned to at least one role that contains that permission; (ii) if a user is not assigned to a particular permission originally, that person cannot be assigned to any role that contains that permission. Another expectation of RMP is the discovered role should be considered good in some sense. There is no unanimous criterion in the literature on the goodness of a role set. The commonly used criterion is the number of derived roles. The main advantage of RBAC over other access control schemes is simplicity of administration that is reflected by the fact that the number of required roles is usually much less than the number of permissions. So it naturally leads to that the less the number of roles, the less administrative cost would be incurred. The RMP with the completeness constraint and the objective of minimizing the number of roles is called the basic RMP.

However, the basic RMP originates purely from the administrative angle and does not reflect any user perspective. An optimal solution for a basic RMP instance may lead to a system with the least administrative cost. But it does not necessarily mean that the system would be favored by the users. For example, a RBAC solution with 10 roles and each user gets up to 2 roles may be favored by users over a RBAC solution with 8 roles and each user gets at least 6 roles. From the user perspective, a good RBAC system is a system with less role assignments to users, so users do not have to frequently switch roles between sessions. For example, a doctoral student who works as teaching assistant at a university would wish to get only one role that is designed for student teaching assistant rather than two roles, one for student and the other for faculty. That is why we propose the user RMP. The user RMP essentially is extended from the basic RMP. The only difference is the uniform constraint on the maximal role assignments for all users. The user RMP can be used to model the role mining case that the system engineer has prior knowledge on the maximal role assignments. The knowledge would help filter out many unwanted RBAC solutions and improve the quality of the mined role set. Note that the user RMP assumes all users have the same tolerance of excess role assignments, which might not be true in some cases. We will address the issue later by proposing other RMP variants. The formal definition of user RMP is given as below.

Problem 1 (User RMP). Given m users, n permissions, user-permission assignments $UPA_{m\times n}$ and an positive integer t, it is to discover user-role assignments $UA_{m\times k}$ and a role set $PA_{k\times n}$ such that: (1) the number of roles k is minimized, (2) the role assignments *UA* and the permission-role assignments *PA* accurately and completely reconstruct the user-permission assignments *UPA*, and (3) no user has more than t role assignments.

It can be roughly formulated as below:

$$\min k \quad s.t. \begin{cases} UPA_{m\times n} = UA_{m\times k} \otimes PA_{k\times n}, \\ \sum_j UA_{ij} \leqslant t \quad \forall i, \end{cases} \tag{1}$$

where $\otimes$ is the Boolean product operator [14].

The completeness constraint for the basic RMP is not necessary in some cases. Case 1: a bottom-up role mining approach is used to explore potentially promising roles, rather than finalization. The advantage of bottom-up role mining is being able to discover roles in an automated fashion by utilizing a computer. However, it typically ignores the semantics within an institution/organization due to the difficulty of processing semantic information. As a result, it happens that the derived roles do not meet practical needs, e.g. permissions grouped in a role do not have any semantic connection. In this case, it is not necessary to enforce the completeness constraints, as the enforcement might miss some potentially interesting roles. Case 2: the existing user-permission assignments contain noise, e.g. a personnel change occurs and is not timely reported after the data is collected. For that case, enforcing the completeness constraints would cause the over-fitting problem. Relaxing the completeness constraint could alleviate the problem. The approximate RMP is the RMP with the objective of minimizing the number of roles and the constraint that the reconstructed user-permission assignments approximate the original assignments to a certain degree. Same as the basic RMP, the approximate RMP does not consider the user perspective. Users would always prefer a RBAC solution with few role assignments, regardless of the overall system administrative cost. By extending the approximate RMP, we propose and study the approximate user RMP, which is the same as the approximate RMP, except that a user cannot be assigned to more than a certain number of roles. The number could come from prior knowledge or experience. Again here we assume that all users have the tolerance of excess role assignments.

Problem 2 (Approximate user RMP). The approximate user RMP is the same as the user RMP, except for that *UPA* can be inaccurately reconstructed with the number of errors not greater than a predefined value δ.

It can be described as the following optimization problem.

$$\min k$$
$$s.t. \begin{cases} \sum_{ij} (UPA_{m\times n} - UA_{m\times k} \otimes PA_{k\times n}) \leqslant \delta, \\ \sum_{j} UA_{ij} \leqslant t \quad \forall i. \end{cases} \tag{2}$$

Both the user RMP and approximate user RMP assume that all users have the same tolerance of excess role assignments and exert the same constraint of the maximal role assignment for all users without catering for users' concrete needs. It is possible that some users have high tolerance, while some others have low tolerance. If we enforce the same maximal role assignment constraint, the constraint may be too strict for some users, which limits the feasible solution space and results a poor role set, and too loose for some other users, which leads to a role set of low score in user satisfaction. To deal with the problem, we propose and study the personalized RMP variants. The core is the concept of *personalized tolerance*. Instead of exerting the same constraint for all users, we allow users to express their preferences on tolerance of excess role assignments, which can be done by conducting a survey. So for every user, we include a personalized constraint $\sum_j UA_{ij} \leqslant t_i$ (t_i is provided by user i) to a role mining model. The way would guarantee that the derived RBAC solution is tailored to all users and the overall user satisfaction is maximized. So the above discussion motivates us to propose the personalized RMP and approximate personalized RMP. The personalized RMP can be used to model the RBAC system that allows each user to exert his/her own preference on the maximal role assignment. It can also be used

to model the case that the role mining engineer has prior knowledge on the maximal role assignments of all users. The prior knowledge may come from common sense or experience learned from similar organizations and institutions. From the model perspective, the user RMP can be viewed as a special case of the personalized RMP, where all t_i are the same.

Problem 3 (Personalized RMP). Given m users, n permissions, user-permission assignments $UPA_{m\times n}$ and personal preferences on the maximum number of roles, denoted by a vector T, it is to discover user-role assignments $UA_{m\times k}$ and a role set $PA_{k\times n}$ such that: (1) the number of roles k is minimized, (2) the role assignments *UA* and the permission-role assignments *PA* accurately and completely reconstruct the user-permission assignments *UPA*, and (3) each user i gets no more than t_i roles.

The personalized RMP is also a minimization problem. The only difference from the user RMP is that each user is subject to different limits on the maximal role assignments.

$$\min k \quad s.t. \begin{cases} UPA_{m\times n} = UA_{m\times k} \otimes PA_{k\times n}, \\ \sum_j UA_{ij} \leqslant t_i. \end{cases} \tag{3}$$

The approximate personalized RMP is extended from the approximate RMP with incorporation of personalized tolerance. Its formal definition is as below.

Problem 4 (Approximate personalized RMP). The approximate personalized RMP is relaxed from the personalized RMP, such that *UPA* can be inaccurately reconstructed, while the number of errors cannot be greater than a predefined value δ.

It can be described using optimization model as well.

$$\min k \quad s.t. \begin{cases} \sum_{ij} (UPA_{m\times n} - UA_{m\times k} \otimes PA_{k\times n}) \leqslant \delta, \\ \sum_j UA_{ij} \leqslant t_i. \end{cases} \tag{4}$$

The regular RMPs and personalized RMPs differ not only in their formulations, but in solutions. If we apply standard optimization solvers, they would appear no much difference, because a RMP problem and its personalized variant have the same objective and the same number of constraints. But as all RMP variants studied in this paper are hard problems (will be proved later), we have to resort to heuristic algorithms for large instances. Heuristic algorithms designed for a RMP problem and its personalized variant do differ. Intuitively, it is more difficult to solve personalized RMPs as one needs to carter for various needs. As an analogy, it would be relatively easy to teach a class of students with the same background. If students come with different prior knowledge, a teacher may have to slow down to take care of everyone so to maximize the overall class performance. We will provide detailed discussion on the difference of solutions later.

4. Theoretical analysis and optimization model

In this section, we analyze computational complexity and build optimization models for the four user-oriented RMP variants. The results would provide us some insights into the problems. First, we study their computational complexity.

Statement 1. The user RMP, the personalized RMP, and their approximate versions are NP-hard.

The basic RMP is known to be NP-hard, as it can be reduced to the classic NP-hard set cover problem [29]. The basic RMP problem can be further reduced to the user RMP problem. If we set the value of the maximal role assignment to be a sufficiently large number, then the user RMP problem is essentially the same as the basic RMP problem. So the user RMP problem is NP-hard. As the user RMP problem is a special case of the approximate user RMP problem with the error ratio being 0, so the approximate user RMP problem is NP-hard as well. The personalized RMP problem is also NP-hard, as it is generalized from the user RMP. Since the approximate personalized RMP problem is generalized from the personalized RMP problem, so it is NP-hard as well.

Next, we will build optimization models for the four problems. Among many existing role mining approaches, the optimization approach has been favored by researchers, due to the existence of many public and commercial optimization software packages. The user-oriented RMP problems can be formulated by optimization models as well, which enables an engineer to directly adopt an existing software package. In the previous section, when we define the four problems, we provide rough optimization models, which cannot be directly fitted into any optimization software package. In the following, we will elaborate the optimization models and put them in the standard optimization form.

We formulate the user RMP first, which can be viewed as a variant of the basic RMP with a constraint that the number of user-role assignments is less than δ. Suppose we have located a set of q candidate roles, represented by a binary matrix $\boldsymbol{CR} \in \{0,1\}^{q\times n}$, where $\boldsymbol{CR}_{kj} = 1$ means candidate role k contains permission j. Then the user-oriented exact RMP is reduced to finding the minimum roles from $\boldsymbol{CR}$ to completely reconstruct existing user-permission assignments while no one can have more than t roles. The problem can be formulated as the following integer linear program (ILP).

$$\begin{aligned}
&\text{minimize } \sum_{k=1}^{q} d_k \\
&\begin{cases}
\sum_{k=1}^{q} \boldsymbol{UA}_{ik}\boldsymbol{CR}_{kj} \geqslant 1, & \text{if } \boldsymbol{UPA}_{ij} = 1, \\
\sum_{k=1}^{q} \boldsymbol{UA}_{ik}\boldsymbol{CR}_{kj} = 0, & \text{if } \boldsymbol{UPA}_{ij} = 0, \\
d_k \geqslant \boldsymbol{UA}_{ij}, & \forall i,j, \\
\sum_{j} \boldsymbol{UA}_{ij} \leqslant t & \forall j, \\
d_k, \boldsymbol{UA}_{ij} \in \{0,1\}.
\end{cases}
\end{aligned} \tag{5}$$

In the model, $\boldsymbol{UA}_{ik}$ and d_k are binary variables. The detailed description of the model is given as follows:

- Binary variable UA_{ik} determines whether candidate role k is assigned to user i and binary variable d_k determines whether candidate role k is selected. So the objective function $\sum_k d_k$ represents the number of selected roles.
- The first constraint enforces that if user i has permission j, at least one role containing permission j has to be assigned to user i.
- The second constraint enforces that if user i has no permission j, no role containing permission j can be assigned to user i.
- The third constraint $d_k \geqslant UA_{ij}$ ensures d_k to be 1 as long as one user has role k.
- $\sum_j UA_{ij} \leqslant t$ enforces that no user can have more than t roles.

The approximate user RMP can be viewed as a relaxed version of the user RMP by not strictly enforcing the reconstruction conditions. As opposed to top-down role discovery approaches, which define roles by examining the semantic meaning of a large number of operational processes and user responsibilities, role mining is a bottom-up approach, which mines roles by analyzing patterns from existing user-permission assignments. So role mining is typically suitable for implementing RBAC systems for large-scale organization. As role mining discovers roles purely from existing assignments, so the result may not satisfy practical needs and thus additional steps may be required. From that perspective, it may not be necessary to discover a role set and role assignments to completely reconstruct the excising assignments. By relaxing the competentness constraint, we might be able to discover better roles. It is the rational of the approximate spareness RMP.

To model the approximation constraint, we introduce some slacking variables to the constraints that enforce completeness. The final optimization model is as below.

$$\text{minimize} \sum_t d_k$$

$$\begin{cases} \sum_{k=1}^{q} UA_{ik} CR_{kj} + V_{ij} \geqslant 1, & \text{if } UPA_{ij} = 1, \\ \sum_{k=1}^{q} UA_{ik} CR_{kj} - V_{ij} = 0, & \text{if } UPA_{ij} = 0, \\ MU_{ij} - V_{ij} \geqslant 0, & \forall i, j, \\ U_{ij} \leqslant V_{ij}, & \forall i, j, \\ \sum_{i,j} U_{ij} \leqslant \delta, & \\ d_k \geqslant UA_{ik}, & \forall i, k, \\ \sum_j UA_{ij} \leqslant t & \forall i, \\ d_j, UA_{ik}, U_{ij} \in \{0, 1\}, V_{ij} \geqslant 0. & \end{cases} \tag{6}$$

In the model, UA_{ik}, V_{ij}, U_{ij} and d_k are variables and M is a large enough constant. The detailed descriptions of the model are given as follows:

- In the first two constraints, V_{ij} acts as an auxiliary variable. Without V_{ij}, the constraints would enforce the exact coverage as the ILP model for the user-oriented exact RMP. With the existence of V_{ij}, the exact coverage constraint is relaxed. The value of V_{ij} indicates whether the constraint for element (i, j) is violated.

- The third and fourth constraints convert V_{ij} to a binary value U_{ij}. If V_{ij} is 1, which means the constraint for element (i, j) is violated, U_{ij} has to be 1; otherwise U_{ij} is 0. The fifth constraint $\sum_{i,j} U_{ij} \leqslant \delta \cdot \sum_{ij} UPA_{ij}$ enforces the error rate to be less than δ.
- The constraint of $d_k \geqslant UA_{ik}\ \forall i, k$ enforces d_k to be 1 as long as a user is assigned to role K. So the objective function represents the number of roles being selected.
- $\sum_j UA_{ij} \leqslant t$ ensures no user can be assigned to more than t roles.

The optimization model for the personalized RMP problem can be obtained by simply modifying the optimization model for the user RMP problem such that the constraint $\sum_j UA_{ij} \leqslant t$ is changed to $\sum_j UA_{ij} \leqslant t_i$ to reflect personal preferences on the maximal role assignments. Similarly, the optimization model for the approximate personalized RMP problem can be obtained by modifying the optimization model for the approximate user RMP problem.

5. Heuristic algorithms

The optimization models in the previous section allow us to directly adopt fruitful optimization research results. However, the problems are NP-hard in nature, which means that for problems of mid to large data size, specially designed efficient heuristics are still required. So in this section, we provide fast heuristic algorithms for the presented four problems. We start with the approximate personalized RMP problem, as other problems can be viewed as its special cases. For instance, if all users have the same limit on the maximal role assignments, the approximate personalized RMP problem is equivalent to the approximate user RMP problem. If the allowed approximation errors are 0, then the approximate personalized RMP problem is equivalent to the personalized RMP problem. So a solution to the approximate personalized RMP problem can be easily extended to the other three problems. In the following, we assume that no two users have the same permissions in a data set, because duplicate users do not affect the final RBAC solution. This assumption is also employed in other role mining methods, e.g. [32]. In practice, we can achieve that by simply removing users with the same permission assignments. To do so, we group all users who have the exact same set of permissions, which can be done in a single pass over the data by maintaining a hash table of the sets of permissions gradually discovered.

Recall that the approximate personalized RMP is to find a minimum set of roles to reconstruct the existing user-permission assignments with the constraints that (i) the maximal role assignments for user i are not more than a given number t_i and (ii) the total reconstruction errors are less than a predefined threshold δ.

The general structure of our algorithm is to iteratively choose a candidate role and assign it to users until the total reconstruction errors are less than the given threshold δ, while the constraints on the maximal role assignments for all users are carefully enforced from start to finish. The procedure can be described in Algorithm 1, where *RSelector* and *CGenerator* refer to the function of selecting a candidate role form the candidate pool and the function of generating candidate roles respectively.

Indeed, the approach of discovering roles in an iterative fashion has been observed in many role mining methods, e.g. the Lattice Model [3], the FastMinder [32] and the optimization-based heuristic [14]. The distinguishing elements of our algorithm are (i) the way of generating candidate roles and (ii) the rule for role selection at each iteration.

Algorithm 1. Approximate personalized RMP

Input: $UPA, t_1, \ldots, t_m$
Output: UA, PA

$UA \leftarrow \emptyset, PA \leftarrow \emptyset, UPA' \leftarrow \emptyset$;
$CRoles \leftarrow UPA$; /* Generate initial candidate roles*/
while $\|UPA' - UPA\|_1 \leqslant \delta$ **do**
  while $\exists i$, s.t. $|UA_i| = t_i - 1$ **do**
    $NewRole \leftarrow UPA_i \backslash UPA'_i$
    $PA \leftarrow PA \cup NewRole$ /* update PA */
    $UA_{ij} = 1$ if $NewRole \subseteq UPA_i$ /* update UA */
  end while
  Call RSelector /* Select a candidate role */
  Call CGenerator /* Update candidate roles */
end while

5.1. Candidate role generation

One key feature of our algorithm is a dynamic role generation strategy. Given n permissions, there are 2^n possible roles. If we consider too many candidate roles, the computing time is expensive. Conversely, if we consider only a very limited set of candidate roles, we might not be able to find a good role set. To avoid the two extreme cases, our strategy is dynamic candidate role generation. Specifically, rather than generating a static set of roles at the start of the algorithm, we generate a small set of promising roles at each iteration of the algorithm and the role set is updated according to the remaining user-permission assignments as the algorithm proceeds. There are two benefits: (i) we do not need to maintain and consider a large pool of candidate roles all the time; (ii) the candidate role pool always keeps the potentially interesting roles.

Motivated from the existing role mining literature, we consider three different ways of generating candidate roles. The first way is called *itself*, which is inspired by [16]. In [16], every user's permission set $UPA_{i\cdot}$ is treated as a candidate role. All candidate roles are generated once and then some algorithm is applied to iteratively select roles from the candidate role pool. Differently, our *itself* strategy is dynamic in nature. Initially, we treat every user's permission set as a candidate role. Then a candidate role is selected from the initial candidate pool and assigned to appropriate users. Subsequently, we treat every user's uncovered permission set as a candidate role. In other words, the candidate role pool is updated and the next role is selected from the updated candidate pool. The above procedure continues until some termination condition is satisfied. To illustrate it, we use Table 2 as an example. *itself* initially makes (001111), (001110) and (111100) as candidate roles. Suppose (001110) is selected and is assigned to all users. Resultantly, the remaining permissions for $u1$–$u3$ are (000001), (000000) and (110000), respectively. Then (000001) and (110000) are used as candidate roles for the next step.

The second method is called *intersection*, which is inspired by [32]. In [32], they generate an initial candidate role set by making every user's permission set $UPA_{i\cdot}$ as a candidate role. In addition to that, they include all of the potentially interesting roles by computing all possible intersection sets of all the roles created in the initial phase. Given n initial roles, there are 2^n possible role intersections. It is not possible to consider all of them. So the common practice is to consider the intersections of every pair of

Table 2
Existing assignments

	p1	p2	p3	p4	p5	p6
$u1$	0	0	1	1	1	1
$u2$	0	0	1	1	1	0
$u3$	1	1	1	1	0	0

Algorithm 2. CGenerator – itself

Input: *UPA*, *UPA′*
Output: *CRoles*
1: *CRoles* $\leftarrow \emptyset$;
2: **for** $i = 1 \rightarrow |UPA|$ **do**
3: **if** $UPA_i \setminus UPA'_i \neq \emptyset$ **then**
4: $CRoles \leftarrow \{CRoles, UPA_i \setminus UPA'_i\}$
5: **end if**
6: **end for**

initial roles. In [32], candidate roles are generated once and then used throughout a role mining process. To incorporate the dynamic feature, we update the candidate role set after every iteration, by using the remaining user-permission assignments, denoted *UPA′*. The way of generating candidate roles from *UPA′* is the same as that for the original user-permission assignments *UPA*. To illustrate it, consider Table 2 again. The initial candidate roles are: (001111), (001110), (111100) and (001100) that is the intersection of $u2$ and $u3$. Note that (001100) is also the intersection of $u1$ and $u2$, and $u1$ and $u3$. The candidate role set only keeps unique roles. Suppose (001110) is selected and assigned to $u2$ only. Then the uncovered assignments for all users are: (001111) and (111100). For the next step, the candidate roles would include the uncovered assignments (001111) and (111100), and their intersection (001100).

The third method is called *association*, which is inspired by [24]. The original *association* method takes advantage of the relationship between permissions (columns of a binary matrix) and generates candidate roles of permissions, which frequently occur together. Given an input matrix *UPA* of $m \times n$, a binary association matrix A of size $n \times n$ (each row is a candidate role) is constructed such that $A_{ij} = 1$ if the ratio of users having both permissions i and j is greater than τ, which is a value between 0 and 1. Existing results showed that τ being 0.9 usually leads to a good result, which will be used in the experimental study section. To incorporate the dynamic flavor, our *association* method runs iteratively. At each iteration, we generate candidate roles by running the original association method on the remaining *UPA*.

For illustration purpose, we formally present the itself candidate role generation method in Algorithm 2. The other two candidate role generation methods can be derived from it.

5.2. Role selection and assignment

In this section, we study how to select roles from a candidate role pool and assign them to users. We also discuss how to enforce maximal role assignment constraints for each user along the iterative role selection process. We tried the following candidate role selection strategies. (i) *greedy* – choose the candidate role that covers the most remaining permission assignments; (ii) *fewest* – select the candidate role

Algorithm 3. RSelector – greedy

Input: UPA, UPA', $CRoles$, UA, t
Output: r, UA, PA, UPA'

```
if ∃i s.t. |UA_i:| = t − 1 then
    temp ← UPA_i: \ UPA'_i:
    if ∃j s.t. UPA_j: ⊇ temp, j ≠ i then
        r ← temp
    else
        r ← UPA_i:
        UA_i: ← ∅
    end if
else
    r ← argmax_{r∈CRoles} |UPA_i: s.t. UPA_i ⊇ r|
end if
Update PA by including r;
Update UA by assigning r to all valid users;
Update UPA';
```

with the fewest permissions; (iii) *most* – select the candidate role with the most permissions; (iv) *rand* – choose a random candidate role. Among which, the first strategy is widely used in heuristic algorithms for various role mining problems. But it is the most time consuming, because at each iteration the algorithm needs to examine all candidate roles in order to find the best one. The last strategy is the fastest one, as there is nearly no time cost for role selection. But since the selection process is blinded, the quality of the returned role set has no guarantee. The time cost of the second and third strategies are in the between. They have been used to solve the basic RMP problem in [3] and are reported of decent results. For illustration purpose, the complete procedure of the greedy role selection method is given in Algorithm 3. We will omit the algorithm for the other selection methods, as it is not difficult to derive them by modifying Algorithm 3.

To enforce the constraint that no user gets more than t_i roles, we make some special arrangement, when a user U_i has been covered by $t_i - 1$ roles and still has uncovered permissions. In such a case, we create a new role of the remaining permissions of U_i and then assign the new role to the users who have all permissions in the new role. We repeat the procedure until there is no user reaching the limit of the maximal role assignments. This step is to enforce the constraints on the maximal role assignments. So technically we utilize two ways of role generation, one being selective and the other being compulsory. The compulsory role generation step is reflected in Algorithm 1. Note that in the algorithm the outer while loop is used to monitor the coverage of *UPA*. The loop terminates when the approximation constraint is satisfied.

5.3. Computational complexity analysis

The key of the above user-oriented role mining algorithm is the continuous updating of candidate roles. At each iteration of the algorithm, candidate roles are generated and a candidate role is selected. So the total computations then depend on the number of iterations, the candidate role generation method and

the candidate role selection method. Consider the most computationally expensive one, the intersection candidate role generation method and the greedy role selection method. Suppose the data set consists of m users and n permissions. According to our algorithm, at each iteration, at least one user's permissions are completely covered. So the maximum required iterations are m. At each iteration step, each candidate role is compared against each remaining user. Denote the number of candidate roles as l, which is m for the itself candidate role generation method, m^2 for intersection, and n for association. At each iteration, the number of remaining users is less than m and each user (or role) has up to n permissions. So the incurred computations at each iteration cannot be over mnl. Therefore the computation complexity of our algorithm is upper bounded by m^2nl.

6. Experiments and results

In this section, we conduct experiments on both synthetic and real data sets to examine the effect of the maximal role assignment constraint on the mined RBAC solutions and evaluate the performance of the heuristic algorithms.

6.1. Synthetic data sets

The synthetic data sets are generated as follows. Firstly, generate a set of unique roles $PA_{k\times n}$ and unique user-to-permission assignments $UA_{m\times k}$ in a random fashion. *UPA* is the Boolean product of *PA* and *UA*. Two parameters ρ_1 and ρ_2 are employed to control the density of 1's cells in *PA* and *UA* respectively, which then determine the density of 1's cells in *UPA*. Note that each row of *UA* and *PA* is ensured to have at least one 1's cell.

The first experiment is to study the effect of the maximal role assignment constraints with respect to the evaluation criterion, the number of roles. We utilize the heuristic algorithm with the itself candidate role generation method and the greedy candidate role selection method. We generate *UA*, *PA* and *UPA* with m of 200, n of 50, k of 15, ρ_1 of 0.2 and ρ_2 of 0.2. In the generated data set, the maximum role assignment is 7 and the minimum is 1. Firstly, we consider the user RMP, in which all users share the same constraint on the maximal role assignments. We vary the maximal role assignment constraint from 10 to 4 and see how it affects the RBAC solutions. The experimental results are reported in Fig. 1. We observe that the maximal role assignment constraint affects the RBAC solution significantly. When the maximal number of role assignments is greater than 7, it does not affect the RBAC solution at all, as the number of the returned roles is 14, which is the same as the underlying truth. However, when

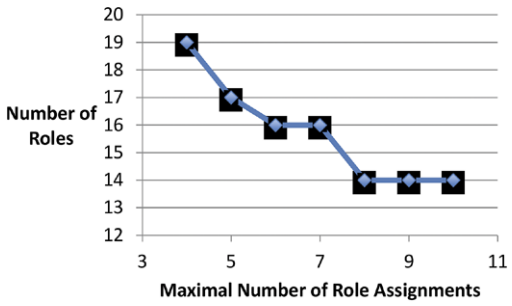

Fig. 1. Results on user RMP. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/JCS-140519.)

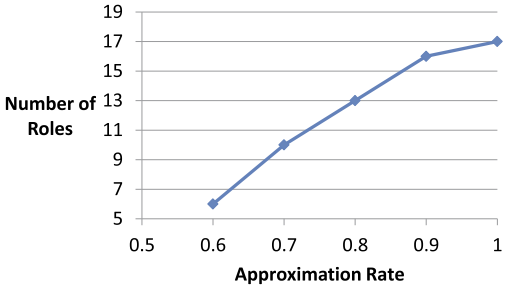

Fig. 2. Results on approximate user RMP. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/JCS-140519.)

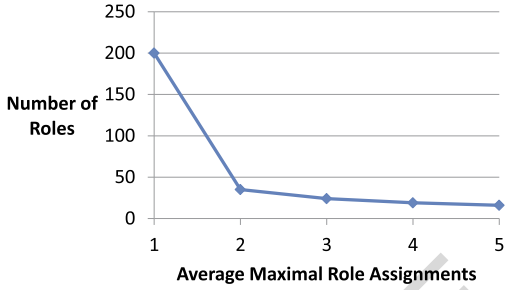

Fig. 3. Results on personalized RMP. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/JCS-140519.)

the maximal number of role assignments decreases, the number of roles increases rapidly. When the constraint is 4, the number of returned roles becomes 19. But the result does not deviate significantly from the underlying truth, which in some way demonstrates the performance of our algorithm. Secondly, we study the approximate user RMP and examine the effect of the approximation rate on the number of roles. For the same data set, we set the maximal role assignment constraint to be 5 and then vary the approximation rate from 0.6 to 1. As reported in Fig. 2, the size of the mined role set increases as the approximate rate increases. It is because more roles are needed for greater coverage. We also observe that the increasing trend slows when the approximation rate increases. It is because the algorithm runs in a greedy fashion. The roles picked earlier have greater coverage. Thirdly, we study the personalized RMP. According to the synthetic data generation process, the average number of role assignments is $\rho_2 \times k$, which is 3 in this case. We pose a random maximal role assignment constraint on each user. The maximal role assignment for user i is a random number drawn from a Poisson distribution. We vary the mean of the Poisson distribution. The higher mean indicates looser constraints in general. Note that as the constraint is randomly generated, so it does not mean that the user supposed to be assigned to more roles would have a greater value of the maximal role assignments. As reported in Fig. 3, the average maximal role assignments play a significant role on the RBAC solution. The number of roles increases rapidly when the average maximal role assignments decreases. On the extreme case when the number of the average maximal role assignments is 1, the system has to generate a unique role for each user. Secondly, we study the approximate personalized RMP and examine the effect of the approximation rate on the number of roles. The experiment is conducted on the same data set. For each user, we set a maximal role assignment constraint as a positive random integer with mean of 5. We vary the approximation rate from 0.6 to 1. The experimental results plotted in Fig. 4 shows a similar trend as that in Fig. 2 that the number of roles increases as the approximation rate increases.

The second experiment is to study the algorithms. The framework of our algorithms is dynamic and iterative in nature. At each iteration, a set of candidate roles are generated and then some role selection method is applied. The iteration terminates when some condition is satisfied. We tried several candidate role generation methods, including itself, intersection, and association, and several role selection method, including greedy, fewest, most and random. Firstly, we study the role generation methods. We consider the user RMP problem and run the iterative algorithm with the three different candidate role generation methods respectively coupled with the same greedy candidate role generation method. All experiments are run against the same data set used before. The maximal role assignment constraint is set to 3. The experimental results are reported in Table 3. It shows that itself and association have the comparable computing time cost, while intersection takes much more time. The reason is that intersection

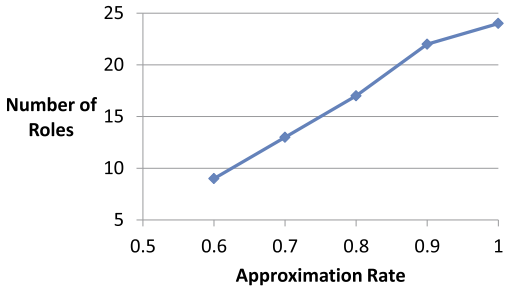

Fig. 4. Results on approximate personalized RMP. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/JCS-140519.)

Table 3
Comparison on number of roles for candidate role generation methods

	Time (s)	Number of roles
Itself	6	17
Intersection	42	16
Association	7	20

generates O(m^2) candidate roles at each time, while itself and association produce m and n candidate roles respectively. In terms of the number of roles, intersection has the best performance, as it returns a role set of size 16, while itself 17 and association 20. The results suggest that for small and medium data sets intersection is preferred and for large data sets itself should be chosen.

Secondly, we study the role selection methods. The experimental setting is as follows. The user RMP problem is considered. A testing data set is generated with m and n fixed to 50 and the maximal role assignment to 3. We vary the true number of roles k from 4 to 10 and the density rate of 1's elements from 0.05 to 0.25. For each data generation parameters set, we generate five matrices. We run the iterative algorithm coupled with the greedy, fewest, most, and rand candidate role generation methods respectively against the testing data sets. We report the average results across the five matrices generated with the same parameters setting in Figs 5–8. We observe that in terms of the number of roles, the fewest and greedy methods have comparable performance and are better than the most and rand method. We also observe that in terms of computing time, the greedy has the worst performance. The reason is that the greedy candidate role generation method needs to compute the contribution of every candidate role, which takes much time. While, the fewest and most methods only need to count the number of contained permission for every candidate role. The rand method incurs the least computing time, because a random candidate role is randomly picked at each iteration. Surprisingly, although the random method has no rational behind, its performance in terms of both the number of returned roles and computing time is decent. It suggests that for large instances an iterative role search algorithm coupled with the random candidate role selection method could be a practical solution. Another surprising observation is that for quite a few parameter settings, the fewest method performs better than the greedy method. Greedy search is a classic searching strategy used in optimization. It intends to find a global optimal (or good) solution by traversing a series of local optimal solutions. The fewest method, mathematically speaking, removes a clique with few edges from a bipartite graph at each iteration. But it is hard to mathematically explain why the fewest method performs better than the greedy method. But the experimental result implies that in practice one could try both methods and pick the better returned solution.

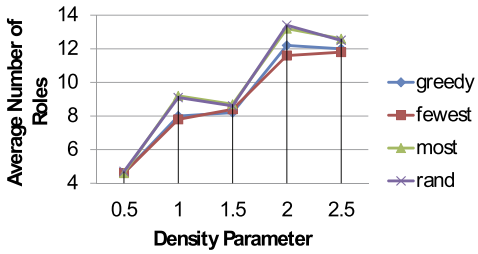

Fig. 5. Results on average number of roles by varying density parameter. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/JCS-140519.)

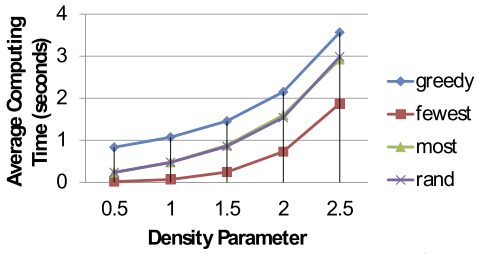

Fig. 6. Results on average computing time by varying density parameter. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/JCS-140519.)

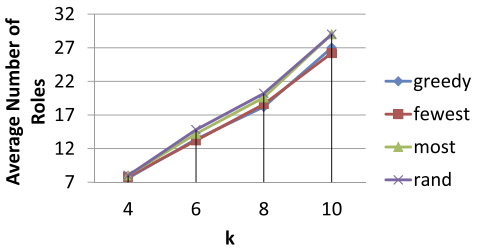

Fig. 7. Results on average number of roles by varying true number of roles. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/JCS-140519.)

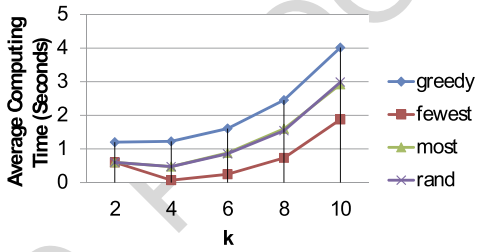

Fig. 8. Results on average computing time by varying true number of roles. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/JCS-140519.)

6.2. Real data sets

We also conduct experiments on benchmark access control data sets. They are *americas_small*, *apj*, *healthcare*, *domino*, *firewall1* and *firewall2*, which can be found at the HP website.[2] *americas_small* and *apj* are user profiles from Cisco firewalls. *healthcare* was obtained from the US Veteran's Administration. The *domino* graph is from a set of user and access profiles for a Lotus Domino server. *firewall1* and *firewall2* are results of running an analysis algorithm on Checkpoint firewalls. These results assert whether packets can reach from some sources to some destinations. Sources and destinations are disjoint IP address ranges. The *firewall1* and *firewall2* are already in the two-dimensional binary matrix form. Rows and columns correspond to disjoint IP address ranges. A 1's cell indicates that a packet reaches from the IP address range corresponding to the row to the IP address range corresponding to the column. The approach that we take to solve the *firewall1* and *firewall2* instances is the same as what are used for the other data sets, as all data sets come in a two-dimensional binary matrix form and need to be decomposed into the product of two binary matrices such that the constraints are satisfied and the objective function is optimized. Note that the *firewall1* and *firewall2* data are different from the firewall policy data studied in [10]. The firewall policy data studied in [10] are many policy rules, e.g. subject A can perform operation B on object C. The contribution of [10] is to represent the raw firewall policy data as a three-dimensional binary matrix (subject–operation–object) and then transforms it into a two-dimensional binary matrix (subject–[operation, object]), so that any bottom-up role mining approach based on Boolean matrix decomposition can be applied here. So the approach in [10] and ours are not comparable. Descriptions on the data sets including the number of users, the number of permissions, and

[2] http://www.hpl.hp.com/personal/Robert_Schreiber/.

Table 4
Data description

Data set	$\|U\|$	$\|P\|$	$\|UPA\|$
healthcare	46	46	1,486
domino	79	231	730
firewall1	365	709	31,951
firewall2	325	590	36,428
apj	2,044	1,164	6,841
americas small	3,477	1,587	105,205

the size of user-permission assignments are given in Table 4. More detailed descriptions can be found in [3].

The first experiment evaluates the user-oriented exact RMP. We want to know whether our user-oriented role mining approach can effectively enforce the maximal role assignment constraint and whether the output of the algorithm is comparable to the optimal RBAC solution without the constraint. From the previous experimental results on synthetic data sets, we learned that the itself candidate role generation method coupled with the greedy candidate role selection method returns good solutions and also runs fast. So in the following experiments, we will use that strategy. We also learned that our algorithm exhibits similar behavior on the four user-oriented role mining variants. So for convenience, we here only consider the user RMP problem, for which all users have the same maximal role assignment constraint. For reference convenience, we call our user-oriented role mining algorithm as *Dynamic*. We run *Dynamic* algorithm on those real data sets with different maximal role assignment constraints. We compare our results with the benchmark role mining algorithm, *Lattice* [3]. As far as we know, *Lattice* has the best reported result with respect to the minimization of the number of roles and the minimization of the system administrative cost. The experimental results are reported in Tables 5–8. In these tables, δ denotes the approximation rate. The exact RMP requires the approximation rate to 1. So we only look at the portion of the results with $\delta = 1$. Other parameters are: t denotes the maximum number of role assignments enforced in our algorithm, $|UA|$ denotes the size of user-permission assignments and $|PA|$ denoting the size of permission-role assignments. Note that δ and t has no effect on the *Lattice* algorithm, as *Lattice* returns an exact RBAC solution and the solution is unique.

In the results, when t decreases, the size of UA decreases accordingly. However, the value of $|UA| + |PA|$ changes in an opposite direction. This matches our expectation. Specifically, when t is a small value, each user gets few role assignments. Thus, we need roles with more permissions, so each user can still get enough permission assignments. When the constraint becomes more strict, the number of required roles increases. As a result, the value of $|PA|$ increases accordingly.

Furthermore, we are pleased to see that even with the constraint being enforced, the complexity of the RBAC solution returned by *Dynamic* is still comparable to that of *Lattice*. For example, in Table 9, when t is 2, *Dynamic* returns a RBAC solution with only 19 roles, while *Lattice* returns a solution with 15 roles and the maximum role assignments of 4. Another observation is that the $|UA|$ value of the solutions returned by *Dynamic* can be much less than that of the solutions returned by *Lattice*. For instance, in Table 6, the value of $|UA|$ for *Dynamic* with δ of 0 and t of 2 is 3,621, while that for *Lattice* is 4,782.

The second experiment is to study the user-oriented approximate RMP. We want to know how the RBAC solution varies with the approximation rate. We run the *Dynamic* algorithm by varying the value of δ from 0.95 to 0.80. Results are reported in Tables 5–8. We observe that when the complexity of the RBAC solutions decrease drastically when δ increases. For instance, in Table 5, with t of 8, only 39 roles are required to cover the 95 percent of permission assignments (i.e., $\delta = 0.95$), while 88 roles

Table 5
fire1

	δ	t	$\|R\|$	$\|UA\|$	$\|UA\| + \|PA\|$
Dynamic	1.00	2	88	406	7,204
		4	85	454	6,890
		6	83	600	6,897
		8	80	1,516	6,638
	0.95	2	36	255	5,738
		4	48	361	5,685
		6	41	529	5,523
		8	39	1,416	5,671
	0.90	2	29	272	4,672
		4	27	330	4,347
		6	25	482	3,868
		8	14	1,464	2,970
	0.85	2	17	250	2,661
		4	9	422	1,762
		6	8	573	1,852
		8	7	1,334	2,055
	0.80	2	10	287	1,912
		4	8	426	1,655
		6	6	1,131	1,839
		8	6	1,131	1,839
Lattice	1.00	9	66	874	1,953

are required for the complete coverage (i.e., $\delta = 1$). In cases where data noise is believed to exist, the approximate version of *Dynamic* appears to be more useful.

To summarize, the two experiments have demonstrated the effectiveness of our user-oriented RMP approach. We highlight that one primary advantage of *Dynamic* is that it allows a RBAC engineer to tune the maximal role assignment constraint to reflect the real need. This feature is not supported by any existing role mining method. More importantly, the overall system complexity of the resultant solution is comparable to that of the optimal solution without any constraint.

7. Conclusion

In this paper, we studied the role mining problem from the end-user perspective. Unlike other existing role mining approaches which primarily aim to reduce the administrative workload, our approach strives to incorporate better user experience into the role decision process. As end-users prefer simple role assignments, we add a constraint that mandates the maximum number of roles a user can have to the role mining process. The number usually can be determined in practice by a brief study on the general business processes of an organization. Basing on this rationale, we formulated user-oriented role mining as four specific problems, the user RMP, the approximate user RMP, the personalized RMP, and the approximate personalized RMP. The user RMP is applicable to the case where there is a prior knowledge on the maximal role assignments that a user can take. The personalized RMP is applicable the case that users are allowed to express their own interests on the upper limits of role assignments. Their approximation versions can be used in cases, where exact completeness is not required, to explore interesting roles. We

Table 6
americas_small

	δ	t	$\lvert R\rvert$	$\lvert UA\rvert$	$\lvert UA\rvert + \lvert PA\rvert$
Dynamic	1.00	2	257	3,621	25,934
		4	254	3,932	23,610
		6	246	4,269	22,194
		8	246	4,269	22,194
	0.95	2	224	3,283	23,174
		4	184	3,566	21,349
		6	183	3,780	20,109
		8	183	3,780	20,109
	0.90	2	204	3,420	21,919
		4	155	3,621	20,870
		6	153	3,639	21,214
		8	145	4,102	21,735
	0.85	2	177	3,126	20,929
		4	135	3,442	17,806
		6	132	3,704	16,527
		8	127	4,036	14,698
	0.80	2	170	3,126	18,772
		4	126	3,424	16,486
		6	121	37,022	14,862
		8	115	38,815	14,307
Lattice	1.00	10	192	4,782	9,830

Table 7
fire2

	δ	t	$\lvert R\rvert$	$\lvert UA\rvert$	$\lvert UA\rvert + \lvert PA\rvert$
Dynamic	1.00	2	11	392	1,609
	0.95	2	11	392	1,609
	0.90	2	6	301	1,033
	0.85	2	6	301	1,033
	0.80	2	6	301	1,033
Lattice	1.00	3	10	434	1,110

Table 8
domino

	δ	t	$\lvert R\rvert$	$\lvert UA\rvert$	$\lvert UA\rvert + \lvert PA\rvert$
Dynamic	1.00	2	23	114	721
	0.95	2	17	97	693
	0.90	2	14	92	689
	0.80	2	10	87	672
Lattice	1.00	3	20	110	713

Table 9
healthcare

	δ	t	$\|R\|$	$\|UA\|$	$\|UA\| + \|PA\|$
Dynamic	1.00	2	19	54	432
		3	18	77	411
	0.95	2	5	52	191
		3	6	54	201
	0.90	2	5	51	222
		3	6	54	201
	0.85	2	5	49	193
		3	6	54	201
	0.80	2	5	48	190
		3	3	72	126
Lattice	1	4	15	106	209

Table 10
apj

	δ	t	$\|R\|$	$\|UA\|$	$\|UA\| + \|PA\|$
Dynamic	1.00	2	564	2,310	6,015
		3	497	2,802	5,912
		4	485	2,911	5,528
	0.95	2	463	2,264	5,375
		3	392	1,621	4,428
		4	380	1,573	4,335
	0.90	2	405	1,312	4,490
		3	356	1,452	4,232
		4	352	1,501	4,043
	0.85	2	344	892	4,012
		3	311	1,487	3,851
		4	311	1,487	3,851
	0.80	2	313	892	3,812
		3	288	1,145	3,580
		4	288	1,145	3,580
Lattice	1.00	6	454	2,437	4,117

studied existing role mining methods, and found that some of them can be applied to our problems with simple modification. For better efficiency, we also designed new algorithms tailored to our problems, which are based on a dynamic candidate role generation strategy. Experimental results demonstrate the effectiveness of our approach in discovering a user-oriented RBAC solution while without increasing the overall administrative workload too much.

Future work can go along two directions. One is to study the feasibility of employing some statistical measures such as Bayesian information criterion to facilitate the role mining process. The motivation is that sometimes the accurate sparseness constraint (the maximum role that a user can have) is not available. We could employ some statistical criteria to choose the RBAC model with a good balance of model complexity and describability. The other direction is to consider the dynamic sparseness constraint. In this work, we assume that the same sparseness constraint is enforced to everyone. However, it might

be the case that some user requires many role assignments due to some need. In such cases, a more practical role mining approach is to minimize the sparsity of the whole user-role assignments rather than enforcing a sparseness constraint for every user.

References

[1] K. Brooks, Migrating to role-based access control, in: *ACM Workshop on Role-Based Access Control*, 1999, pp. 71–81.
[2] E.J. Coyne, Role engineering, in: *RBAC'95: Proceedings of the First ACM Workshop on Role-based Access Control*, ACM, New York, NY, USA, 1996, p. 4.
[3] A. Ene, W. Horne, N. Milosavljevic, P. Rao, R. Schreiber and R.E. Tarjan, Fast exact and heuristic methods for role minimization problems, in: *SACMAT'08: Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*, ACM, New York, NY, USA, 2008, pp. 1–10.
[4] A. Ene, W.G. Horne, N. Milosavljevic, P. Rao, R. Schreiber and R.E. Tarjan, Fast exact and heuristic methods for role minimization problems, in: *SACMAT*, 2008, pp. 1–10.
[5] E.B. Fernandez and J.C. Hawkins, Determining role rights from use cases, in: *Proceeding of the 2nd ACM Symposium on Access Control Models and Technologies, SACMAT'97*, 1997.
[6] M. Frank, D. Basin and J.M. Buhmann, A class of probabilistic models for role engineering, in: *Proceedings of the 15th ACM Conference on Computer and Communications Security*, 2008.
[7] M. Frank, A.P. Streich, D. Basin and J.M. Buhmann, A probabilistic approach to hybrid role mining, in: *Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS'09*, ACM, New York, NY, USA, 2009, pp. 101–111.
[8] F. Geerts, B. Goethals and T. Mielikainen, Tiling databases, in: *Discovery Science*, Springer, 2004, pp. 278–289.
[9] Q. Guo, J. Vaidya and V. Atluri, The role hierarchy mining problem: Discovery of optimal role hierarchies, in: *ACSAC*, 2008, pp. 237–246.
[10] S. Hachana, F. Cuppens, N. Cuppens-Boulahia, V. Atluri and S. Morucci, Policy mining: A bottom-up approach toward a model based firewall management, in: *ICISS*, 2013, pp. 133–147.
[11] A. Kern, M. Kuhlmann, A. Schaad and J.D. Moffett, Observations on the role life-cycle in the context of enterprise security management, in: *SACMAT*, 2002, pp. 43–51.
[12] M. Kuhlmann, D. Shohat and G. Schimpf, Role mining – revealing business roles for security administration using data mining technology, in: *SACMAT'03: Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies*, ACM, New York, NY, USA, 2003, pp. 179–186.
[13] H. Lu, Y. Hong, Y. Yang, L. Duan and N. Badar, Towards user-oriented RBAC model, in: *DBSec*, 2013, pp. 81–96.
[14] H. Lu, J. Vaidya and V. Atluri, Optimal Boolean matrix decomposition: Application to role engineering, in: *IEEE 24th International Conference on Data Engineering*, 2008, pp. 297–306.
[15] H. Lu, J. Vaidya and V. Atluri, An optimization framework for role mining, *Journal of Computer Security* **22**(1) (2014), 1–31.
[16] H. Lu, J. Vaidya and V. Atluri, An optimization framework for role mining, *Journal of Computer Security*, accepted.
[17] H. Lu, J. Vaidya, V. Atluri and Y. Hong, Extended Boolean matrix decomposition, in: *IEEE International Conference on Data Mining*, 2009.
[18] H. Lu, J. Vaidya, V. Atluri and Y. Hong, Constraint-aware role mining via extended Boolean matrix decomposition, *IEEE Transactions on Dependable and Secure Computing* **9**(5) (2012), 655–669.
[19] X. Ma, R. Li and Z. Lu, Role mining based on weights, in: *SACMAT*, 2010, pp. 65–74.
[20] I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. Calo and J. Lobo, Mining roles with semantic meanings, in: *SACMAT'08: Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*, ACM, New York, NY, USA, 2008, pp. 21–30.
[21] I. Molloy, N. Li, T. Li, Z. Mao, Q. Wang and J. Lobo, Evaluating role mining algorithms, in: *SACMAT'09: Proceedings of the 14th ACM Symposium on Access Control Models and Technologies*, ACM, New York, NY, USA, 2009, pp. 95–104.
[22] I. Molloy, N. Li, Y.A. Qi, J. Lobo and L. Dickens, Mining roles with noisy data, in: *Proceeding of the 15th ACM Symposium on Access Control Models and Technologies, SACMAT'10*, ACM, New York, NY, USA, 2010, pp. 45–54.
[23] G. Neumann and M. Strembeck, A scenario-driven role engineering process for functional RBAC roles, in: *SACMAT*, 2002, pp. 33–42.
[24] M. Pauli, M. Taneli, G. Aristides, D. Gautam and M. Heikki, The discrete basis problem, *IEEE Transactions on Knowledge and Data Engineering* **20**(10) (2008), 1348–1362.
[25] H. Roeckle, G. Schimpf and R. Weidinger, Process-oriented approach for role-finding to implement role-based security administration in a large industrial organization, in: *ACM Workshop on Role-Based Access Control*, 2000, pp. 103–110.

[26] J. Schlegelmilch and U. Steffens, Role mining with ORCA, in: *SACMAT'05: Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies*, 2005, pp. 168–176.
[27] D. Shin, H. Akkan, W. Claycomb and K. Kim, Toward role-based provisioning and access control for infrastructure as a service (IAAS), *J. Internet Services and Applications* **2**(3) (2011), 243–255.
[28] A.P. Streich, M. Frank, D. Basin and J.M. Buhmann, Multi-assignment clustering for Boolean data, in: *Proceedings of the 26th Annual International Conference on Machine Learning, ICML'09*, ACM, New York, NY, USA, 2009, pp. 969–976.
[29] J. Vaidya, V. Atluri and Q. Guo, The role mining problem: finding a minimal descriptive set of roles, in: *SACMAT*, 2007, pp. 175–184.
[30] J. Vaidya, V. Atluri, Q. Guo and H. Lu, Edge-RMP: Minimizing administrative assignments for role-based access control, *Journal of Computer Security* **17**(2) (2009), 211–235.
[31] J. Vaidya, V. Atluri, Q. Guo and H. Lu, Role mining in the presence of noise, in: *DBSec*, 2010, pp. 97–112.
[32] J. Vaidya, V. Atluri and J. Warner, Roleminer: mining roles using subset enumeration, in: *The 13th ACM Conference on Computer and Communications Security*, 2006, pp. 144–153.