Role Mining Using Answer Set Programming

Wei Ye, Ruixuan Li, Xiwu Gu*, Yuhua Li, Kunmei Wen

School of Computer Science and Technology Huazhong University of Science and Technology Wuhan, P.R.China

Abstract

With the increasing adoption of role-based access control (RBAC) in business security, role mining technology has been widely applied to aid the process of migrating a non-RBAC system to an RBAC system. However, because it is hard to deal with a variety of constraint conflicts at the same time, none of existing role mining algorithms can simultaneously satisfy various constraints that usually describe organizations’ security and business requirements. To extend the ability of role mining technology, this paper proposes a novel role mining approach using answer set programming (ASP) that complies with constraints and meets various optimization objectives, named constrained role miner (CRM). Essentially, the idea is that ASP is an approach to declarative problem solving. Thus, either to discover RBAC configurations or to deal with conflicts between constraints, ASP programs don’t need to specify how answers are computed. Finally, we demonstrate the effectiveness and efficiency of our approach through experimental results.

Keywords: RBAC, Constraint, Role Mining, Answer Set Programming

1. Introduction

Currently, role-based access control (RBAC) [1, 2] has become the predominant access control model because it greatly simplifies the security man-
agement. The key feature of RBAC is that each role is a collection of permissions, and all users acquire permissions only through the roles. However, it is costly to develop and maintain an RBAC system though RBAC reduces the management cost.

In order to build high quality RBAC system, researchers have proposed two important approaches: the top-down approach and the bottom-up approach. The top-down approach [3, 4] often starts with expert analysis of business processes and builds RBAC system from such analysis. However, the top-down approach is time consuming since it is human-intensive [5]. The bottom-up approach can discover roles from existing user-permission assignments automatically. Such a computing-intensive approach is called role mining.

Given the same user-permission assignments, different role mining algorithms will build different RBAC systems. Therefore, it requires a measurement to evaluate how good an RBAC state is. The measurement used in Vaidya et al. [6] is the number of roles while the measurement used in Zhang et al. [7] and Ene et al. [8] is the total number of edges when an RBAC state is represented by a graph visually. Guo et al. [9] aim to minimize the number of roles and the edges in role hierarchy graph. Molloy et al [10] summarized the previous multiple ways of measures and proposed the notion of weighted structural complexity.

Constraint is a defined relationship among roles or a condition related to roles. One of the most common constraints is a separation-of-duty policy. For instance, a user cannot be a member of both mutually exclusion roles. In addition, constraints can be used to reflect business requirements. For example, there is only one person in the role of CEO in a company. As an essential part of the RBAC models, constraints play an important part in defining the security requirements of the system [11, 12, 13, 14].

Nonetheless, one main limitation of existing role mining methods is that, the construction process of an RBAC system can not simultaneously meet various constraints. For example, two constraints are required to be satisfied. The role mining algorithm meets the first constraint but fails to satisfy the second constraint. That means there is a conflict between the two. Then, we should add an algorithm to resolve the conflict in the role mining algorithm. The next step in the example is to add the third constraint. The third constraint may have conflicts with the first two constraints. Thus, we may need to implement more different algorithms to resolve these two conflicts. Meanwhile, the first conflict resolution algorithm probably need-
s to be modified so as to ensure that the first two constraints still do not conflict. Obviously, with the increasing number of constraints, these would become impossible tasks. What’s more, you can not combine various conflict resolution algorithms with the role mining algorithms in many cases.

Leveraging the approach of answer set programming in artificial intelligence, we propose a ASP-based novel approach to construct an RBAC system that can comply with constraints and meet multi-objective optimization at the same time, namely constrained role miner (CRM). In the field of artificial intelligence, ASP has been viewed as an effective programming language for knowledge representation and declarative problem solving [15]. Different from traditional imperative programming languages (C++, Java, etc), it is about ”what to do”, without considering many details of ”how to do”, for solving a problem. ASP allows us to adopt mature ASP solvers that have been proved to work well in practice. Moreover, its rich modeling language eases the understanding and explanation of the problem. With the advantage of ASP, we do not need to implement a variety of specific conflict resolution algorithm, only to describe the constraints problem with ASP modeling language. The case of role mining problem is the same, and the problem can also be solved with ASP approach. Finally, we compute an answer set of the ASP program with ASP solver, and extract the solution if the problem is solvable.

The main contributions of this paper are as follows.

- This paper proposes a novel role mining approach using ASP that can comply with various kinds of constraints and meet multi-objective optimization at the same time, namely constrained role miner (CRM).

- This paper presents experiments to demonstrate the effectiveness of our approach. According to experimental evaluation, CRM is also better than the existing role mining approaches in case of no constraints.

The rest of the paper is organized as follows. We discuss related work in Section 2. In Section 3, we review the notions of role mining problem and the main concepts of ASP. In Section 4, we describe constrained role miner and demonstrate how CRM works by using ASP. In Section 5, We show the results of experiment. Finally, we conclude the paper and discuss future works in Section 5.
2. Related Work

There are two basic role engineering approaches: top-down and bottom-up. While the top-down approach defines roles by examining the business processes, the bottom-up approach have been proposed to use data mining techniques to build RBAC system.

Coyne [16] firstly defined the role engineering problem and proposed the concepts of the top-down approach. Kuhlmann et al. [17] proposed the concepts of role mining and how to use data mining techniques for finding roles from user-permission assignments. The ORCA algorithm proposed by Schlegelmilch and Steffens [18] is a hierarchical clustering algorithm. However, it does not allow overlapping roles. RoleMiner[19] proposed by Vaidya et al. is a two-phase algorithm based on subset enumeration. Pair Count (PC) algorithm proposed by Molloy et al. [5] is based on a new idea for prioritizing roles. HPr algorithm [8] was proposed by a group of researchers from HP Labs, it aims to find a minimal set of roles. HierarchicalMiner(HM) algorithm [10] was proposed by Molloy et al. This approach is based on formal concept analysis and the semantics of roles.

Clearly, the main drawback of the above role mining algorithms is that they cannot deal with constraints. Therefore, they discover roles when the available information is limited to the user-permission relation. A close related work was proposed by Kumar et al.[20], their algorithm guarantees that no role contains more than a given number of permissions in the discovered configurations. The main drawback of this algorithm is that it only deals with one kind of the cardinality constraints by tradition imperative programming language when there are four kinds of cardinality constraints in all. This algorithm can not deal with more constraints at the same time.

Lu et al. [21] define the role mining problem with negative authorizations and proposed an approach to discover underlying constraints from the extended Boolean matrix decomposition. An assumption is proposed that the user-permission assignments imply the information of constraints. However, a lot of the constraints may not be embodied by the user-permission assignments since constraints usually describe high-level security and business requirements. In contrast, our work can deal with all of the constraints whether are embodied by the user-permission assignments or not.

Hu et al. [22] propose an ASP-Based approach to constraint-enhanced role engineering. Role engineering has a two-phase process. The first stage is role mining. The goal of this stage is to construct the RBAC system
from a non-RBAC system. The second stage is role updating. It is a post-maintenance for the existing RBAC system. Their work is just for the second stage. Their method tweaks the existing RBAC system in order to satisfy the constraints via answer set programming. The optimization goal of their method is to minimize the change between the initial RBAC system and result RBAC state. Our work realizes the synchronization of constructing RBAC system from scratch and meeting the constraints.

3. Preliminaries

In this section, we will review the main concepts of ASP and the notions of role mining problem and constraint.

3.1. ASP Preliminaries

ASP is an approach to declarative problem solving. Rather than solving a problem by telling a computer how to solve the problem, the idea is simply to describe what the problem is and leave its solution to the computer. By comparison with other approaches such as SAT (Satisfiability Checking) and CP (Constraint Programming), ASP is an expressive nonmonotonic language based on stable model semantics, which allows elegant knowledge representation such as causality, defaults, and incomplete information. Then, we review the main concepts of ASP. More details can reference [23].

An answer set program is a finite set of rules of the form

\[ h \leftarrow b_1, \ldots, b_m, \neg b_{m+1}, \ldots, \neg b_n \]  

where \(0 \leq m \leq n\), \(h\) and \(b_i\) are atoms, and \(\neg\) denotes default negation. In addition, \(h\) is called the head of the rule and \(\{b_1, \ldots, b_m, \neg b_{m+1}, \ldots, \neg b_n\}\) is called the body of the rule. When the rule body is empty, the rule is called a fact.

The ground logic program \(\Pi\) is denoted as \(P(\Pi)\), which is obtained by all possible substitution of elements of the Herbrand universe for the variables. Let \(M\) be a subset of the Herbrand base of \(\Pi\). We say that \(M\) is called an answer set of a program \(\Pi\) if \(M\) is the minimal set of the program \(P(\Pi)^M\), which is obtained from \(P(\Pi)\) by

- removing all rules having a negative literal \(\neg b_i\) in its body where \(b_i \in M\) and \(i \in [m + 1, n]\),
- and then eliminating \(\neg b_i\) in the bodies of the remaining rules.
3.2. Role Mining Problem

In this paper, we will review the basic definitions in RBAC and role mining problems which are the foundation of our work.

**Definition 1 (RBAC Model).** The RBAC model has the following components:

- $U, R, P$, users, roles and permissions respectively;
- $PA \subseteq P \times R$, a many-to-many mapping of permission-to-role assignments;
- $UA \subseteq U \times R$, a many-to-many user to role assignment relationships;

We can use an $m \times n$ binary matrix $UPA$ to define the user-permission assignment relationship. Where $m$ is the number of users and $n$ is the number of permissions. The element $UPA_{ij} = 1$ means that the $i^{th}$ user has the $j^{th}$ permission.

**Definition 2 (Boolean Matrix Multiplication).** A Boolean matrix multiplication between an $n \times k$ Boolean matrix $A$ with $a_{ij} \in \{0, 1\}$ and a $k \times m$ matrix $B$ with $b_{lj} \in \{0, 1\}$ is $A \otimes B = C$, where $C$ is matrix with $c_{ij} \in \{0, 1\}$ and

$$c_{ij} = \bigvee_{l=1}^{k} (a_{il} \land b_{lj}).$$

**Definition 3 (User Permission Assignment Matrix Decomposition).** Consider an $n \times m$ binary matrix $UPA$, which presents the user-permission assignment. Assume Binary matrices $UA$ and $PA$ with dimensions $n \times k$ and $k \times m$ respectively, where $UA$ presents a user-role assignment and $PA$ presents a role-permission assignment. $UA \otimes PA$ is called a decomposition of $UPA$ if

$$(UPA)_{ij} = \bigvee_{l=1}^{k} ((UA)_{il} \land (PA)_{lj}).$$

**Definition 4 (Role Mining Problem).** Given an access control configuration $\rho = \langle U, P, UPA \rangle$, while $U$ is a set of all users, $P$ is a set of all permissions, $R$ is a set of roles, $UPA \subseteq U \times P$ is the user-permission relation, $RH \subseteq R \times R$ is the relationship of role hierarchy, and $DUPA \subseteq U \times P$ is the direct user-permission assignment relation. Role mining problem aims to find an RBAC state $\langle R, UA, PA, RH, DUPA \rangle$ that is consistent with $\rho$. 


The basic role mining problem is an important special case of the role mining problem. It can be defined as follows.

**Definition 5 (Basic Role Mining Problem).** Given the user-permission matrix, the basic Role Mining problem aims to find a user-to-role assignment $UA$ and a role-to-permission assignment $PA$, $UA \otimes PA = UPA$ and minimizing the number of roles.

The role mining with constraints problem is the variant of the role mining problem. It can be defined as follows.

**Definition 6 (Role Mining With Constraints Problem).** Given an access control configuration $\rho = \langle U, P, UPA \rangle$ and $C$ is a set of constraints. where $U$ is a set of all users, $P$ is a set of all permissions, $UPA \subseteq U \times P$ is the user-permission relation and $DUPA \subseteq U \times P$ is the direct user-permission assignment relation. Role mining with constraints problem aims to find an RBAC state $\langle R, UA, PA, RH, DUPA \rangle$ that is consistent with $\rho$ and is satisfied with $C$ at the same time.

In a set of constraints $C = \{c_1, c_2 \ldots c_n\}$, for example, $c_1$ means that the number of permissions owned by role is no more than three. $c_2$ means that the number of users which belong to a role is no more than three.

Currently, the weighted structural complexity is the most important measure in role mining. The weighted structural complexity (WSC) is defined as follows.

**Definition 7 (Weighted Structural Complexity).** Given $W = (w_r, w_u, w_p, w_h, w_d)$ where $w_r, w_u, w_p, w_h, w_d \in [0, +\infty]$, the weighted structural complexity of an RBAC state $\gamma$, which is denoted as $WSC(\gamma, W)$, is computed as

$$WSC(\gamma, W) = w_r \cdot |R| + w_u \cdot |UA| + w_p \cdot |PA| + w_h \cdot |tr(RH)| + w_d \cdot |DUPA|$$

where $|\cdot|$ denotes the size of the set or relation, and $tr(RH)$ denotes the transitive reduction of role-hierarchy.

A transitive reduction is the minimal set of relationships that describes the same hierarchy. For example, $tr(\{(r1, r2), (r2, r3), (r1, r3)\}) = \{(r1, r2), (r2, r3)\}$, as $(r1, r3)$ can be inferred.
Essentially, the weighted structural complexity is a multi-objective optimization function. In role mining, the goal is to minimize the weighted structural complexity. We can adjust the weights of WSC to meet different optimization objectives. For example, by setting $w_r = 1$, $w_u = w_p = 0$, and $w_h = w_d = \infty$, we aim at minimizing the number of roles. In this paper, we do not allow direct user permission assignments by setting $w_d = \infty$. Because the direct user permission assignments defeats the purpose of RBAC.

3.3. Constraint Specification

In this section, we introduce constraints in role-based access control. We consider three types of role-based constraints. They are cardinality, prerequisite, and mutual exclusion constraints. These constraints have been considered in existing literature [12][24].

**Cardinality constraints.** There are four kinds of cardinality constraints. The first one is the role-user cardinality constraint. It is satisfied if and only if the role $r$ is assigned to at least $l$ users and no more than $u$ users, where $l, u \in [0, \infty)(l < u)$ are called the lower bound and the upper bound respectively.

In practice, we require a role be assigned to at least a certain number of users in order to meet workload. In addition, the upper bound in a constraint makes sure that the role is not assigned to too many users because of resource restrictions. In contrast, the role-permission cardinality constraint is satisfied if and only if the number of permissions owned by role is at least $l$ and no more than $u$.

Similarly, two other kinds of cardinality constraints are the cardinality constraint of user and permission. The cardinality constraint of user means that the number of roles to which a user can be assigned is at least $l$ and no more than $u$; the cardinality constraint of permission means that number of roles to which a permission can be assigned is at least $l$ and no more than $u$.

**Prerequisite constraints.** A prerequisite constraint is represented as $PRE(\text{cond})$, where $\text{cond}$ is called the prerequisite condition. Prerequisite conditions are defined based on competency and appropriateness whereby a user is the member of role $R_1$ only if the user is already the member of role $R_2$, or a permission $P_1$ can be assigned to a role only if the role has possessed permission $P_2$.

**Mutual exclusion constraints.** The first kind of mutual exclusion constraint can be represented as $MEC(R, m)$, where $R$ is a set of roles and
$m \in [2, |R|]$ is an integer. A mutual exclusion constraint $MEC(R, m)$ is satisfied if and only if no user is assigned to $m$ or more roles in $R$.

Similarly, another kind of mutual exclusion constraint can be represented as $MEC(P, m)$, where $P$ is a set of permissions and $m \in [2, |P|]$ is an integer. A mutual exclusion constraint $MEC(P, m)$ is satisfied if and only if no role owns $m$ or more permissions in $P$.

4. Constrained Role Mining

4.1. Computational Complexity

The computational complexity of the Role Mining Problem (and of some of its variants) was considered in several papers. In this section we define the decisional version of the Role Mining With Constraints Problem and we show that it is NP-hard. Next we recall the decisional version of the Role Mining Problem.

**Definition 8 (Role Mining Decision Problem).** Given a set of users $U$, a set of permissions $P$, a user-permission assignment $UPA$, and a positive integer $k < \min\{|U|, |P|\}$, are there a set of roles $R$, a user-to-role assignment $UA$, and a role-to-permission assignment $PA$ such that $|R| \leq k$ and $UPA = UA \otimes PA$.

Role Mining Problem is proved NP-complete by Vaidya et al [6]. This has been proved by reducing it to the Set Basis Decision Problem.

The decisional version of the Role Mining With Constraints Problem can be defined, as follows:

**Definition 9 (Role Mining With Constraints Decision Problem).** Given a set of users $U$, a set of permissions $P$, a user-permission assignment $UPA$, a constraint set $C$, and a positive integer $k < \min\{|U|, |P|\}$, are there a set of roles $R$, a user-to-role assignment $UA$, and a role-to-permission assignment $PA$ such that $|R| \leq k$, $UPA = UA \otimes PA$, while this configuration satisfies $C$.

**Theorem 1.** Role Mining With Constraints Problem is NP-hard.

Proof: Given an instance of the Role Mining decision Problem, here is how we transform it to an instance of the Role Mining with Constraints decision Problem: In the Role Mining decision Problem, $C$ is a set of constraints
and $C$ is an empty set. Now, the answer to the decision role mining with constraints problem directly provides the answer to the decision role mining problem. The transformation is clearly polynomial. Thus, the theorem holds.

4.2. Knowledge Modeling And Representation

In this section, the issues on knowledge modeling and representation for constrained Role Mining are discussed. In the role mining research, the goal is to discover roles from existing user-permission assignments. Given an $m \times n$ binary matrix $UPA$ representing the user-permission assignment, it could be seen as a slightly damaged chess board ($m \times n$). If the $U_i$ gets the $P_j$, then $(UPA)_{ij} = 1$ and the $(i, j)^{th}$ cell of the chess board is regarded as good square. Similarly, If the $U_i$ do not have the $P_j$, then $(UPA)_{ij} = 0$ and the $(i, j)^{th}$ cell of the chess board is regarded as bad square. Find a covering of the board using $k$ tiles so that all the good squares on the board are covered. Essentially, $k$ tiles can be equated to $k$ roles. If no such covering exists, report there is no solution.

In particular, you can swap any two rows or any two columns in matrix $UPA$ because they are symmetric. Therefore, it is allowed that good squares are covered using tiles after you swap any two rows or any two columns in the chess board.

In addition, we need to check for the following conditions:

- Bad cell needs not to be covered.
- All the good cells on the board must be covered.
- The shape of the tiles must only be square or rectangular.
- All of the constraints should be satisfied.

Next, let us explain these rules. As we all know, a tile represents a role, so if the $(i, j)^{th}$ cell of the chess board is covered by a tile, that means the user $U_i$ gain the permission $P_j$ through this role. if the $(i, j)^{th}$ bad cell is covered by a tile, this means that user $U_i$ get the permission $P_j$ which do not belong to him through roles. So we require that bad cell needs not be covered by tiles. In addition, in this paper, we do not allow direct user permission assignments because the direct user permission assignments defeats the purpose of RBAC. So we require that all the good cells on the board must be covered, which means all users should get permissions through roles. Finally, if a role has some permissions, some users get these permissions through this
Figure 1: Running example
role, intuitively, it is easy to see that the tiles which represent roles must be square or rectangular in the chess board.

Then, we will use the following running example to illustrate our approach. The original RBAC state is given in Figure 1(a). There are 10 users, 12 permissions, and 7 roles in the original state. The user-permission relation resulted from the state is given in Figure 1(b). The output of constrained role miner for the running example is given in Figure 1(c) and Figure 1(d).

In Figure 1(c) and Figure 1(d), we treat a lattice as a role. For instance, in Figure 1(c), \{(U_7, U_8, U_9), (P_3, P_6, P_7, P_8, P_0, P_10, P_11)\} signifies No.2 lattice which is dyed green. That means No.2 lattice represent a role which has a set of permissions \(P_3, P_6, P_7, P_8, P_0, P_10, P_11\) and the set of users \(U_7, U_8, U_9\) get these permissions through No.2 role.

In every lattice, each lattice which represents a role inherits all permissions associated with its subconcepts, and users are inherited in the other direction. Therefore, we can remove redundant permissions and users from each node. The result is called the reduced role hierarchy and is shown in Figure 1(d).

Figure 2 is a slightly damaged chess board \((10 \times 12)\) which is corresponding to the user-permission relation in Figure 1(b). If the \(U_i\) gets the \(P_j\), then \((UPA)_{ij} = 1\) and the \((i, j)\)th cell of the chess board is black. Similarly, if the \(U_i\) do not have the \(P_j\), then \((UPA)_{ij} = 0\) and the \((i, j)\)th cell of the chess
board is gray.

We can swap any two rows or any two columns in chess board because they are symmetric. For examples, We can swap P0 and P9 column in Figure 2, the result is Figure 3. In Figure 3, the cells surrounded by red thread is covered by a tile, this tile is corresponding to the red lattice in Figure 1(c). This role have a set of permissions \((P0, P10, P11)\), and the set of user \((U0 \sim U9)\) get these permissions through this role. Then we can continue to swap P3 and P5 column, the result is Figure 4. In Figure 4, the cells surrounded by red thread is covered by a tile, this tile is corresponding to the green lattice in Figure 1(d). This role have a set of permissions \((P3, P6, P7, P8, P0, P10, P11)\), and the set of user \((U7, U8, U9)\) get these permissions through this role.

By using ASP, we can use two predicates \(user()\) and \(per()\) to represent the dimension and size of the board.

\[
user(1..m). \quad // \text{m rows} \quad (1) \\
per(1..n). \quad // \text{n columns} \quad (2)
\]

We can also use predicate \(good(i, j)\) to indicate that the \((i, j)\)th cell of the board is good cell in ASP.

\[
ua(R,U) : \neg user(U), per(R), not n.ua(R,U). \quad (3) \\
n._ua(R,U) : \neg user(U), per(R), not ua(R,U). \quad (4)
\]
A logic program consists of facts (as Rules (1) and (2)) and rules (as Rules (3) and (4)), each of which is terminated by a period ‘.’. The connectives ‘:-’ and ‘,’ can be read as if and and, respectively.

Rule (3) defines $ua(R, U)$ to represent that user(U) is assigned to $role(R)$ while rule (4) defines $n\_ua(R, U)$ to represent that user(U) is not assigned to $role(R)$. Rule (5) defines $pa(R, P)$ to represent that $per(P)$ is assigned to $role(R)$ while rule (6) defines $n\_pa(R, P)$ to represent that $per(P)$ is not assigned to $role(R)$. Rules (7), (8) and (9) make sure that bad cell needs not be covered and all the good cells on the board must be covered. Rules (10)–(14)
define the roles in RBAC system.

Next, we define predicate \( h(R_1, R_2) \) to indicate that the relationship between \( R_1 \) and \( R_2 \) is direct inheritance. Then we define \( r_{ua}(R, U) \) to represent that \( user(U) \) belonging to \( role(R) \) and \( user(U) \) is not inherited from other roles. Similarly, we define \( r_{pa}(R, P) \) to represent that \( role(R) \) have permission \( per(P) \) and \( per(P) \) is not inherited from other roles. Moreover, each role should cover the good cells as much as possible.

Finally, we use optimization statement in ASP to minimize the WSC:

\[
\sharp\text{minimize}\{role(R), r_{ua}(R, U), r_{pa}(R, P), h(R_1, R_2)\} \quad (15)
\]

If only want to minimize the number of the roles, we can use optimization statement \( \sharp\text{minimize}\{role(R)\} \).

Because we do not need to implement a variety of specific conflict resolution algorithm with CRM, we are easy to add or maintain constraints as needs change. In practice, CRM can deal with all of the constraints related to RBAC. First, we take one mutually exclusive constraint for example. In the running example, we do not allow \( U_7 \) belongs to two different roles. The ASP program for this constraint is (16). The result of this constraint is in Figure 5(a).

\[
: -r_{ua}(R_1, U_7), r_{ua}(R_2, U_7), R_1! = R_2, role(R_1), role(R_2), u(U). \quad (16)
\]

Then, we can take cardinality constraint which is one of the important constraints required by RBAC model for examples.

There are four kinds of cardinality constraints. We can set the upper limit of the number as \( u \) and the lower limit as \( l \). The cardinality constraint can be represented in ASP as following:

The number of roles to which a permission can be assigned is limited.

\[
l\{r_{pa}\{R, P\} : role(R)\}u : -per(P). \quad (17)
\]

We set \( l=0 \) and \( u=2 \). That means the number of roles to which a permission can be assigned is not more than two in the running example. In the meanwhile, we request that constraints (17) and (18) should be satisfied at the same time. The result of the two constraints is in Figure 5(b). Constraint (18) is a simple example for prerequisite (PRE) constraints which means that \( U_0, U_1 \) and \( U_2 \) should belong to \( R_4 \).

\[
Ua(4, 0), Ua(4, 1), Ua(4, 2). \quad (18)
\]
Figure 5: Constraints in running example
The number of permissions owned by role is limited.

\[ l\{r\_pa\{R, P\} : \text{per}(P)\}u : -\text{role}(R). \tag{19} \]

We set \( l=0 \) and \( u=3 \). That means the number of permissions owned by role is not more than three in the running example. The result of this constraint (19) is in Figure 5(c).

The number of users which belong to a role is limited.

\[ l\{r\_ua\{R, U\} : \text{user}(U)\}u : -\text{role}(R). \tag{20} \]

We set \( l=0 \) and \( u=3 \). That means the number of users which belong to a role is not more than three in the running example. The result of this constraint (20) is in Figure 5(d).

The number of roles to which a user can be assigned is limited. The ASP program for this constraint is (21).

\[ l\{r\_ua\{R, U\} : \text{role}(R)\}u : -\text{user}(U). \tag{21} \]

5. Evaluation Results

In this section, we evaluate the effectiveness of role mining using ASP. To study the performance of role mining using ASP, we implement CRM and run it on three datasets, including university, healthcare and Domino. The university datasets was used to evaluate HierarchicalMiner (HM) algorithm by Molloy et al [10]. The healthcare data was from the US Veteran’s Administration; the domino data was from a Lotus Domino server. Meanwhile, we use eight important role mining algorithms to compare with CRM. Datasets that have been used in our paper are shown in Table 1. Experiments were performed on a Windows 7 computer with Intel Core i7 processor and 16GB RAM. ASP programs were executed with Clingo [23].

Firstly, we want to evaluate the basic role mining problem in case of no constraints, which do not allow direct user permission assignments and just minimize the number of roles. For this problem, we use the scheme \( W1 : w_r = 1, w_u = w_p = w_h = 0 \) and \( w_d = \infty \). Table 2 shows the minimum number of roles discovered by each role mining algorithm. We can see that CRM has fewer number of roles than other role mining approaches on average.

In Table 3, we present the results from evaluating the nine algorithms with three datasets in case of no constraints. We use the scheme \( W2 : \)
Table 1: Sizes of the real-world datasets

| Dataset      | |User| |Perm| |UPA| |Density |
|--------------|---|---|---|---|---|---|---|
| Healthcare   | 46 | 46 | 1468 | 0.702 |
| University   | 493 | 56 | 3955 | 0.143 |
| Domino       | 79 | 231 | 730 | 0.04 |

Table 2: Minimal number of roles for \( W = \langle 1, 0, 0, 0, \infty \rangle \)

<table>
<thead>
<tr>
<th></th>
<th>CRM</th>
<th>HM</th>
<th>GO</th>
<th>CM</th>
<th>PC</th>
<th>DM</th>
<th>HPr</th>
<th>HPe</th>
<th>ORCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Healthcare</td>
<td>13</td>
<td>17</td>
<td>16</td>
<td>31</td>
<td>24</td>
<td>27</td>
<td>14</td>
<td>15</td>
<td>46</td>
</tr>
<tr>
<td>University</td>
<td>17</td>
<td>21</td>
<td>18</td>
<td>32</td>
<td>31</td>
<td>42</td>
<td>17</td>
<td>18</td>
<td>56</td>
</tr>
<tr>
<td>Domino</td>
<td>20</td>
<td>31</td>
<td>20</td>
<td>62</td>
<td>64</td>
<td>31</td>
<td>20</td>
<td>27</td>
<td>231</td>
</tr>
<tr>
<td>Average</td>
<td>16.7</td>
<td>23</td>
<td>18</td>
<td>41.7</td>
<td>43</td>
<td>33.3</td>
<td>17</td>
<td>20</td>
<td>111</td>
</tr>
</tbody>
</table>

Table 3: Minimal WSC for \( W = \langle 1, 1, 1, \infty \rangle \)

<table>
<thead>
<tr>
<th></th>
<th>CRM</th>
<th>HM</th>
<th>GO</th>
<th>CM</th>
<th>PC</th>
<th>DM</th>
<th>HPr</th>
<th>HPe</th>
<th>ORCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Healthcare</td>
<td>138</td>
<td>146</td>
<td>142</td>
<td>197</td>
<td>192</td>
<td>325</td>
<td>305</td>
<td>202</td>
<td>223</td>
</tr>
<tr>
<td>University</td>
<td>605</td>
<td>605</td>
<td>615</td>
<td>624</td>
<td>622</td>
<td>683</td>
<td>906</td>
<td>651</td>
<td>1828</td>
</tr>
<tr>
<td>Domino</td>
<td>417</td>
<td>423</td>
<td>476</td>
<td>532</td>
<td>583</td>
<td>615</td>
<td>762</td>
<td>693</td>
<td>756</td>
</tr>
<tr>
<td>Average</td>
<td>387</td>
<td>391</td>
<td>411</td>
<td>451</td>
<td>466</td>
<td>541</td>
<td>658</td>
<td>515</td>
<td>936</td>
</tr>
</tbody>
</table>

Table 4: CRM role mining results

<table>
<thead>
<tr>
<th></th>
<th>No Constraints</th>
<th>With Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WSC</td>
<td>Time</td>
</tr>
<tr>
<td>Healthcare</td>
<td>138</td>
<td>5 mins</td>
</tr>
<tr>
<td>University</td>
<td>605</td>
<td>6 hours</td>
</tr>
<tr>
<td>Domino</td>
<td>417</td>
<td>4 hours</td>
</tr>
</tbody>
</table>

\( w_r = w_u = w_p = w_h = 1 \) and \( w_d = \infty \). The scheme W2 assumes that the cost of adding each element (a role or a relationship) to the RBAC state is 1 while direct user permission assignments are not allowed. The weighted structural complexity thus measures the cost to create the RBAC state. We can see that CRM has smaller WSC than other algorithms on average.

All the existing role mining algorithms are approximation algorithms because Role Mining Problem is NP-complete problem.
ming is designed to find the global optimal solution. Hence, role mining using ASP can find the optimal solution on the premise that it has enough time. For a limited time, role mining using ASP could get the best possible solution. This is the reason why CRM usually has smaller WSC than other algorithms on average.

In Table 4, we show the execution time of role mining using ASP with the scheme W2. The column of ‘No Constraints’ shows the WSC values of CRM without constraints within a pre-set limited time. Moreover, we evaluate CRM on three datasets with three arbitrary constraints as follows:

- The number of roles to which a permission can be assigned is at least one and no more than six.
- The user U0 should belong to the role R2.
- We do not allow U0 belongs to two different roles.

Obviously, the first constraint is a cardinality constraint, the second constraint is a prerequisite constraint and the third one is a mutual exclusive constraint. The column of ‘With Constraints’ shows the WSC values of CRM with three constraints within a pre-set limited time. We could see that role mining using ASP has high cost of time. Although mature parallel ASP solver is still under development, parallel processing could still be a powerful way to reduce time cost in role mining using ASP in the future.

6. Conclusions

While there are many role mining approaches have been proposed recently, none of existing role mining algorithms can simultaneously satisfy various constraints, which usually describe organizations’ security and business requirements. To strengthen the ability of role mining technology, this paper proposes a novel role mining approach using answer set programming that can comply with various constraints and meet various optimization objectives, namely constrained role miner (CRM). To study the performance of role mining using ASP, we implement CRM and run it on three datasets, including university, healthcare and Domino. Meanwhile, we use eight important role mining algorithms to compare with CRM. According to experimental evaluation, CRM is also better than the existing role mining approaches in case of no constraints. Additionally, high degree of ASP skill significantly
affects the performance of the CRM. We are optimizing the CRM to enhance efficiency with great initiative.

7. Acknowledgments

This work is supported by National Natural Science Foundation of China under grants 61173170, 61300222 and 61433006, National High Technology Research and Development Program of China under grant 2007AA01Z403, and Innovation Fund of Huazhong University of Science and Technology under grants 2013QN120, 2012TS052 and 2012TS053.

References


