International Journal of Innovative
Computing, Information and Control
Volume **8**, Number **11**, November **2012**

ICIC International ©2012 ISSN 1349-4198
pp. **7909–7923**

MINING ROLES USING ATTRIBUTES OF PERMISSIONS

Ruixuan Li, Wei Wang, Xiaopu Ma, Xiwu Gu* and Kunmei Wen

School of Computer Science and Technology
Huazhong University of Science and Technology
No. 1037, Luoyu Road, Hongshan Dist., Wuhan 430074, P. R. China
{ rxli; kmwen }@hust.edu.cn; { wangwei_pl; xpma }@smail.hust.edu.cn
*Corresponding author: guxiwu@hust.edu.cn

Received August 2011; revised January 2012

Abstract. *Recently, many approaches were proposed to generate roles using automatic techniques. However, most of these approaches generate many composite roles because they only optimize minimality of the state in role-based access control (RBAC). The responsibility of the composite roles is complex and hardly interpretable, which weakens the robustness of the RBAC state. In this paper, we propose to use operations and resources of permissions as the functional information in role mining algorithm and present a novel approach, role mining with functional features (FMiner), to reduce composite roles. The FMiner approach is a two-phase solution. Firstly, an initial RBAC state is built by formal concept analysis theory. Secondly, relative closeness is defined to measure the functional similarity between roles. We optimize the relative closeness and minimality of the initial RBAC state simultaneously. The experimental results demonstrate the effectiveness of the proposed approach on reducing composite roles.*

Keywords: Role-based access control, Role engineering, Role mining, Operation, Resource

1. **Introduction.** In recent years, role-based access control (RBAC) [1, 2] has successfully been adopted by a variety of enterprise security management products. In an RBAC system, users acquire permissions through roles instead of directly assigning permissions to users. Although RBAC will simplify the security management works, how to build an optimal RBAC state is a challenge work.

To solve this problem, role engineering [3, 4] is introduced to configure an RBAC system, i.e., creating roles, assigning permissions to roles and users to roles. There are two basic approaches towards role engineering: the *top-down* and the *bottom-up*. Under *top-down* approach, roles are generated by analyzing business processes and then assigning the needed permissions to create roles for these business functions [5, 6]. Though this approach can reflect the organization function well, it is time-consuming and costly because of dozens of business processes and tens of thousands of users in an organization. Under *bottom-up* approach, roles are discovered from existing user-permission assignments by data mining techniques. It is easier to develop a tool for building an RBAC state automatically or semi-automatically through the bottom-up approaches.

In role mining field, the minimality is the commonly criterion for generating roles. However, if we only optimize the minimality of the RBAC state, the function of roles would be weakened because minimality and function are often contradictory. The minimality includes a number of roles, a number of assignments, role hierarchies, etc. The less the number of roles is, the more the permissions of every role are assigned. A role with many permissions is hard to be assigned when employees fluctuate and it is uninterpretable for complex functions and unlikely to correspond with business processes. Therefore, a role

with many permissions is often a composite role which is the collection of some real roles. On the other hand, the less number of assignments will also make composite roles because composite roles are in favor of reducing the number of users to role assignments. If the inheritance relationships between roles are reduced, the less number of role hierarchies can be got. However, the permissions of senior roles will be assigned to junior roles, which will increase the opportunity to generate composite roles. Consequently, it is difficult to generate optimal roles by taking minimality as the only criteria. Some functional information should be considered to improve the interpretability and generalization ability of the generated roles [4].

In order to generate roles with semantic meaning, Frank et al. [7, 8] and Molloy et al. [21] propose to use the attributes of users together with basic user-permission assignments to improve the interpretability of roles. However, few researches employ the attributes of permissions. Ma et al. [14] view the permissions with different weights since the permissions cannot be treated evenly. Therefore, [14] uses association rule mining technology to discover frequent patterns. These frequent patterns are described by item sets and can be viewed as roles. In addition to weights, operations and resources are also important attributes of the permissions. A permission means that a user has the access to operate a resource. In an RBAC system, a role is usually generated to agglomerate the permissions that can complete a task or process. For example, the role "Undergrad" in a university contains two permissions: "Register_UndergradClass" and "Withdraw_UndergradClass". An undergraduate achieves the necessary operations to access resource "UndergradClass" by the role "Undergrad". "Register_UndergradClass" and "Withdraw_UndergradClass" have the same resource "UndergradClass". The role "Patient" in a hospital is another case with almost the same operations. "Patient" contains permissions "View_OldMedicalRecords", "View_RecentMedicalRecords", "View_Medical RecordsWithThirdPartyInfo", "Sign_LegalAgreement", "View_Prescriptions" and "View _Bills". Intuitively, operations and resources of the permissions can be used as the functional information to discover meaningful roles.

Together with considering the operation and resource attributes of permissions, the number of the composite roles will be reduced. If a role in a business process with a set of strong related resources or a set of strong related operations, we view it as a single responsibility. In object-oriented programming, the single responsibility principle states that every object should have a single functionality, and that functionality should be entirely encapsulated by the class. We can add or remove users without modifying the RBAC configuration frequently through the roles with single responsibility. In extreme cases, every role with only one permission may be more adaptive to the fluctuations of users.

Based on the above observation, we propose a novel approach for role mining that considers the user-permission assignments together with the attributes of permissions. This approach is a two-phase solution. In the first phase, we build an initial RBAC state by formal concept analysis. Every permission only belongs to one role in this initial RBAC state. In the second phase, we prune the initial RBAC state based on weighted structural complexity (WSC) and relative closeness (RC). WSC is a widely used criterion that measures the minimality of RBAC states. In the pruning process of HierarchicalMiner [21] and StateMiner [13], the permission set of a role r is added to all of its immediate senior roles and the user set of r is added to all its immediate juniors. This pruning strategy will cause the following problems:

- When we add the permission set of r to the permission set of its immediate seniors, the added permission set may weaken the functional features of immediate seniors, which makes the responsibilities of immediate seniors unclear.
- Some roles that have immediate juniors but are not associated with permissions may be viewed as composite roles. If the relationship of the immediate juniors of a composite role is weak, the function of this composite role is also hard to interpret because it does not meet the principle of single responsibility.

These problems are the main reasons of creating composite roles. Our approach uses relative closeness to overcome these problems. Relative closeness is the measure of functional feature similarity between roles. The functional features include assignment cohesion degree, operation centrality and resource centrality of roles. Usually, the proportion of the number of common users to all users associated with two permissions is used to measure the similarity between these two permissions. The user similarity can be calculated by the similar method. If the users of a role have the same permission assignments and the permissions have the same user assignments, we consider the cohesion degree of the role should be high. If a role refers to an operation or related operations, we view the role's operation centrality is high. Similarly, the resource centrality is measured by the same method. These functional features will help to reduce composite roles effectively.

The remainder of the paper is organized as follows. We discuss the related work in Section 2 and the related definition in Section 3. The limitations of existing applications for role mining drive our motivation and Section 4 proposes the method to find the relative closeness between roles. Furthermore, we provide a novel role mining algorithm, *FMiner*, based on relative closeness and WSC to reduce composite roles. A summary of our experimental results on real dataset is discussed in Section 5. Finally, Section 6 provides some insight into our ongoing and future work.

2. **Related Work.** The role mining algorithms can be divided into two classes according to their outputs [5]. The first class algorithms generate a set of candidate roles and then give every role a priority value. The representative algorithms of this class are $CompleteMiner$ (CM) and $FastMiner$ (FM) [17]. The CM and FM will generate a large number of roles and then use $Origcount(r) \times priority + Count(r)$ to prioritize the candidate roles (where $Origcount(r)$ denotes the users have exactly the permissions in r, $priority$ is a tunable parameter to favor initial roles, and $Count(r)$ is the number of users whose permissions are a superset of r). The second class algorithms use WSC as the common quality measurement to generate a complete RBAC state. *ORCA* [12], $HierarchicalMiner$ (HM) [21], and *GO* [10] are such kind of algorithms. *ORCA* is a hierarchical clustering algorithm where every permission is exactly assigned to only one role. *HM* restructures the initial RBAC state based on the cost decreasing using a greedy strategy. *GO* reduces the number of role-user assignments and permission-role assignments by graph optimization method.

Some researchers add business information into the role mining approaches. Their algorithms are called hybrid role mining approaches. Frank et al. [8] provide a probabilistic model to analyze the relevance of different kinds of business information for defining roles that can explain the given user-permission assignments and describe the meaning from the business perspective. Molloy et al. [21] propose attribute miner to generate roles by attribute information of users.

However, none of the work introduces operations and resources of permissions to improve the effectiveness of the algorithms. Operations and resources contain functional information that can discover the potential reasons for creating roles. In our algorithm, the functional features are extracted from operations and resources of roles, and then the

relative closeness between roles is computed based on these features. The relative closeness is borrowed from CHAMELEON [19] algorithm in data mining field. CHAMELEON is a two-phase clustering algorithm that can discover natural and homogeneous clusters on data sets with different shapes, densities and sizes. The two-phase strategy is a common strategy used in many mining algorithms, such as [13, 21].

We use reduced concept lattices that are created by user-permission assignments to represent the initial RBAC state. The pruning process of the initial RBAC state is guided by relative closeness between roles and WSC of the RBAC state. The experimental results show the effectiveness of reducing composite roles.

3. **Problem Statement and Preliminaries.** In this paper, we follow the basic definitions in NIST standard [1], which is the most widely known as formal description of RBAC model.

Definition 3.1. *The RBAC model contains the following components:*

- $USERS$, $PERMS$, $ROLES$, *the set of users, permissions and roles respectively;*
- $UA \subseteq USERS \times ROLES$, *many-to-many user to role assignment relationships;*
- $PA \subseteq ROLES \times PERMS$, *many-to-many role to permission assignment relationships;*
- $UPA \subseteq USERS \times PERMS$, *many-to-many user to permission assignment relationships*

where $M = |USERS|$, $N = |PERMS|$ *and* $K = |ROLES|$. *The users, permissions and roles can be ordered.* u_i $(i = 1, \ldots, M)$ *indicates the* i*th user,* p_j $(j = 1, \ldots, N)$ *indicates the* j*th permission, and* r_k $(k = 1, \ldots, K)$ *indicates the* k*th role. If the* i*th user has the* j*th permission,* $UPA_{ij} = 1$*; otherwise,* $UPA_{ij} = 0$. *Similarly, the matrix* UA *denotes user to role relationships and* PA *denotes role to permission relationships.*

Definition 3.2. *The original similarity between the* i*th permission and the* j*th permission is defined as*

$$\mathrm{sim}(p_i, p_j) = \frac{|UPA_{\cdot i}^{\mathrm{T}} \cap UPA_{\cdot j}^{\mathrm{T}}|}{|UPA_{\cdot i}^{\mathrm{T}} \cup UPA_{\cdot j}^{\mathrm{T}}|}$$

Definition 3.3. *The original similarity between the* i*th user and the* j*th user is defined as*

$$\mathrm{sim}(u_i, u_j) = \frac{|UPA_{i\cdot} \cap UPA_{j\cdot}|}{|UPA_{i\cdot} \cup UPA_{j\cdot}|}$$

Definition 3.4. *The operation and resource attributes of permission are defined as:*

- OP, *the set of operations, such as "create", "add", "delete" and "read";*
- RS, *the set of resources, such as "accounts", "books" and "students";*
- $POA \subseteq PERMS \times OP$, *many-to-many relationships between permissions and operations;*
- $PRA \subseteq PERMS \times RS$, *many-to-many relationships between permissions and resources.*

Define $A = |OP|$ *and* $B = |RS|$, *where* A *denotes the size of* OP, *and* B *denotes the size of* RS. op_i $(i = 1, \ldots, A)$ *indicates the* i*th operation and* rs_j $(j = 1, \ldots, B)$ *indicates the* j*th resource.*

Figure 1 shows the relationships of roles, permissions, operations and resources. The target of roles is to manage the access control of operations on resources. A role is a collection of permissions and permissions can be split into operations and resources. Operations and resources carry the function and action information of permissions. Such a

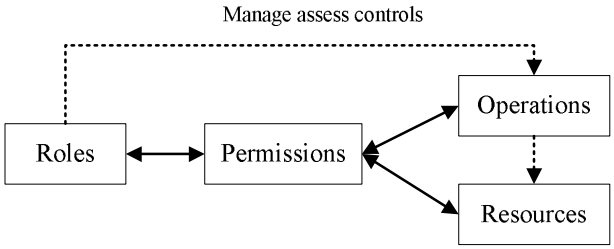


FIGURE 1. The relation of user-resources

split approach can provide good interpretability of a permission. However, the common approach just utilizes naming permissions for reducing the system's redundancy. For example, the permission to delete a book in a library management system can be represented as follows.

- $(Permission)$, DeleteBook
- $(Operation)$, Delete
- $(Resource)$, Book

Here operation and resource are just abstract definitions. The delete operation cannot directly be mapped to the actual method *book.delete*(). Two mappings need to be built. One is mapping permission to operation and the other is mapping permission to resource. Obviously, this representation contains more semantic information than the naming approach. However, the complexity of this approach is something high.

Definition 3.5. *The operation and resource centrality of a role r are defined respectively as follows.*

$$OpC(r_i) = -\sum_{j=1}^{n} \frac{|ROA_j|}{|PA_{i.}|} \log_2 \frac{|ROA_j|}{|PA_{i.}|}$$

$$ResC(r_i) = -\sum_{j=1}^{m} \frac{|RRA_j|}{|PA_{i.}|} \log_2 \frac{|RRA_j|}{|PA_{i.}|}$$

where n is the operation number and m is the resource number of role r_i respectively. $ROA_j = \{p_k | PA_{ik} \equiv 1$ and $POA_{kj} \equiv 1\}$ permissions authorized to role r_i, which has the operation op_j, $RRA_j = \{p_k | PA_{ik} \equiv 1$ and $PRA_{kj} \equiv 1\}$ is the number of permissions authorized to role r_i that has the resource rs_j. $OperCen$ and $ResCen$ are measured by the sum of entropy of operations and resources appeared in r_i respectively. For example, the role "Undergrad" $\{Register\_UndergradClass, Withdraw\_UndergradClass\}$ that contains the same resource "UndergradClass" and two operations "register" and "withdraw". Thus, $ResCen(Undergrad) = 0$ and $OperCen(Undergrad) = 1$.

However, if we use the proportion of 1 to the number of roles to indicate the operation centrality, we cannot distinguish $r_1 = \{op_1\_rs_1, op_1\_rs_2, op_1\_rs_3, op_1\_rs_4, op_2\_rs_1\}$ from $r_2 = \{op_1\_rs_1, op_1\_rs_2, op_1\_rs_3, op_2\_rs_4, op_2\_rs_1\}$, which have the same number of permissions and operations. Intuitively, r_1 has stronger operation association than r_2 because most of permissions have the same operation op_1 in r_1, but the operation distribution of r_2 is mean. Therefore, considering the entropy of operation and resource will be more accurate to describe the operation and resource centrality feature of r_1 and r_2.

Definition 3.6. *The internal permission cohesion is measured by*

$$IPC(r_i) = \frac{\sum_{j=1}^{|rp|} \min(\text{sim}(p_j, rp - p_j))}{|rp|}$$

where rp is the permission set associated with r_i. Analogously, internal user cohesion is measured by

$$IUC(r_i) = \frac{\sum_{j=1}^{|ru|} \min(\text{sim}(u_j, ru - u_j))}{|ru|}$$

where ru is the user set associated with r_i. Every permission in r_i has a similarity with other permissions. If we include all of them, there may be some permissions with very high similarity values that leads to a high overall internal permission cohesion. Therefore, we only consider the lowest similarity with other permissions for every permission.

In the pruning process of our algorithm, we want to merge two roles that their internal permission cohesion, internal user cohesion, operation centrality and resource centrality features are not changed too much between merging before and after. Thus, we firstly describe these features of a role r_i as a vector.

$$RF(r_i) = \langle IPC(r_i), IUC(r_i), OpC(r_i), ResC(r_i)\rangle.$$

Definition 3.7. *After define the features of a role r as a vector, we use the relative closeness between r_i and r_j to measure the feature change of r_i and r_j merging before and after as follows:*

$$RC(r_i, r_j) = \sum_{w_i \in w, RF_i \in RF} w_i \times \frac{RF_i(r_i + r_j) - 0.5 \times RF_i(r_i) - 0.5 \times RF_i(r_j)}{RF_i(r_i + r_j)}$$

where $w = \langle fu, fp, fo, fr\rangle, fp+fu+fo+fr = 1$ is the weight vector of features in a role. $RF = \langle IPC, IUC, OpC, ResC\rangle$ is the feature vector of role. $RF(r_i+r_j) - 0.5 \times RF(r_i) - 0.5 \times RF(r_j)$ indicates the information gain between view r_i and r_j as a composite role and view them as two roles. The information gain ratio of merging before and after is measured by $(RF(r_i + r_j) - 0.5 \times RF(r_i) - 0.5 \times RF(r_j))/RF(r_i + r_j)$. In these expressions, we suppose roles have the same weight because if we use $N_i/(N_i+N_j)$ or $N_j/(N_i+N_j)$ where N_i and N_j are the number of permissions in r_i and r_j respectively to indicate the weights that the information gain ratio turns to roles have more permissions. It is understood that the number of permissions in a role affects slightly to its creation. Therefore, the higher gain ratio is, the more dissimilar between roles merging before and after.

Definition 3.8. *Roles are formally described by concept lattices in our algorithm, thus we review the definition of concept lattice. A formal context is a triple $\kappa = (G, M, I)$, where G and M are sets of objects and attributes respectively, and $I \subseteq G \times M$ is an incident relation. gIm means that object g has the attribute m.*

In role mining field, we view the user-permission assignments as a formal context, where G is the set of all users and M is the set of all permissions. A concept of the context (G, M, I) is a pair (X, Y), where $X \in G$ and $Y \in M$ satisfy the following properties:

$$Y = \{m \in M | (\forall g \in X) gIm\},$$

i.e., Y is the set of all properties shared by all objects in X.

$$X = \{g \in G | (\forall m \in Y) gIm\},$$

i.e., X is the set of all objects that share all properties in Y. X is also called the extent and Y the intent of the concept (X, Y). The set of all concepts of the context is denoted by $B(G, M, I)$. A concept (X_1, Y_1) is a subconcept of (X_2, Y_2), denoted as $(X_1, Y_1) \leq (X_2, Y_2)$ if and only if $X_1 \subseteq X_2$ (or, equivalently, $Y_1 \supseteq Y_2$).

Definition 3.9. *For the given* $W = \langle wr, wu, wp, wh\rangle$ *where* $\langle wr, wu, wp, wh\rangle \in Q^+ \bigcup \{\infty\}$. *The weighted structural complexity* $wsc(\gamma, W)$ *of an RBAC state* γ *is denoted as follows:*

$$wsc(\gamma, W) = wr \times |R| + wu \times |UA| + wp \times |PA| + wh \times |t\_r(RH)|$$

where Q^+ *is the set of all non-negative rational numbers,* $|.|$ *indicates the size of the set or relation, and* $t\_r(RH)$ *indicates the transitive reduction of role-hierarchy. A transitive reduction is the minimal set of relationships that describes the same hierarchies of roles. For example,* $t\_r(\{(r_1, r_2), (r_2, r_3), (r_1, r_3)\}) = \{(r_1, r_2), (r_2, r_3)\}$ *for* (r_1, r_3) *can be inferred.*

4. **Algorithms.** In this section, we present a two-phase algorithm to find a set of roles with high cohesion on features and relatively simple responsibilities. In the first phase, we use reduced concept lattices to build initial RBAC state. In the second phase, we reconstruct this RBAC state based on relative closeness and WSC. Algorithm 4.1 gives the details of computing the relative closeness between roles. The internal permission cohesion and internal user cohesion of r_1 and r_2 are computed in Lines 1-18. Lines 19-32 give the method to get operation centrality and resource centrality of r_1 and r_2. The value of relative closeness between r_1 and r_2 is computed in Lines 33-37.

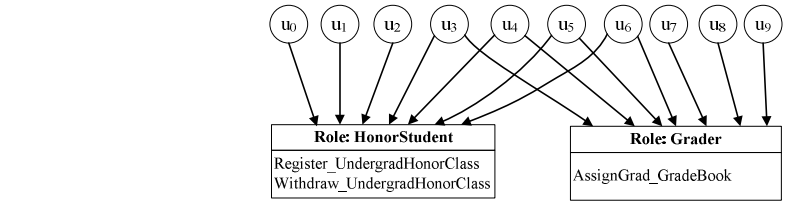
Algorithm 4.2 gives the details of reconstructing RBAC states in the second phase. There are three pruning cases need to be considered.

Case 1. A role r does not associate with a new user or a permission. If removing r can make value of WSC decrease, we remove it and update the role-hierarchy. r is solely used as a connection point of its immediate senior and immediate junior roles. Removing it will not affect the features of other roles.

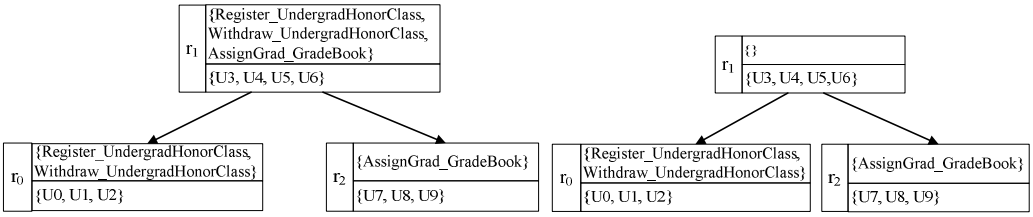
Case 2. A role r associates with some users but no permissions. If removing r can decrease value of WSC, we remove it and assign the users of r to its immediate junior roles. However, if removing r cannot make value of WSC decrease, *HierarchicalMiner* will retain r as a composite role and r inherits all permissions of its immediate juniors. In such case, *StateMiner* uses the global optimization function that includes WSC and global dissimilarity of the RABC state to determine whether retain r or not. In *HierarchicalMiner* and *StateMiner*, all users of r will be added to its juniors. In *FMiner*, if adding ru to r_j that an immediate junior of r can make $RC(\sum jun(r) - r_j, r_j) > \tau$, we will add ru to r_j. Otherwise, the hierarchy relation between r and r_j will be retained. Thus, r is removed when $\forall r_j \in jun(r)$, $RC\left(\sum jun(r) - r_j, r_j\right) > \tau$, where τ is the threshold of relative closeness.

For example, Figure 2(a) shows an example that $\{u_0, u_1, u_2\}$ is assigned to the role "HonorStudent", $\{u_7, u_8, u_9\}$ is assigned the role "Grader", and $\{u_4, u_5, u_6, u_7\}$ is assigned to both of these two roles. Figure 2(b) shows the concept lattices of this example and Figure 2(c) shows the reduced concept lattices. On the setting of $W = \langle 1, 1, 1, 1\rangle$, the WSC of current RBAC state is 18. If adding $\{u_3, u_4, u_5, u_6\}$ to $\{\{u_0, u_1, u_2\}$, {Register_GradClass, Withdraw_GradClass}} and $\{\{u_7, u_8, u_9\}$, {AssignGrad_GradeBook}}, the WSC of pruned RBAC state is 19. Thus, *HierarchicalMiner* and *StateMiner* will retain $\{\{u_3, u_4, u_5, u_6\}, \{\}\}$ that is shown in Figure 2 of [21] and Figure 2 of [13] as a composite role. Intuitively, r_0 with the same resource has high cohesion and r_1 is added the new resource "GradBook" that weakens this internal feature of r_0. Therefore, the composite role r_1 should not be created.

Currently, there are a lot of works discussing user similarity and permission similarity based on assignments. *FMiner* focuses on the affection of operation and resource on roles in this example. Thus, we set $fp = 0$, $fu = 0$, $fo = 0.2$ and $fr = 0.8$ because the resources has more influence on the weight of permissions. The relative closeness of r_0

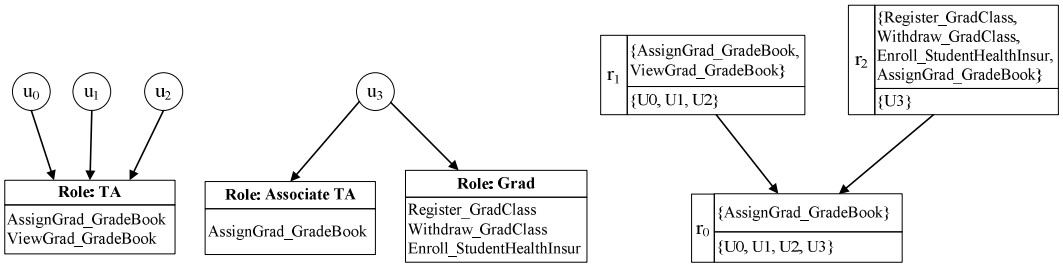


(a) The original RBAC state

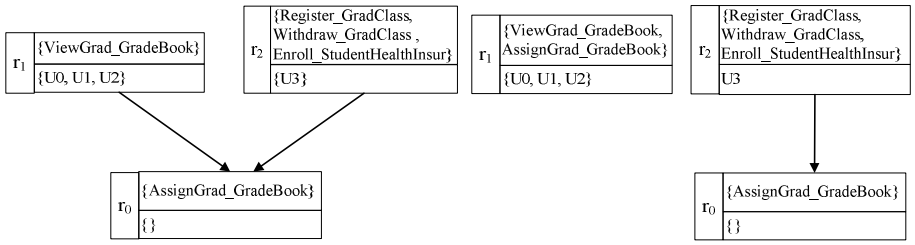


(b) The concept lattice (c) The reduced concept lattice

FIGURE 2. The example for Case 2



(a) The original RBAC state (b) The concept lattice



(c) The reduced concept lattice (d) The final RBAC state

FIGURE 3. The example for Case 3

and r_2 shown in Figure 2(c) will be $RC(r_0, r_2) = 0.92$. Though adding $\{u_3, u_4, u_5, u_6\}$ to its juniors will increase the value of WSC, the high relative closeness between r_0 and r_2 will prevent creating composite role r_1.

Case 3. A role r associates with no user but some permissions. In *HierarchicalMiner*, r will be removed and the permissions in r will be assigned to its immediate senior roles when this action can make WSC decrease. However, in *FMiner*, rp will be added to r_j when $RC(r, r_j) < \tau$ where r_j is an immediate senior role of r. Otherwise, hierarchy relation between r and r_j will be retained.

For example, Figure 3(a) shows an example that the role "TA" is assigned to $\{u_0, u_1, u_2\}$, and the role "Associate TA" and "Grad" are assigned to $\{u_3\}$. "Associate TA" helps "TA" assign grade books. However, there is some information in grade book cannot be known by others except the receivers of grade book or "TA". Thus, the permission "ViewGrad_GradeBook" is not assigned to "Associate TA". The WSC of Figure 3(c) is

Algorithm 4.1. *Compute relative closeness*

Require: two role r_1 and r_2
Require: system configuration $\langle USERS, PERMS, UPA\rangle$
Require: permission-operation relationships POA
Require: permission-resource relationships PRA
Require: weights for internal permission cohesion, internal user cohesion, operation centrality and resource centrality $\langle fp, fu, fo, fr\rangle$
1: $ru_1 = AuthorizedUsers(r_1)$
2: $rp_1 = AuthorizedPermissions(r_1)$
5: Compute original similarity between users in ru_1
6: $Avgsim = 0$
7: for each user $u_i \in ru_1$ do
8: $MinSim = 1$
9: for each user $u_j \in ru_1, u_j \neq u_i$ do
10: if $(MinSim > origsim(u_i, u_j))$ then
11: $MinSim = origsim(u_i, u_j)$
12: end if
13: end for
14: $Avgsim = Avgsim + MinSim$
15: $Avgsim = Avgsim\ |ru_1|$
16: end for
17: $IUC(r_1) = Avgsim$
18: Analogously compute $IPC(r_1)$ on rp_1
19: $ro_1 = AuthorizedOperations(rp_1)$
20: $rr_1 = AuthorizedResources(rp_1)$
21: $Entropy = 0$
22: for each $op_i \in ro_1$ do
23: $cPerm = 0$
24: for each $p_j \in rp_1$ do
25: if $(POA_{ji} \equiv 1)$ then
26: $cPerm = cPerm + 1$
27: end if
28: end for
29: $Entropy+ = \frac{cPerm}{|rp_1|} \log_2 \frac{cPerm}{|rp_1|}$
30: end for
31: $OperCen(r_1) = Entropy$
32: Analogously compute $ResCen(r_1)$ on rr_1
33: $\mathrm{RF}(r_1) = \langle \mathrm{IPC(r_1)}, \mathrm{IUC(r_1)}, \mathrm{OpC}(r_1), \mathrm{ResC}(r_1)\rangle$
34: Analogously compute features of r_2 $RF(r_2)$
35: $r = (ru_1 \bigcap ru_2, rp_1 \bigcup rp_2)$
36: Compute features of r $RF(r)$
37: $RC(r_1, r_2) = \langle fp, fu, fo, fr\rangle \times \frac{RF(r) - 0.5 \times RF(r_1) - 0.5 \times RF(r_2)}{RF(r)}$
38: return $RC(r_1, r_2)$

13. If adding "AssignGrad_GradeBook" to r_1 and r_2 shown in Figure 3(c), the WSC of this pruned state is 12. Such pruning action is supported in *HierarchicalMiner*. In this example, adding "AssignGrad_GradeBook" to r_1 will not change its central resource.

Algorithm 4.2. *The algorithm of FMiner*

Require: system configuration $\langle USERS, PERMS, UPA\rangle$
Require: permission-operation relationships POA
Require: permission-resource relationships PRA
Require: weight factors for complexity,
$W = \langle wr, wu, wp, wh\rangle$
Require: threshold of relative closeness τ
Require: threshold of WSC redundance ϵ
create reduced concept lattice $L = \langle R, UA, PA, RH\rangle$
$t\_r(RH) = t_reduce(RH)$
$wsc_{before} = wsc(L, W)$
$L_{copy} = L$
for each role $r \in L$ do
  $Sen(r) = r_i \in R|(r_i, r) \in t_r(RH)$
  $Jun(r) = r_j \in R|(r, r_j) \in t_r(RH)$
  $rp = AssignedPermissions(r)$
  $ru = AssignedUsers(r)$
  $CNU = 0$, $CNP = 0$
  if ($|rp| > 0$ and $|ru| \equiv 0$) then
    if ($RC(r, r_j) \geq \tau$) then
      $r_j.rp = r_j.rp \bigcup rp$
      delete edge (r, r_j)
      $\forall r_i \in Jun(r)$ add edge (r_i, r)
      $CNU = CNU + 1$
    end if
  end for
  else if ($|rp| \equiv 0$ and $|ru| > 0$)
  for each $r_i \in Jun(r)$ do
    if ($RC(r_j, \sum r_j - r_j) < \tau$) then
      $r_j.ru = r_j.ru \bigcup ru$
      delete edge $< r_j, r >$
      $\forall r_j \in Sen(r)$ add edge (r, r_j)
      $CNP = CNP + 1$
    end if
  end for
  else if ($|rp| \equiv 0$ and $|ru| \equiv 0$) then
  $\forall r_j \in Sen(r)$ add edge (r, r_j)
  $\forall r_i \in Jun(r)$ add edge (r_i, r)
  end if
  if (($CNU \equiv |Sen(r)|$ and $CNP \equiv |Jun(r)|$) then
    delete r
  end if
  Compute $t\_r(RH)$
  $wsc_{after} = wsc(L, W)$
  if ($(1 - \epsilon) \times wsc_{after} < wsc_{before}$) then
  $L = L_{copy}$
  else
  $wsc_{before} = wsc_{after}$
  $L_{copy} = L$
  end if
return L

However, adding r_2 will make its resource centrality more fuzzy. Thus, we only add "AssignGrad_GradeBook" to r_1. When we use the parameter Settings mentioned in Case 2, $RC(r_0, r_1) = 0.2 < \tau = 0.5$ and $RC(r_0, r_2) = 0.67 > \tau = 0.5$. Thus, the final RBAC state of this example is shown in Figure 3(d) with $wsc = 13$.

When we merge the roles with high relative closeness, sometimes this merging will make the WSC increase, e.g., Case 2. Therefore, we give the threshold ϵ to control the growth range of WSC. If ϵ, which judges whether the merging on RABC state is successful or not, is set to zero ($\epsilon = 0$), *FMiner* is the same with *HierarchicalMiner*, which merges roles that can make the value of WSC decrease. On other settings of ϵ, we allow the value of WSC increase below the value $\epsilon \times wsc$.

The details of *FMiner* is shown in Algorithm 4.2. For the given initial RBAC state $\gamma = \langle R, UA, PA, RH \rangle$ and a system configuration $\varphi = \langle U, P, UP \rangle$, we prune the RBAC state γ that makes the relative closeness between r and r_k is almost higher than τ and local WSC increased value below $\epsilon \times wsc$, where r_k is its immediate senior or immediate junior of r.

5. Performance Evaluation.

To evaluate the performance of our algorithm *FMiner*, we implement the algorithm by Java and run the program on the synthetic data set containing 400 users and 14 permissions. This data set has been used to evaluate *HierarchicalMiner* by Molloy et al. [21] and *StateMiner* by Takabi and Joshi [13]. Our experimental platform is a personnel computer with an Intel(R) Core(TM) 2 Duo CPU and 2GB memory.

5.1. Accuracy comparison.

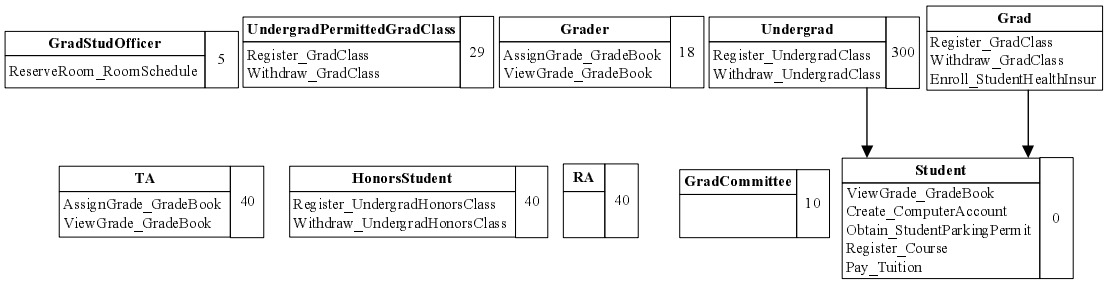
In this section, we evaluate the accuracy of *FMiner* and *HierarchicalMiner*. The accuracy of a role mining algorithm is defined as the ratio of the number of generated roles exactly matching the original role sets to the number of generated role sets.

$$\text{acc} = \frac{numRoles(Match)}{numRoles(Generate)}$$

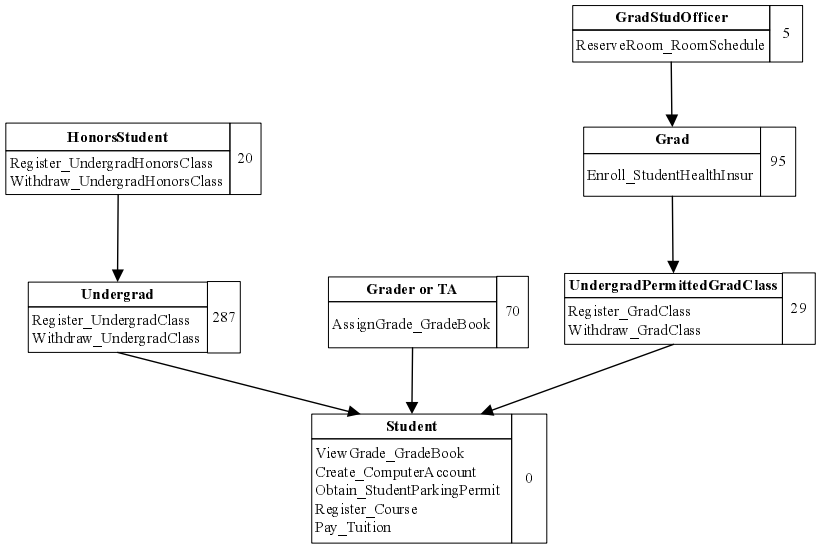
where $numRoles(Match)$ is the number of the exactly matched between the original role sets and the generated role sets, and $numRoles(Generate)$ is the number of all the generated roles.

The original roles are shown in Figure 4(a). Figure 4(b) shows the RBAC state generated by *FMiner*, and Figure 4(c) shows the RBAC state generated by *HierarchicalMiner*. *StateMiner* generates the same roles with *HierarchicalMiner* in Figure 2(b) of [13] with a little different WSC. Therefore, we do not show the RBAC state generated by *StateMiner* in Figure 4. The accuracy of *HierarchicalMiner* in this data set is $6 \div 10 = 60\%$, while that of *FMiner* is $5 \div 7 = 71\%$. The role "RA" and "GradCommittee" are not generated by both algorithms. *FMiner* does not generate any composite roles, while *HierarchicalMiner* generates some composite roles, such as "UndergradPermittedGradClass and HonorsStudent" and "UndergradPermittedGradClass and Grader". These composite roles reduce the accuracy of *HierarchicalMiner*.

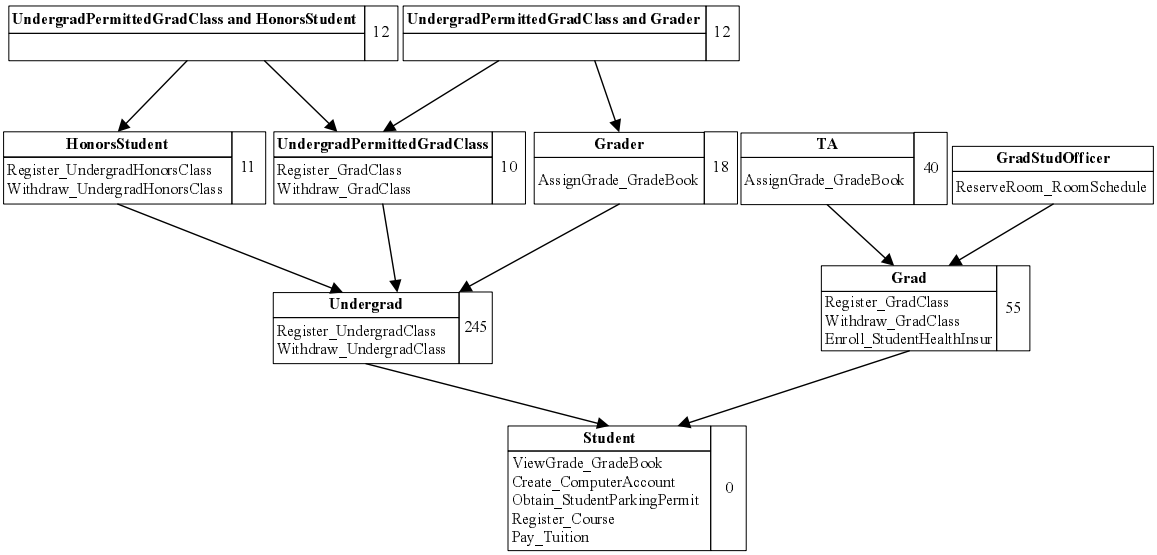
FMiner does not retain the composite roles "HonorStudent and UndergradPermittedGradClass" and "HonorStudent and Grader" because $RC(HonorStudent, Grader) = 0.92$ and $RC(HonorStudent, UndergradPermittedGradClass) = 0.79$. If we view the operation and resource as the same importance on the role feature, "HonorStudent and UndergradPermittedGradClass" will be retained. Though merging "HonorStudent" to "UndergradPermittedGradClass" makes the composite role add a new resource "GradClass", the original operations of these roles are the same. Therefore, the features of this composite role are not changed too much.



(a) The original roles



(b) The *FMiner* result



(c) The *HierarchicalMiner* result

FIGURE 4. These graphs represent the set of roles in the student part of the university datasets. The original roles are shown in the top. The RBAC state with $wsc = 533$ generated by *FMiner* is shown in the middle. The RBAC state with $wsc = 443$ generated by the *HierarchicalMiner* is shown in the last.

TABLE 1. The *FMiner* results for fixed ϵ

	$W = \langle 1,1,1,1 \rangle$					$W = \langle 1,1,2,2 \rangle$				
	R	UA	PA	RH	WSC	R	UA	PA	RH	WSC
Original	32	799	35	19	885	32	799	35	19	885
$\tau = 0.2$	22	599	56	24	701	22	599	56	24	781
$\tau = 0.4$	22	599	56	24	701	22	599	56	24	781
$\tau = 0.6$	21	579	62	23	685	21	579	62	23	770
$\tau = 0.8$	21	572	62	22	677	21	572	62	22	761
$\tau = 1.0$	24	493	69	27	613	24	493	69	27	709
HierarchicalMiner	21	498	67	19	605	21	505	67	20	696
StateMiner	24	498	63	27	612	24	498	63	27	702
optimal	19	496	59	14	600	19	496	57	16	685

TABLE 2. The *FMiner* results for fixed τ

	$W = \langle 1,1,1,1 \rangle$					$W = \langle 1,1,2,2 \rangle$				
	R	UA	PA	RH	WSC	R	UA	PA	RH	WSC
Original	32	799	35	19	885	32	799	35	19	885
$\epsilon = 0.02$	24	532	56	28	640	23	559	56	26	746
$\epsilon = 0.04$	23	559	56	26	664	23	559	56	26	746
$\epsilon = 0.06$	22	559	56	24	701	22	599	56	24	781
$\epsilon = 0.08$	22	559	56	24	701	22	599	56	24	781
$\epsilon = 0.10$	22	559	56	24	701	22	599	56	24	781
HierarchicalMiner	21	498	67	19	605	21	505	67	20	696
StateMiner	24	498	63	27	612	24	498	63	27	702
optimal	19	496	59	14	600	19	496	57	16	685

5.2. **Quality measurement.** In this section, we evaluate the quality of the generated roles by *FMiner* compared to *HierarchicalMiner* and *StateMiner*. The experiments are carried out on the data set with 493 users and 56 permissions. The experiments are to measure RBAC state generated by *FMiner* on different parameter settings.

In the first scheme, we fix the threshold of the relative closeness τ while changing the user specified parameter ϵ. The columns of R, UA, PA, RH, WSC represent the cost for role, user assignment, permission assignment, and the weighted structural complexity of RBAC state. This experiment is done on $W = \langle 1,1,1,1 \rangle$, $W = \langle 1,1,2,2 \rangle$ and $\epsilon = 0.1$. The result is shown in Table 1.

In the experimental results shown in Table 1, the number of roles generally increases with the increasing of τ. As the value of τ gets bigger, the roles have more opportunities to merge with its immediate juniors or seniors. However, our merging strategy is a partial merging approach. Thus, not all the initial roles will be deleted after merging. When the bigger τ make more composite roles in Case 3, the bigger τ also makes more composite roles in Case 2. The initial roles with no permissions are more than those with no users in this data set. Therefore, the value of WSC is reduced with the increase of τ. Usually, the junior roles in an RBAC state have little users and the senior roles have little permissions. Large value of τ will make many roles without permissions but with some users. Some roles without users but with some permissions still exist after pruning process.

In the second scheme, we fix τ while changing ϵ. The experimental results are shown in Table 2. The value of WSC is going up with the increase of ϵ. The larger ϵ means the roles with high relative closeness have more opportunities to be merged. In *FMiner*, if

the merging action makes the increase of WSC beyond the range of ϵ, this action will not be allowed, and the pruned RBAC state will be returned to pruning before. When we set a higher ϵ, the less composite roles will be generated in pruning process. However, the higher ϵ, the more R, UA, PA and RH in an RABC system, which makes the management of the RBAC state more complex. Therefore, the setting of ϵ should be adjusted based on the system and application requirements.

6. **Conclusions.** This paper proposes to use operations and resources of the permissions as the function information in role mining and presents a new role engineering approach *FMiner* that could reduce composite roles. Our algorithm has two main processes. Firstly, we generate the initial RBAC state that each permission only belongs to a role using formal concept analysis. Secondly, we prune this RBAC state based on weighted structural complexity (WSC) and relative closeness. The experimental results demonstrate the effectiveness of the proposed approach on reducing composite roles. For the future work, we will use the business information to create more meaningful initial roles.

Acknowledgments. This research is partially supported by National Natural Science Foundation of China under grants 61173170 and 60873225, National High Technology Research and Development Program of China under grant 2007AA01Z403, Natural Science Foundation of Hubei Province under grant 2009CDB298, Innovation Fund of Huazhong University of Science and Technology under grants 2012TS052, 2011TS135 and 2010MS06 8, and CCF Opening Project of Chinese Information Processing.

REFERENCES

[1] D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn and R. Chandramouli, Proposed nist standard for role-based access control, *ACM Transactions on Information and System Security*, vol.4, no.3, pp.224-274, 2001.

[2] R. S. Sandhu, E. J. Coyne, H. L. Feinstein and C. E. Youman, Role-based access control models, *IEEE Computer*, vol.29, no.2, pp.38-47, 1996.

[3] E. J. Coyne, Role engineering, *Proc. of the 1th ACM Workshop on Role-Based Access Control*, 1995.

[4] M. Frank, J. M. Buhmann and D. Basin, On the definition of role mining, *Proc. of the 15th ACM Symposium on Access Control Models and Technologies*, pp.35-44, 2010.

[5] I. Molloy, N. Li, T. Li, Z. Mao, Q. Wang and J. Lobo, Evaluating role mining algorithms, *Proc. of the 14th ACM Symposium on Access Control Models and Technologies*, pp.95-104, 2009.

[6] A. Baumgrass, M. Strembeck and S. R. Ma, Deriving role engineering artifacts from business processes and scenario models, *Proc. of the 16th ACM Symposium on Access Control Models and Technologies*, pp.11-20, 2011.

[7] M. Frank, D. Basin and J. M. Buhmann, A class of probabilistic models for role engineering, *Proc. of the 15th ACM Conference on Computer and Communications Security*, New York, pp.299-310, 2008.

[8] M. Frank, A. P. Streich, D. Basin and J. M. Buhmann, A probabilistic approach to hybrid role mining, *Proc. of the 16th ACM Conference on Computer and Communications Security*, New York, pp.101-111, 2009.

[9] A. Colantonio, R. D. Pietro and A. Ocello, A cost-driven approach to role engineering, *Proc. of the 2008 ACM Symposium on Applied Computing*, 2008.

[10] D. Zhang, K. Ramamohanarao and T. Ebringer, Role engineering using graph optimisation, *Proc. of the 12th ACM Symposium on Access Control Models and Technologies*, pp.139-144, 2007.

[11] D. Zhang, K. Ramamohanarao, T. Ebringer and T. Yann, Permission set mining: Discovering practical and useful roles, *Proc. of the 2008 Annual Computer Security Applications Conference*, pp.247-256, 2008.

[12] J. Vaidya, V. Atluri and Q. Guo. The role mining problem: Finding a minimal descriptive set of roles, *Proc. of the 12th ACM Symposium on Access Control Models and Technologies*, pp.175-184, 2007.

[13] H. Takabi and J. B. D. Joshi, StateMiner: An efficient similarity-based approach for optimal mining of role hierarchy, *Proc. of the 16th ACM Symposium on Access Control Models and Technologies*, pp.55-64, 2010.

[14] X. Ma, R. Li and Z. Lu, Role mining based on weights, *Proc. of the 15th ACM Symposium on Access Control Models and Technologies*, pp.65-74, 2010.

[15] Q. Guo, J. Vaidya and V. Atluri, The role hierarchy mining problem: Discovery of optimal role hierarchies, *Proc. of 2008 Annual Computer Security Applications Conference*, pp.237-246, 2008.

[16] M. Kuhlmann, D. Shohat and G. Schimpf, Role mining – Revealing business roles for security administration using data mining technology, *Proc. of the 8th ACM Symposium on Access Control Models and Technologies*, pp.179-186, 2003.

[17] J. Vaidya, V. Atluri and J. Warner, Roleminer: Mining roles using subset enumeration, *Proc. of the 13th ACM Conference on Computer and Communications Security*, pp.144-153, 2006.

[18] J. Schlegelmilch and U. Steffens, Role mining with ORCA, *Proc. of the 10th ACM Symposium on Access Control Models and Technologies*, pp.168-176, 2005.

[19] G. Karypis, E.-H. Han and V. Kumar, CHAMELEON: A hierarchical clustering algorithm using dynamic modeling, *IEEE Computer*, vol.32, no.8, pp.68-75, 1999.

[20] A. Ene, W. Horne, N. Milosavljevic, P. Rao, R. Schreiber and R. E. Tarjan, Fast exact and heuristic methods for role minimization problems, *Proc. of the 13th ACM Symposium on Access Control Models and Technologies*, pp.1-10, 2008.

[21] I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. Calo and J. Lobo, Mining roles with semantic meanings, *Proc. of the 13th ACM Symposium on Access Control Models and Technologies*, pp.21-30, 2008.

[22] G. Neumann and M. Strembeck, A scenario-driven role engineering process for functional RBAC roles, *Proc. of the 7th ACM Symposium on Access Control Models and Technologies*, pp.33-42, 2002.

[23] H. Takabi and J. B. D. Joshi, An efficient similarity-based approach for optimal mining of role hierarchy, *Proc. of the 16th ACM Conference on Computer and Communications Security*, 2009.