Role Mining Algorithms Satisfied the Permission Cardinality Constraint

Jingyu Wang, Jingnan Dong, and Yuesheng Tan
(Corresponding author: Jingnan Dong)

School of Information Engineering, Inner Mongolia University of Science and Technology
7 Aerding Street, Baotou City, Inner Mongolia, 014010, People's Republic of China
(Email: 867324154@qq.com)

(Received Nov. 16, 2018; Revised and Accepted Mar. 17, 2019; First Online June 16, 2019)

Abstract

With the increase of access control size in big data, the roles in most of the existing role mining algorithms have overmuch permission, which conveniently results in fraud. Therefore, this paper proposes two kinds of role mining algorithms that satisfy the permission cardinality constraints. Algorithms 1 divide each row of the sorted access control matrix to generate a set of roles. Algorithm 2 intersect the permission of adjacent users of the ordered access control matrix to generate a candidate role sets. The iterative reduction is then performed multiple times on the basis of the candidate role sets to produce most of the decisive role sets. Both algorithms first perform row and column sorting on the access control matrix, which draw on word frequency statistics and frequent item mining, respectively. The data applied in the experiment are a common real data set. Relevant experimental results demonstrate that the performance of algorithm 2 is superior to algorithm 1 and PRUCC-RM.

Keywords: Access Control; Frequent Item Mining; Permission Cardinality Constraints; Role Mining; Word Frequency

1 Introduction

Role based access control has been one of the most widespread way of access control.The key of applying the model is the definition of the roles. The solution of the question is divided in two types, one is a top-down method, which gain roles by analyzing users' circumstance and the business process [18], the other way is a down-top way,which utilize data mining technology to find roles from existing user permission assignment relationship (UPA). The user permission assignment relation that specifies which individuals had access to which resources in the original system can be presented in the form of a boolean matrix. Mitra divides the role mining model into the deterministic model and the probabilistic model [12]. Vaidya defined the basic role mining problem that finding a minimal set of role from an input UPA that provides an equivalent user permission assignment [19], who shows the problem of finding the minimal set of descriptive roles and relationships without disturbing permission assignments is NP-complete. There exist various ways to mine roles.Belim presents an algorithm for analyzing the matrix of authorized user permission for optimal role formation [1], which solves the role mining problem with the help of graph theory knowledge.Dong uses bipartite network models for role mining and solves problems from the perspective of edge importance in complex networks [3]. Huang converted Basic role mining problem to the Set Cover Problem [6]. There are a few role mining algorithms with machine learning model. Constraints are a powerful mechanism for arranging high-level organizational strategies. In addition, there are malicious activities in RBAC [15], by applying the constraints, the rate in database attacks can be reduced, and fraudulent behavior can be prevented. It is necessary for enterprises to conduct role mining meeting constraints in implementing RBAC. Ye proposed a novel role mining approach useing answer set programming (ASP) that meets various optimization objectives, named constrained role miner [21].

There are three constraints in role-based access control. That is separation of duties, cardinality constraints and prerequisite constraints.Separation of duties is widely used in situations where multiple people need to work together to perform a sensitive task, but not by fewer people to prevent fraud. There are for two situations about prerequisite constraint. One is prerequisite constraint on the role which specifies that one user can obtain role $r1$ only after the user has obtained role $r2$. The other is prerequisite constraint of permission which specifies that a permission can be assigned to a certain role only after the role possesses permission p. There are four situations about cardinality constraint, one is permission cardinality which specifies the maximum number of permission a role can have in a RBAC system; Second is user cardinality constraint which specifies the maximum number of user a role can belong to; Third is role usage cardinality constraint which specifies the limited number of roles which each user can have; The last one is permission assignment cardinality constraint which specifies the limited number of roles that each permission can be assigned.

The rest of the paper is framed as follows: Section 2 reflects the related work. Section 3 exhibits the related terms needed to understand the algorithms. Section 4 introduces in detail the execution process and flow chart of the two algorithms proposed in this paper, and carries on the algorithm analysis. Section 5 uses real public data for algorithm experiments,and analyses the results. Finally, we conclude our work in Section 6.

2 Related Works

Although the algorithms of role mining have been proposed [7], the role which most algorithm generate don't meet cardinality constraint [16], which makes many roles own overmuch permission, violating separation of duty constraints easily, and being harmful to redistribution. For roles generated meet permission cardinality constraints. One which uses the combination of clustering and limited permission set mining gets roles with meeting given cardinality constraint [8], it is very time consuming to mine role from a collection of disorganized user permission, because a role is mined each time. The statistics of the user's no-visited permission are required, and also calculates all possible intersections of all users' no-visited permission. Role usage cardinality constraint and permission assignment cardinality constraint is conflicting between the relationship of users role assignment and the relationship of role permission assignment, trying to meet one of the two constraints may lead to violation of the others, constrained role mining problem is a NP complete problem [5], Harika proposes two different frameworks, one is to implement constraint after role mining, and the other is to implement constraint separately or simultaneously in the role mining process. One proposed a role mining method that are based permission cardinality constraint and user cardinality constraint, which merge roles by the similarity of roles to improve precision of roles' state, when it is running, it will consume substantial time and calculation about similarity of roles [11].

One proposed two heuristic role mining algorithms that satisfy both the permission cardinality constraint and the role usage cardinality constraints, since it randomly selects a row with the least number of users, the permission is defined as a role, so there is uncertainty, and the permission sets whose number is minimum is not necessarily a frequent item set [2]. The above method selects a constraint from the user role relationship and the role permission relationship to control the relationship between the user and the permission. The advantage of using role-based access control is that the role is used to bridge the gap between users and permission, users can get the required permission indirectly through the role. A user needs to remember all kinds of ID (identities) and passwords in multi-server environment [14], and outsourcing large-scale computing tasks to the cloud [10], so there are a large number of relationships between users and permission stored in enterprise. By implementing role-based access control, access control matrix can be simplified to facilitate enterprise storage and analysis.

The purpose of finding the smallest set of roles is to be able to express users and permissions with it. By mining association rules between permission. It can help define a role so that it can replace more relationship between users and permission. Take into account this consideration, the paper decided to design the algorithm based on the ideas of word frequent statistics and frequent item mining.

3 Constrained Role Mining Problem

Fundamental RBAC model has been introduced. We will only introduce some concepts about implementing the algorithm in this part.

Definition 1 (RBAC)**.** *The Role Based Access Control (RBAC) model comprises the following components [17]:*

- $U = \{u_1, u_2, \ldots, u_n\}$, *U represents user set;*
- $P = \{p_1, p_2, \ldots, p_n\}$, *P represents permission set;*
- $R = \{r_1, r_2, \ldots, r_n\}$, *R represents the role set;*
- $UA \subseteq U \times R$, *UA is a relationship from U to R;*
- $PA \subseteq R \times P$, *UA is a relationship from R to P.*

Definition 2 (Basic role mining problem, RMP)**.** *Given a set of users U, a set of permission P and a user permission access control matrix UPA, find a group of roles R, user role assignment relationship UA and role permission assignment relationship PA. Satisfy $UA \otimes PA = UPA$, and minimizes $|R|$.*

Definition 3 (Permission cardinality constraint)**.** *The permission cardinality constraint specifies the maximum number of permission a role can have in a RBAC system.*

Definition 4 (User cardinality constraint)**.** *The user cardinality constraint is specified as the maximum number of user that a role can belong to in a RBAC system.*

Definition 5 (Role usage cardinality constraint)**.** *The role usage cardinality constraint is defined as the limited number of roles which each user can have.*

Definition 6 (Permission assignment cardinality constraint)**.** *The permission assignment cardinality constraint is defined as the limited number of roles that each permission can be assigned.*

When an enterprise establishes a RBAC system, the relationship of user and permission will change with the development of the enterprise. Therefore, the company should control user cardinality constraint and role usage cardinality constraint in a dynamic process. The permission assignment cardinality constraint does not need to be considered, because the purpose of controlling the permission assignment cardinality can be achieved by controlling the role usage cardinality constraint. The permission cardinality constraint could satisfy the separation of duties constraints to prevent roles from having mutually exclusive permission. When a user accesses the allocation of resources in the cloud, if the number of permission of the role is limited, the hierarchical management of the role can be facilitated [20]. So the algorithms proposed in this paper is to mine the role in the access control matrix meeting the permission cardinality constraint.

Definition 7 (Weighted Structural Complexity, WSC)**.** *[13]. Given $W =< w_r, w_u, w_p, w_h, w_d >$, where $w_r, w_u, w_p, w_h, w_d \in Q^+ \cup \{\infty\}$, the Weighted Structure Complexity (WSC) and RBAC state γ, which is denoted as $wsc(\gamma, w)$, is computed as follows. We have*

$$\begin{aligned} wsc(\gamma, w) &= w_r * |R| + w_u * |UA| + \\ &\quad w_p * |PA| + w_h * |t\_reduce(RH)| + \\ &\quad w_d * |DUPA| \end{aligned} \quad (1)$$

Where $|\cdot|$ denotes the size of the set or relations, and $t_reduce(RH)$denotes the transitive reduction of the rolehierarchy.Since the proposed algorithm does not consider the role in heritance, and does not allow the authority to be directly assigned to the user, by setting $w_r = w_u = w_p = 1, w_h = w_d = \infty$. Arithmetic involving ∞ is defined as follows: $0 * \infty = 0$,$\forall x \in Q^+$ $x * \infty = \infty$,$\forall x \in Q \cup \{\infty\}$ $x + \infty = \infty$.

Definition 8 (Role Mining Problem Satisfied the Permission Cardinality Constraint)**.** *Given a set of user $U = \{u_1, u_2, \cdots, u_n\}$, a set of permission $P = \{p_1, p_2, \cdots, p_n\}$, a user permission access control matrix UPA, and a positive integer $t, t > 1$. Find a group of roles $R = \{r_1, r_2, \cdots, r_q\}$, a user role assignment relationship UA and a role permission assignment relationship PA, Satisfy $UA \otimes PA = UPA$, $\forall r_i \in R$ $|PermsR(r_i)| \leq t$, $1 \leq i \leq q$, $PermsR(r_i)$ represents the permission that role r_i own, and minimize WSC.*

The evaluation goal of role mining cannot be measured by reducing the number of the roles.Because simply reducing the number of roles, in order to achieve, finally lead to the increment of UA and PA. Taking the reduction of WSC as the final metric can fully measure the definitive role mining situation.

4 Algorithm Overview

In machine learning algorithms, FP-growth algorithm is an algorithm founded on association rules.It can mine a group of items with a strong correlation.So it could be used to basket analysis,merchant arranges placement of goods conveniently and bundled sale of goods.Mapping the user permission access control matrix to the user's shopping list, where the permission represents the product. If you use FP-growth algorithm, you can mine the frequent permission set in the access control matrix and define the frequent item set as a role. Where the permission cardinality constraint can be defined as the permission set has at most several permission, so as to achieve the purpose of satisfying the defined cardinality constraint.When the data set is large, the FP-growth algorithm recursively generates a large number of conditional pattern libraries and conditional FP-tree. In this situation, the algorithm needs excessive memory and has low efficiency [4]. Additionally,a role mining algorithm as a method for frequent item mining. It becomes less efficient due to the larger set of roles and multiple iterations of the FP-growth algorithm.This paper presents a method of clustering frequent permission sets in the access control matrix, clustering similar permission sets, and generating role sets by iterative simplification. Finally, the set of roles corresponding to each user is found in the access control matrix by using the generated set of roles.

4.1 Role Mining Algorithm Satisfied the Permission Cardinality Constraint

Algorithm 1 Role mining algorithm based on word frequency statistics

```
Input: Access control matrix:$UPA$,Permission
    Cardinality:$Limited$.
Output: Role set:$R$, the relationship of user and
    role:$UA$,$WSC$.
Begin
Column of $UPA$ is sorted from left to right in descending order according to the number of users of the permission.
Sort the rows of $UPA$ from top to bottom in descending order,depending on the location of the permission of $UPA$.If the user has permission to the front left,it will be ranked first.
The location which is 1 is replaced by the original order of the permission in UPA.
Define every $Limited$ permission from left to right for each user in UPA as a role.
Define each role as a key in a key-value pair whose number of occurrences is defined as the value of the key.
Generate user role relationships ($UA$) based on generated role sets ($R$) and access control relationship F($UPA$).
$WSC = |UA| + |PA| + |R|$, UA represents the role set.
End
```

By sorting the rows and columns of UPA, it is possible to have similar permission gathered together.Defining every *limited* permissions for each user in the access control as a role, which not only meets permission cardinality constraint, but also limits the number of each user's role. In addition, the idea of the algorithm draws on word frequency statistics.By sorting the UPA,each row of the UPA can be treated as a string of characters,and the permission cardinality constraint is the number of characters that specify the intercepted zero-free word (the user does not get the permission).

Algorithm 2 Iterative based role mining algorithm

```
Input: Access control matrix:$UPA$,Permission
    Cardinality:$Limited$,Iterative benchmark:$IteraBench$.
Output: Role set:$R$, the relationship of user and
    role:$UA$,$WSC$.
 1: Begin
 2: Column of $UPA$ is sorted from left to right in descending order according to the number of users of the permission.
 3: Sort the rows of $UPA$ from top to bottom in descending order,depending on the location of the permission of $UPA$.The user has permission to the front left,which will be ranked first.
 4: The location which is 1 is replaced by the original order of permission in $UPA$.
 5: Define the maximum intersection of the permission of neighboring users in the matrix as a role.
 6: Sort the generated role set according to the number of permission included, and deletes the smaller role.
 7: Select a role greater than the iteration cardinality line ($IteraBench$) from the bottom to the top of the matrix as a temporary role $r^{`}$.
 8: $\forall R \supset r^{`}, R = R - r^{`}$
 9: Sort candidate role set.
10: Replace the relationship between users and roles in the access control matrix with candidate role set.
11: if There is a user's remaining permission which cannot be replaced by the roles in candidate role set then
12:     Generate a role to add to candidate role set for each user's remaining permissions that cannot be replaced with roles in role set,
13:     goto 7
14: else
15:     Split the role who's the number of permission greater than $Limited$ and generate some new roles.
16: end if
17: Generate user role relationships ($UA$) based on generated role sets ($R$) and access control relationship ($UPA$)
18: $WSC = |UA| + |PA| + |R|$, $UA$ is the role set.
19: End
```

Algorithm 2 also is required to perform the same sorting as algorithm 1. The intersection of the permissions owned by each user and neighboring users' is to extract the largest identical portion locally. The global optimum is accomplished as much as possible by local optimization. Iterative benchmark defines the criteria for iteration in the process of role set reduction,if it is too large,it is easy to violate the permission cardinality constraint. In the 6th step, if the size of the candidate role is less than 4 and its frequency is less than 2, we will delete it.About the permission owned by a small number of users, it is likely to be a key permission, so this article also defines it as a role, when assigning, it should limit the assigned amount of such roles and prevent unauthorized operation.In order to ease the understanding of the processing of the two algorithms, we present flow charts of the two algorithms. As showed in Figure 1 and Figure 2, We can see from the figure that the two algorithms are the same ones at the beginning and end, but the access matrix processing method after sorting is different. It is worth mentioning that in fact, the content of the loop in the flowchart of algorithm 2 will only be executed once. Because the process is performed only once, the remaining permissions of all users are generated by the relevant roles, and the loop will not be run.

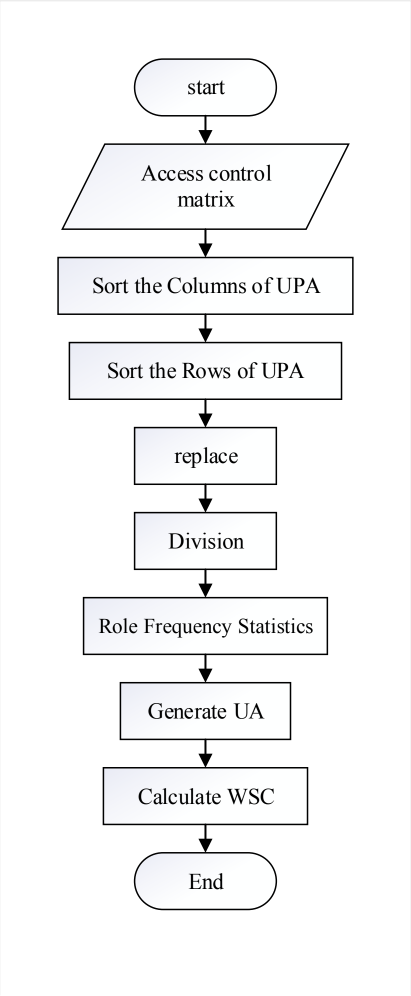


Figure 1: The flow chart of Algorithm 1

4.2 Analysis of Algorithms

We will illustrate the general operation of the algorithms, for example, Table 1 is an access control matrix. After column sorting and row sorting, it will produce the matrix

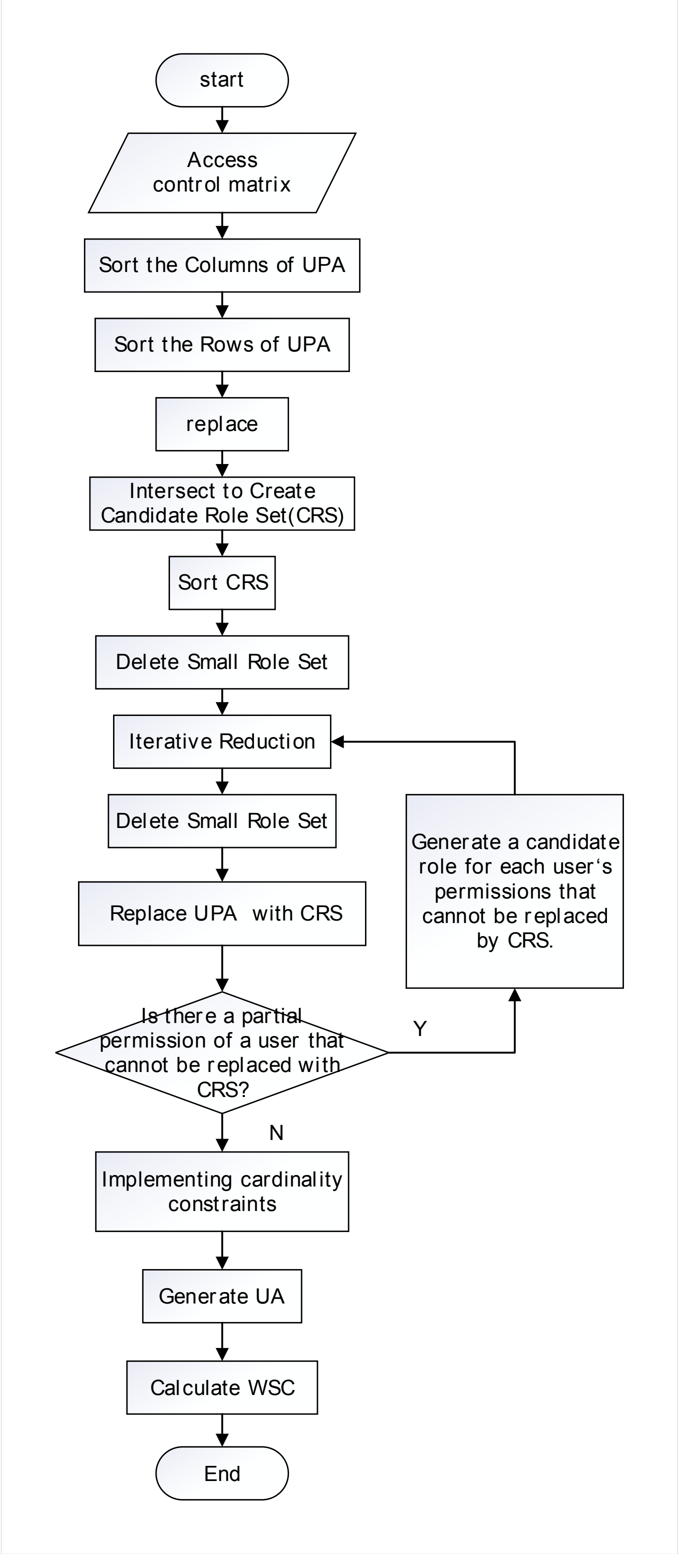

start
Access control matrix
Sort the Columns of UPA
Sort the Rows of UPA
replace
Intersect to Create Candidate Role Set(CRS)
Sort CRS
Delete Small Role Set
Iterative Reduction
Delete Small Role Set
Replace UPA with CRS
Is there a partial permission of a user that cannot be replaced with CRS?
Y
Generate a candidate role for each user's permissions that cannot be replaced by CRS.
N
Implementing cardinality constraints
Generate UA
Calculate WSC
End

Figure 2: The flow chart of Algorithm 2

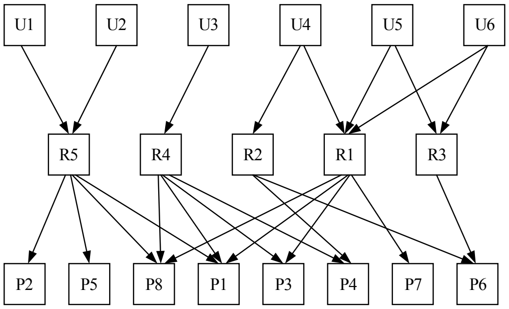


Figure 3: The result of Algorithm 1

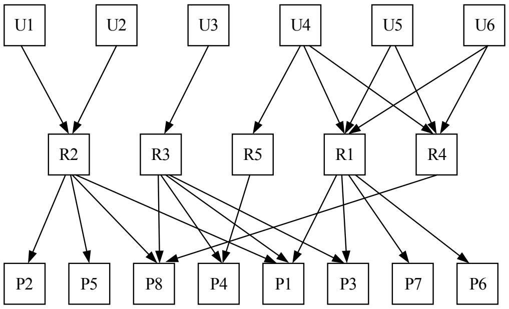


Figure 4: The result of Algorithm 2

shown in Table 2, The number in the matrix is the number of the column's permission, From the Table, we can see that sorting makes the original disordered access control matrix more regular. It is more convenient to mine roles on the matrix. Algorithm 2 mine the roles from the perspective of the line of the UPA. In this example,the permission cardianlity is set to four, The relationship between users,roles and permissions through algorithm 1 is shown in Figure 3, Each segmented word represents a role. Because a role in a candidate role set might be a subset of another role, we set a benchmark for iterative reduction. The number of inclusion relationships between roles decreases by reduction. The relationship between users, roles and permission through algorithm 2 is given in Figure 4. The time complexity is required for algorithm 1 to sort the access control matrix once is $O(c\log_2 c)$,where c represents the number of columns in the access control matrix. The time complexity required to perform a row sort is $O(r\log_2 r)$, where r represents the number of rows in the access control matrix. Furthermore, generating a set of roles requires traversing the access control matrix once, and generating a user role assignment relationship also needs to be traversed once. Therefore the total time complexity of algorithm 1 is $O(c\log_2 c+r\log_2 r+2cr)$.The spatial complexity of algorithm 1 is $O(cr+|UA|+|PA|)$. UA represents user role relationship. PA represents role permission relationship.

Algorithm 2 also needs to first sort the row and column of the access control matrix, and the time complexity is $O(c\log_2 c+r\log_2 r)$, Each row in the access control needs to produce an intersection with the adjacent row, which needs to be compared $2c(r-2)$ times. Multiple iterative

Table 1: Access control matrix

user	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**
U1	1	1	0	0	1	0	0	1
U2	1	1	0	0	1	0	0	1
U3	1	0	1	1	0	0	0	1
U4	1	0	1	1	0	1	1	1
U5	1	0	1	0	0	1	1	1
U6	1	0	1	0	0	1	1	1

Table 2: Sorted access control matrix

user	**P1**	**P8**	**P3**	**P7**	**P6**	**P5**	**P4**	**P2**
U4	1	8	3	7	6	0	4	0
U5	1	8	3	7	6	0	0	0
U6	1	8	3	7	6	0	0	0
U3	1	8	3	0	0	0	4	0
U1	1	8	0	0	0	5	0	2
U2	1	8	0	0	0	5	0	2

reduction occurs when the final role set is generated. In the process of iteration, only if the role's number of permission is greater than the given value (that is greater than the IteraBench defined in the algorithm 2), the role can be iterated and simplified with the previous roles in the array of role sets. So the time complexity of the part is approximately $O(|R|cr)$. At last, generating user role relationships needs to be compared $|PA|cr$ time. Consequently, the total time complexity of algorithm 2 is approximately $O(n \log_2 n + (1 + |R| + |PA|)cr)$, n is the maximum of c and r. The spatial complexity of algorithm 2 is $O(|UA| + |PA| + cr)$. There are a few places in algorithm 1 and algorithm 2 which can be changed to parallel operations, such as the division of permissions for each user and the generation of intersections of adjacent rows.

5 Experimental Evaluation

In the following sections, we present the experimental evaluation of our algorithms and PRUCC-RM which is proposed by Blundo and satisfies the permission cardinality constraint [2]. The test platform hardware is 3.4Ghz Intel CPU and 8 GB memories. The operation system is Windows 7, the program is run in a VMware virtual machine, the operating system image used is Ubuntu, and its version number is 16.04 LTS. In the Spark pseudo-distributed cluster, using Scala high-level programming language to perform programming experiments in Intellij IDEA.

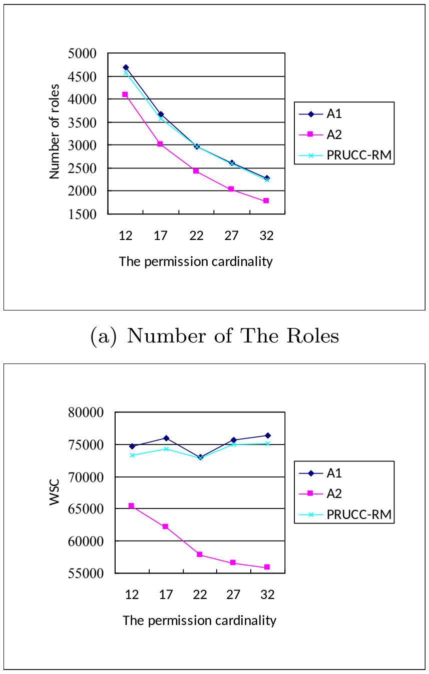
In order to be possible to repeat this experiment and the data sets used are all public data sets, which have been used in the literature [2, 5, 8]. The URL for downloading the role mining tool RMiner is given there [9]. Through this website, not only you download the public data set, but you can also download the role mining tool RMiner. Table 3 lists sizes, execution time and defined iteration benchmark for each data set, where userMaxPerm represents the maximum number of permissions a user has in the data set. *IteraBench* is the threshold for each iteration of different data sets, and the reader can choose the threshold of the iteration. Finally, we also represent the execution time of three algorithms in the worst case.Algorithm 2 takes longer than algorithm 1 because it requires multiple iterations. Since PRUCC-RM does not have row and column ordering, it takes the least amount of time. When the data set is small, the running time of the three algorithms is approximately equal. Defining algorithm 1 as A1 and algorithm 2 as A2.

From Figure 5 to Figure 12, it can be observed that as the permission cardianlity increases, the number of roles generated by the three algorithms is gradually decreasing. Because the maximum number of permission each role can have, resulting in some users have all the permission which can be replaced by fewer roles.Some data sets appear with the increase of the permission cardinality, and the number of roles subsequently grows. Algorithm 1 divides the permission of each user according to the permission cardinality. When the permission cardianlity is set to an appropriate value, the number of generated roles is minimized. When the permission cardianlity exceeds the appropriate value, more roles are generated to satisfy the relationship between the user and the permission. In terms of weighted structural complexity, the overall trend of algorithm 1 decrease first and then increases with the increase of the permission cardinality. Because as the number of the permission cardinality increases, the number of roles decreases overall, while the size of $|PA|$ may increase, and the size of $|UA|$ may decrease. When a suit-

Table 3: Characteristics of real data sets

Data Set	|U|	|P|	userMaxPerm	IteraBench	Execution time(s)		
					A1	A2	PRUCC-RM
Americas_Large	3485	10127	733	10	251.541	273.533	24.113
Americas_Small	3477	1587	310	13	20.179	24.968	11.006
Apj	2044	1164	58	12	11.531	12.258	9.701
Emea	35	3046	554	10	7.444	7.778	7.231
Healthcare	46	46	46	10	7.176	7.132	7.165
Domino	79	231	209	10	7.058	7.310	7.385
Firewall1	365	709	617	15	7.841	8.137	7.599
Firewall2	325	590	590	12	7.443	7.827	7.875

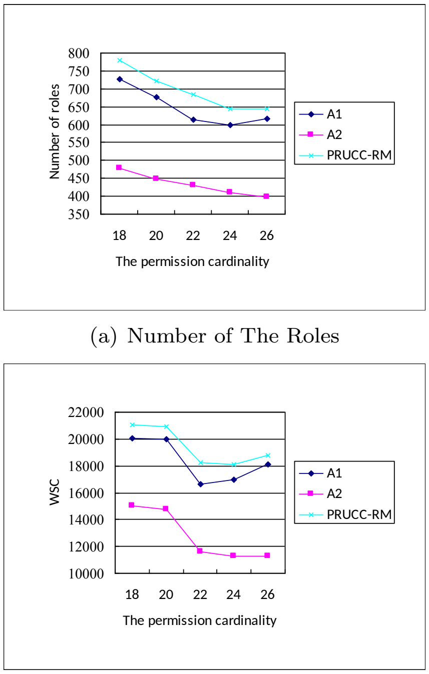
[1] http://code.google.com/p/rminer/



(a) Number of The Roles

(b) WSC

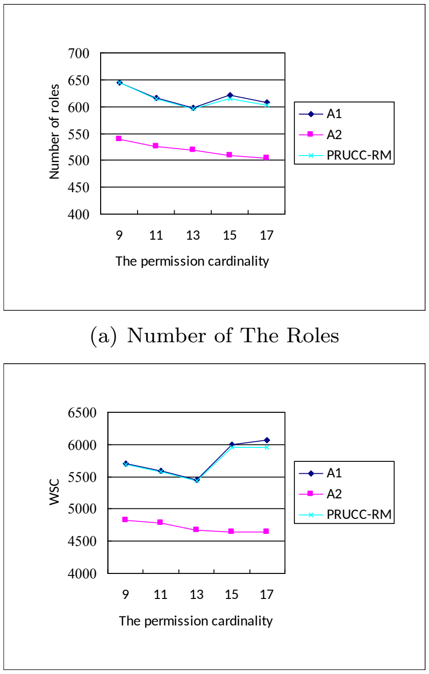
Figure 5: Experimental results of three algorithms in Americas_Large



(a) Number of The Roles

(b) WSC

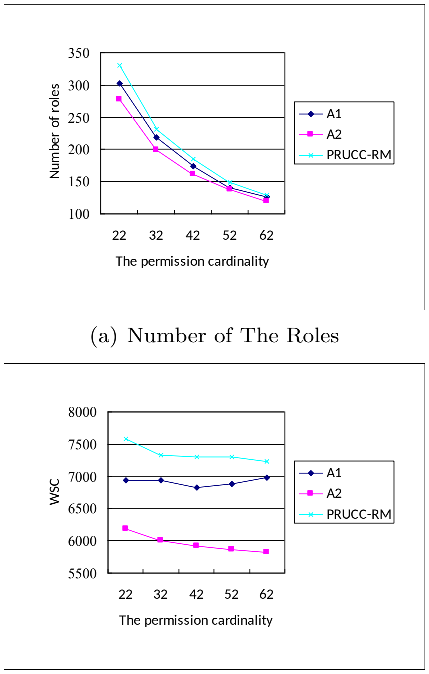
Figure 6: Experimental results of three algorithms in Americas_Small



(a) Number of The Roles

(b) WSC

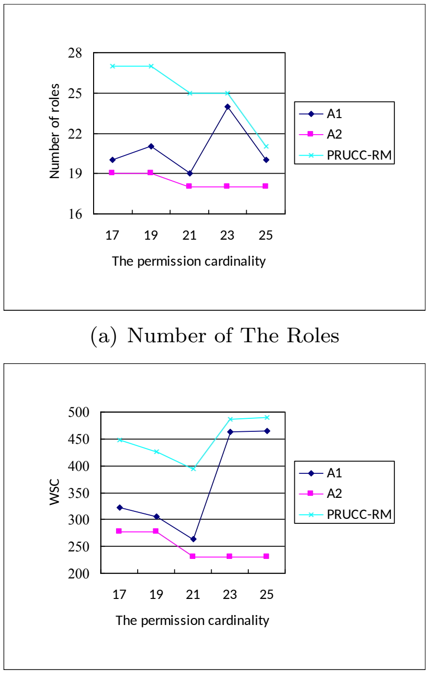
Figure 7: Experimental results of three algorithms in Apj



(a) Number of The Roles

(b) WSC

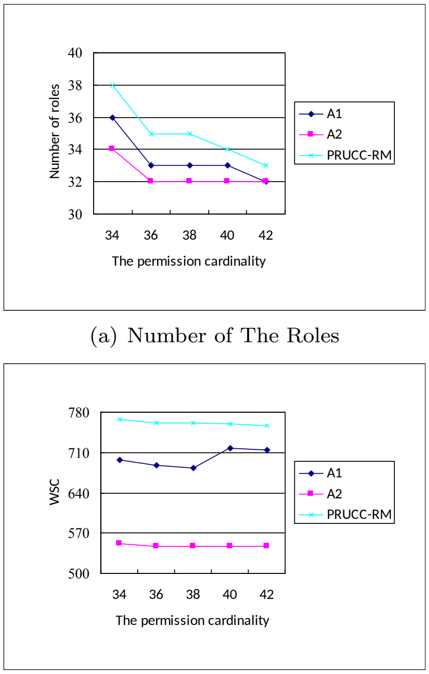
Figure 8: Experimental results of three algorithms in Emea



(a) Number of The Roles

(b) WSC

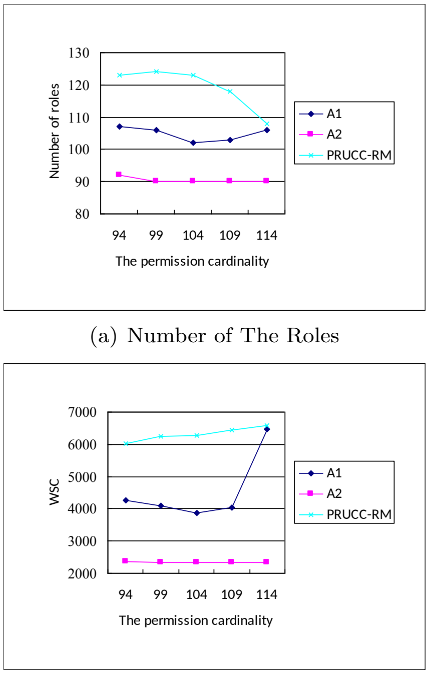
Figure 9: Experimental results of three algorithms in Healthcare



(a) Number of The Roles

(b) WSC

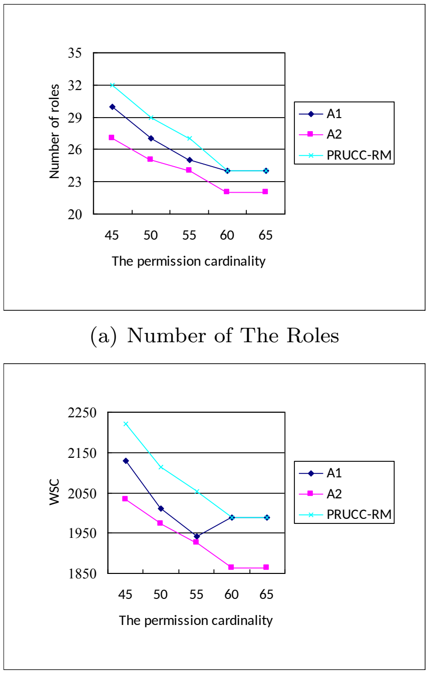
Figure 10: Experimental results of three algorithms in Domino



(a) Number of The Roles

(b) WSC

Figure 11: Experimental results of three algorithms in Firewall1



(a) Number of The Roles

(b) WSC

Figure 12: Experimental results of three algorithms in Firewall2

able permission cardinality is reached, the weighted structural complexity reaches a minimum.The performance of PRUCC-RM is similar to algorithm 1, the biggest difference is that PRUCC-RM has no sequence of rows and columns. The overall tendency of algorithm 2 decreases with the increase of the permission cardinality. Because the permission cardinality is increased, Algorithm 2 can be better mine more frequent permission sets to define as roles, so the overall weighted structural complexity is reduced. Figure 7 and Figure 5 show that the performance of PRUCC-RM is between algorithm 1 and algorithm 2. Because the results produced by PRUCC-RM are related to the location of the permission in UPA, different results are produced when adjusting the position of each permission (*i.e.*, each column) is adjusted in the UPA. Algorithm 1 needs to be sorted by row and column, so it has nothing to do with the position of the permission in UPA.

6 Conclusion

In this paper, we have proposed two algorithms to solve the role mining problem meeting the permission cardianlity constraint in the public data set.Compared with PRUCC-RM. The experimental results demonstrate that the second algorithm performs better than algorithm 1 and PRUCC-RM. However, the algorithm 1 is more simple. The two algorithms proposed in this paper can be applied in the field of frequent item mining. In addition, we can also define the candidate roles with higher frequency as the definitive role, and the roles owned by a small number of users can be managed separately because they are either very important roles or unimportant. Any kind of role mining method has its limitations. Algorithm 2 proposed in this paper is an open role mining method, and the iteration benchmark in the running process of the program can be debugged by the user. The permission cardinality constraint can be adjusted according to the requirements of the number of permissions that the role in the enterprise.Practical experience shows that RBAC is very suitable for systems where the relationship between users and permission does not vary frequently.Users' circumstance and the business process should also be considered after role mining.

Acknowledgments

This study was supported by the National Natural Science Foundation of China (No.61662056) and Inner Mongolia Natural Science Foundation of China (No.2016MS0608, No.2016MS0609). The authors gratefully acknowledge the anonymous reviewers for their valuable comments.

References

[1] S. V. Belim and A. N. Mironenko, "Using the graph-theoretic approach to solving the role mining prob-

lem," in *2018 Dynamics of Systems, Mechanisms and Machines (Dynamics)*, pp. 1–5, Omsk, Russia, Nov 2018.

[2] C. Blundo, S. Cimato, and L. Siniscalchi, "Prucc-RM: Permission-role-usage cardinality constrained role mining," in *IEEE Computer Software and Applications Conference*, pp. 149–154, Torino, Italy, July 2017.

[3] L. Dong, Y. Wang, R. Liu, B. Pi, and L. Wu, "Toward edge minability for role mining in bipartite networks," *Physica A: Statistical Mechanics and its Applications*, vol. 462, pp. 274–286, 2016.

[4] C. Fu, X. Wang, L. Zhang, and L. Qiao, "Mining algorithm for association rules in big data based on hadoop," *AIP Conference Proceedings*, vol. 1955, no. 1, p. 040035, 2018.

[5] P. Harika, M. Nagajyothi, J. C. John, S. Sural, J. Vaidya, and V. Atluri, "Meeting cardinality constraints in role mining," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 1, pp. 71–84, 2015.

[6] H. Huang, S. Feng, J. Liu, and H. Du, "Handling least privilege problem and role mining in rbac," *Journal of Combinatorial Optimization*, vol. 30, no. 1, pp. 63–86, 2015.

[7] J. Jiang, X. Yuan, and R. Mao, "Research on role mining algorithms in rbac," in *Proceedings of the 2018 2nd High Performance Computing and Cluster Technologies Conference*, HPCCT 2018, pp. 1–5, Beijing, China, June 2018.

[8] R. Kumar, S. Sural, and A. Gupta, "Mining rbac roles under cardinality constraint," in *International Conference on Information Systems Security*, pp. 171–185, Gandhinagar, India, Dec 2010.

[9] R. Li, H. Li, W. Wei, X. Ma, and X. Gu, "Rminer: A tool set for role mining," in *Acm Symposium on Access Control Models and Technologies*, pp. 193–196, Amsterdam, The Netherlands, June 2013.

[10] L. Liu, Z. Cao, and C. Mao, "A note on one outsourcing scheme for big data access control in cloud," *International Journal of Electronics and Information Engineering*, vol. 9, pp. 29–35, Sep 2018.

[11] X. Ma, R. Li, H. Wang, and H. Li, "Role mining based on permission cardinality constraint and user cardinality constraint," *Security and Communication Networks*, vol. 8, no. 13, pp. 2317–2328, 2015.

[12] B. Mitra, S. Sural, J. Vaidya, and V. Atluri, "A survey of role mining," *Acm Computing Surveys*, vol. 48, no. 4, pp. 1–37, 2016.

[13] I. Molloy, C. Hong, T. Li, Q. Wang, N. Li, E. Bertino, S. B. Calo, and J. Lobo, "Mining roles with multiple objectives," *Acm Transactions on Information and System Security*, vol. 13, no. 4, pp. 1–35, 2010.

[14] H. T. Pan, C. S. Pan, S. C. Tsaur, and M. S. Hwang, "Cryptanalysis of efficient dynamic id based remote user authentication scheme in multi-server environment using smart card," in *Proceedings - 12th International Conference on Computational Intelligence and Security, CIS 2016*, pp. 590–593, Wuxi, Jiangsu, China, Dec 2016.

[15] U. P. Rao and N. K. Singh, "Weighted role based data dependency approach for intrusion detection in database," *International Journal of Network Security*, vol. 19, no. 3, pp. 358–370, 2017.

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

[17] P. Sarana, A. Roy, S. Sural, J. Vaidya, and V. Atluri, "Role mining in the presence of separation of duty constraints," in *Information Systems Security* (Sushil Jajoda and Chandan Mazumdar, eds.), pp. 98–117, Cham, 2015. Springer International Publishing.

[18] Y. Tian, Y. Peng, G. Gao, and X. Peng, "Role-based access control for body area networks using attribute-based encryption in cloud storage," *International Journal of Network Security*, vol. 19, no. 5, pp. 720–726, 2017.

[19] J. Vaidya, V. Atluri, and G. Qi, "The role mining problem: A formal perspective.," *Acm Transactions on Information and System Security*, vol. 13, no. 3, pp. 1–31, 2010.

[20] J. Wang, J. Liu, and H. Zhang, "Access control based resource allocation in cloud computing environment," *International Journal of Network Security*, vol. 19, no. 2, pp. 236–243, 2017.

[21] W. Ye, R. Li, X. Gu, Y. Li, and K. Wen, "Role mining using answer set programming," *Future Generation Computer Systems*, vol. 55, pp. 336–343, 2016.

Biography

Jingyu Wang is a Ph.D and professor in school of Information Engineering, Inner Mongolia University of Science and Technology, member of the Chinese Computer Society, and a member of the ACM Institute. His main research interests include Big data access control, information security and data mining. Contact him at 13734728816@126.com.

Jingnan Dong is a postgraduate in school of Information Engineering, Inner Mongolia University of Science and Technology,His main research interests include big data, data mining and cloud computing. Contact him at 867324154@qq.com.

Yuesheng Tan is a postgraduate and professor at the School of Information Engineering, Inner Mongolia University of Science and Technology, member of the Chinese Computer Society. His main research interests include high performance computing cloud computing, large-scale data processing and mining. Contact him at 13604729678@139.com.