Abstract
With the rapid development of Internet of Things (IoT) and mobile technologies, the service offerings available in the IoT and mobile environments are increasing dramatically. How to provide more intelligent and personalized services for users becomes a challenging issue. Several context aware service recommendation approaches have been reported to leverage roles to represent common knowledge within user communities, based on which intelligent services can be recommended for users. Prior studies on context aware role mining mainly focus on mining roles from a fixed data set of user behavior patterns, while most of them neglect the dynamic change of the input data. The frequent change of the user data will result in the changing of extracted roles, and how to efficiently update the extracted roles according to the change of the input user data remains a challenging issue. In this paper, towards this issue, we introduce a novel role updating approach of context aware role mining. Experiments show that compared with the algorithm that takes all the updated data as the input, our approach can significantly decrease the update time.

Keywords: role mining, context aware, role updating

1. Introduction
The Internet of Things (IoT) refers to the inter-networking of physical devices, vehicles, buildings and other items embedded with electronics, software, sensors, actuators, and network connectivity that enable these objects to collect and exchange data (ITU, 2016). With the objective of allowing people and things to be connected anytime, anywhere, with anything and anyone, ideally using any path/network and any service (Sundmaeker, 2010), IoT will have a tremendous impact on the way people work and live. With the prevalence of IoT, an increasing number of real-world things will be connected to the IoT environment by exposing their features and capabilities as services, which will lead to a proliferation in services (Huang, 2016). Similarly, as an essential enabling technology of IoT, the popularity of the mobile technology also stimulates the boom of available services in the app stores.

Context-aware communication and computing is a key technology that enables intelligent interactions what the IoT paradigm envisions (Perera, 2014). Services provided by the pervasive and mobile environments of IoT are context-aware in essence. The increasing capability of collecting context information of users, e.g., location, time, and even users’ current status, offers many opportunities for service providers to provide personalized services for their customers, e.g., to make these devices more intelligent and adaptive to user behaviors by automatically recommending suitable services (Adomavicius, 2011; Woerndl, 2009). This is especially important due to the increasing number of services available in the IoT environment.

Most existing service recommendation techniques are based on individual intelligence or the local knowledge (e.g., similar neighbors) of users (Su, 2009), while neglect the common knowledge among the user community. In (Wang, 2012), the notion of "roles" is adopted to represent user groups, which represent the abstract characterization of user behaviors within a certain context. Users can play different roles in their daily life, and their roles change dynamically as the context changes. Users who play the same role are likely to share the same preferences and behavior patterns. Once a user’s role in a given context is identified, the services closely related to the role can be recommended to the user. The service recommendations made by those who play the same role as the user are usually more reliable or acceptable than those who seldom play the role. However, users’ roles are usually not explicitly specified; therefore, how to extract users’ roles from users’ behavior patterns becomes a key issue.

Many works have been conducted to find effective ways of context aware role mining (Wang, 2012; Wong, 2012; Wong, 2014; Hong, 2015). Given a three-dimension matrix <user, context, behavior>, these approaches can be used to generate a user-role assignment matrix as well as a role-behavior assignment matrix. In this way, the latent roles in the dataset can be identified. In these approaches, the input data set is usually fixed. However, in the pervasive IoT environment, the newly coming user behavior patterns and the behaviors of existing users under various contexts may change frequently, which will result in the update of mined roles. Most existing works on role mining neglect the dynamic change of the input data set and how to update the extracted roles when user behavior patterns change have not been sufficiently discussed. Towards this issue, in this paper, we propose a novel approach to update roles by adapting the existing result of role mining rather than re-mining the whole dataset when the user behavior patterns change. Experiments on a simulated data set show that compared with the algorithm that takes all the updated data as the input, our approach can significantly decrease the update time.
time. In addition, compared with a role updating algorithm in (Wong, 2014), ours can guarantee the completeness of the extracted roles in the updating process.

The structure of the paper is organized as follows. Section 2 discusses related works. Then the definition of context-aware role mining and the role updating problem are presented in Section 3. The proposed role updating algorithms are given in Section 4, followed by experimental analysis in Section 5. Section 6 concludes the paper.

2. RELATED WORKS

The rapid growth of Internet of Things and mobile techniques poses many opportunities as well as challenges. Smartphones and the ubiquitous sensors in the IoT environment make it possible to gather users’ information such as users’ behaviors and preferences (Varshney, 2012; Cao, 2010; ITU, 2016). The richness of the context-aware information offers many opportunities to help people in wise decision-making and provide personalized and intelligent services for users.

The concept of roles has been widely used in many areas such as service recommendation, information security, and linguistics (Biddle, 2013; Fukazawa, 2006; Masolo, 2004). RBAC (Role-Based Access Control) (Sandhu, 1996) is an example of the use of roles in information security domain, which is actually the state-of-art standard for access control to sensitive information. Roles are usually manually defined in RBAC. However, as the scale of organizations becomes increasingly larger, as well as the constantly changing personnel, it is increasingly difficult to manually define roles; therefore, many works have attempted to automatically mine roles from large scale of data (Vaidya, 2007; Frank, 2009).

Most existing role mining approaches in the area of RABC only take two dimensional parameters, i.e., <user, permission>, into consideration and neglect the “context”, which is an important parameter in IoT service recommendation. There are only a few papers targeting at the context-aware RABC (Aich, 2009; Kulkarni, 2008), but they did not describe how to discover or identify context-aware roles. As a result, many works started to study context aware role mining and role based recommendation systems. For instance, Wang et al. (Wang, 2012) proposed an approach to mine the minimal set of roles from the matrix of <user, context, behavior>, and created a role tree based on Formal Concept Analysis. Wong et al. (Wong, 2012; Wong, 2014) proposed a more effective context aware role mining approach which can guarantee the completeness of the extracted roles. Hong et al. (Hong, 2015) proposed a context-aware recommendation approach by using role-based trust network. Based on the extracted roles, a role-based trust model is also created to calculate the context-aware trust value between two users.

As the scale of the input of role mining is becoming increasingly larger, it is of vital importance to make sure that the execution time of the mining algorithm is acceptable. Because the input of role mining can change frequently, it is clear that updating roles based on the changed dataset will be more efficient than re-mining the whole dataset. Only limited works have been done on updating the result of role mining when the input changes. For example, the online role mining algorithm proposed by Wong et al. (Wong, 2014) can be used to update roles, but it will leave out some roles during the updating process. Our work is based on the role mining algorithm proposed in that paper. Compared with the role updating algorithm in (Wong, 2014), our approach can guarantee the completeness of the extracted roles in the updating process.

3. PROBLEM DESCRIPTION

In this section, we firstly introduce the concept of role mining, and we then depict the role updating problem in context aware role mining.

The concept role denotes the abstract characterization of user behaviors within a certain context, which can be used to group users that share similar behavior patterns (Wang, 2012). A role is represented as a set of triples <time, scene, behavior>, where time and scene are parts of context. For example, a triple <business hours, on bus, listen to music> belongs to a part of the role “music lover”. The input of role mining is the behavior patterns of multiple users, which are represented as a user-context-behavior matrix (UCB). Given a set of users U, a set of contexts C, and a set of behaviors B, UCB is used to represent which behavior will be taken by a user under certain context, in particular, the value $b_j$ of cell $<u_i$, $c_k>$ in UCB denotes that user $u_i$ has behavior $b_j$ under context $c_k$. An example of UCB is shown in Figure 1(a). Note that 0 in some cells means that the user has no behavior under the context.

\[
\begin{array}{cccc}
\text{c1} & \text{c2} & \text{c3} & \text{c4} \\
\text{u1} & 1 & 0 & 1 & 0 \\
\text{u2} & 1 & 3 & 1 & 0 \\
\text{u3} & 2 & 2 & 1 & 0 \\
\text{u4} & 2 & 2 & 1 & 2 \\
\text{u5} & 2 & 2 & 1 & 3 \\
\end{array}
\]

(a) UCB

\[
\begin{array}{cccc}
\text{c1} & \text{c2} & \text{c3} & \text{c4} \\
\text{r1} & 1 & 0 & 1 & 0 \\
\text{r2} & 2 & 2 & 1 & 0 \\
\end{array}
\]

(c) CBA

\[
\begin{array}{cccc}
\text{r1} & \text{r2} \\
\text{ui} & 1 & 0 \\
\text{ui} & 1 & 0 \\
\text{ui} & 0 & 1 \\
\text{ui} & 0 & 1 \\
\end{array}
\]

(d) UA

Figure 1 Example of role mining from a UCB
The purpose of context-aware role mining is to generate a role-context-behavior assignment matrix (CBA) and a user-role assignment matrix (UA) from the given UCB, where CBA represents the assignment of contexts and behaviors to a group of roles and UA represents how the users are assigned to the roles. CBA should meet the following conditions:

1) the number of contexts assigned to a role in CBA should be at least \( \text{MINC}(\geq 2) \);
2) the number of users assigned to a role in CBA should be at least \( \text{MINU}(\geq 2) \);
3) the user sets of two different roles in CBA should not be totally identical;
and 4) any set of roles which meets the above three conditions is a subset of CBA.

Please note that MINC and MINU can be manually set. Without loss of generality, we set MINC and MINU as 2, respectively, throughout the paper.

Figure 1 shows two roles extracted from UCB and the corresponding CBA and UA. The role mining problem has proven to be a NP-hard problem (Vaidya, 2007); therefore it is of vital importance to minimize the time complexity of the corresponding algorithm. Many prior studies have been proposed towards this direction. However, these works mainly focus on mining roles from a fixed input, and no sufficient attention has been put on the dynamical change of roles when the input data set changes. In particular, this paper mainly considers the following two scenarios, i.e., new users with new behavior patterns are involved in the role mining process, and some existing users are deregistered from the process. In other words, some new rows are added to the original UCB matrix and some existing rows are removed.

4. ROLE UPDATING IN CONTEXT-AWARE ROLE MINING

In this section, we introduce the algorithms of role updating in context aware role mining.

Generally speaking, removing users is easily to achieve. The basic procedure is as follows. We firstly locate the roles that are related to the users to be removed, and then determine whether these roles could be kept (i.e., the number of cells within a role still meets the minimal requirement of a role) after removing the corresponding cells. Therefore, in this paper, we put our focus on the scenario of adding users.

4.1 Existing solutions

Wong et al. (Wong, 2012; Wong, 2014) proposed an effective context aware role mining approach (Algorithm C1), which can completely extract all the potential roles. The basic idea of C1 is to generate role patterns for each user behavior pattern, and compare them to the UCB matrix by matrix subtraction. Clearly, if we use the approach to re-mine the whole updated dataset when the input changes, we can definitely get the complete result of roles. However, given the result of role mining on the original dataset, re-mining the whole updated dataset will be more time-consuming compared with updating the existing roles according to the changed input. Therefore, our goal is to find an updating algorithm to minimize the mining time while keeping the completeness of the mined result. As algorithm C1 can guarantee the completeness of mined result, we can use it as a standard to measure the match rate of the proposed algorithms.

An online role mining algorithm is also proposed in (Wong, 2014), denoted by O1, to handle the role updating problem. The basic idea of Algorithm O1 is summarized as follows. Given an original UCB matrix \( \Delta \text{UCB}_0 \), the new coming users are accumulated to a proper scale, denoted by \( \Delta \text{UCB}_1 \). Afterwards, the existing roles extracted from \( \Delta \text{UCB}_0 \) are updated based on the data of \( \Delta \text{UCB}_1 \), and new roles are extracted from \( \Delta \text{UCB}_1 \). Finally, the results of role mining of \( \Delta \text{UCB}_0 \) and \( \Delta \text{UCB}_1 \), i.e., \( \text{UA}^*0 \), \( \text{UA}^*1 \), \( \text{CBA}^*0 \), and \( \text{CBA}^*1 \) are merged, and the subset roles among them are removed.

Algorithm O1 is simple and fast, nevertheless it cannot guarantee the completeness of potential roles, namely, it may leave out some potential roles. An example is illustrated in Figure 2. As \( u_4 \) and \( u_5 \) cannot constitute a role, algorithm O1 will not find any new role. However, \( u_3 \) and \( u_5 \) can constitute a new role \(<2, 2, 0, 0>\).

![Example scenario where algorithm O1 may leave out some roles](image)

To sum up, when a batch of new users are added into the UCB matrix, the newly generated roles can be classified into two types. One is the roles which are generated solely based on the new users. The other is the roles generated by both the new users and exiting users in the original UCB. Algorithm O1 can find the first type of roles but leave out the roles belonging to the second type.

4.2 Our approach

Based on Algorithm C1 in (Wong, 2014), we propose two role updating algorithms U1 and U2. Both of them are completeness-guaranteed and U2 is an improved version of U1 by optimizing the procedure of removing subset roles.

In Algorithm U1, for each newly coming user, we firstly check whether its behavior pattern is in accord with the roles generated from the original dataset, and add a new row in the UA matrix. We then generate role patterns for each
new user behavior pattern. Let $S = \{<\text{context}, \text{behavior}>\}$ of a user behavior pattern where the behavior is non-zero, role patterns are candidate roles selected from the power set of $S$. As there is a minimal context requirement for a role, the cardinality of a role pattern, i.e., the number of non-zero behaviors within a role pattern, should be at least $\text{MINC}$. Therefore, for each new user behavior pattern, the number of role patterns generated by it should be:

$$N = \sum_{i=\text{MINC}}^{N_1} C_n^i,$$

where $N_1$ refers to the number of non-zero elements of a user behavior pattern.

For example, as shown in Figure 3, given a user behavior pattern $<1, 2, 0, 3, 2>$, $S = \{<c_1,1>, <c_2,2>, <c_3,3>, <c_5,2>\}$ and the power set of $S$: $2^S = \{<c_1,1>, <c_2,2>, <c_3,3>, <c_5,2>, <c_1,1>, <c_2,2>, <c_3,3>, <c_5,2>, <c_1,1>, <c_2,2>, <c_3,3>, <c_5,2>, <c_1,1>, <c_2,2>, <c_3,3>, <c_5,2>, <c_1,1>, <c_2,2>, <c_3,3>, <c_5,2>, <c_1,1>, <c_2,2>, <c_3,3>, <c_5,2>, <c_1,1>, <c_2,2>, <c_3,3>, <c_5,2>, <c_1,1>, <c_2,2>, <c_3,3>, <c_5,2>, <c_1,1>, <c_2,2>, <c_3,3>, <c_5,2>, <c_1,1>, <c_2,2>, <c_3,3>, <c_5,2>, <c_1,1>, <c_2,2>, <c_3,3>, <c_5,2>, <c_1,1>, <c_2,2>, <c_3,3>, <c_5,2>, <c_1,1>, <c_2,2>, <c_3,3>, <c_5,2>, <c_1,1>, <c_2,2>, <c_3,3>, <c_5,2>, <c_1,1>, <c_2,2>, <c_3,3>, <c_5,2>, <c_1,1>, <c_2,2>, <c_3,3>, <c_5,2>, <c_1,1>, <c_2,2>, <c_3,3>, <c_5,2>\}$. Suppose that $\text{MINC}=2$, we can get $N=11$ and the generated role patterns are $rp_1, rp_2, ..., rp_{11}$.

![Figure 3 Example role patterns generated by a user behavior pattern](image)

Some of these generated role patterns may already exist in the CBA matrix generated from the original UCB matrix, namely, they are roles generated from the original dataset. These role patterns should be excluded, and the remaining role patterns are the potential roles. Next, for each potential role, we need to find the user who can be assigned to it. Based on the approaches proposed in (Wong, 2014), we use matrix subtraction to find out the users who can be assigned to each role pattern. If a user matches a certain role pattern, the non-zero behaviors of the user and the role pattern should be identical. Therefore, we extract the columns (i.e., the contexts where the behavior of the role pattern is non-zero) from the original UCB, and construct a new matrix based on the data of these columns. As for the role pattern, we also extract the non-zero behaviors and construct a new vector. We then subtract each row in the new constructed matrix to the new vector. If the result of a row is all zero, i.e., all the cells in the row are zero, then the user corresponding to the row can be assigned to this role pattern. Afterwards, we count the number of the users in the role pattern to determine whether it meets the minimum condition of a role (i.e., larger than $\text{MINU}$). Finally, the new user will be added to the UCB matrix.

![Algorithm U1](image)

In Line 1 of Algorithm U1, $\Delta\text{UCB}$ represents a batch of new users. For each user $user$, the function $\text{addUser}$ in Line 3 is used to find the matched roles of the user and add this information to the UA matrix. More detail of function $\text{addUser}$ is presented in Algorithm U1-addUser. Line 4 finds the role patterns which are generated by $user$ but do not exist in CBA. In Lines 5-15, for each newly generated role pattern, we will find the users who can be assigned to it by using matrix subtraction. $F$ in Line 9 is a matrix that is used to store the result of function $\text{subtraction}$, which selects certain columns (denoted by $position$) out of the UCB matrix to form them into a new matrix and subtracts the variable $rp$ from each row of the new matrix. The reason of adding an element $1'$ to the $userlist$ in Line 15 is that the user who generates this role pattern is not in the UCB matrix at present and this user will be added to the UCB matrix in line 20. In Lines 16-19, we check whether the role pattern meets the minimum requirement of the user number to form a role. If it does, the role pattern will be viewed as a new role and be added to UA and CBA.
Algorithm U1-addUser

Input: UA, CBA, user
Output: UA
initialize newRow;
for i=0 to size(CBA)-1 do
  role = CBA.getRow(i);
  position = find(role);
  flag = TRUE;
  for j=0 to size(position)-1 do
    if user[position[j]] != role[position[j]] then
      flag = FALSE;
      break;
    if flag == TRUE then
      newRow.add(1);
    else
      newRow.add(0);
  UA.addRow(newRow);

In addition, Line 21 in Algorithm U1 removes the subset roles in CBA and UA. Subset roles refer to the redundant roles which are subsets of other roles. For example, for behavior patterns <2, 2, 1, 0, 1> and <2, 2, 1, 1, 1> of two users, <2, 2, 0, 0, 1> is the role formed by these two users and <2, 2, 0, 0, 0>, <2, 0, 0, 0, 1> and <0, 2, 0, 0, 1> are its subset roles. The detailed process of removing the subset roles is presented in Algorithm U1-removeSubsetRoles.

Algorithm U1-removeSubsetRoles

Input: CBA, UA
Output: CBA, UA
for i=0 to size(ΔUCB)-1 do
  if col1 equals col2 then
    if role1 is a subset of role2 then
      delete role1 from CBA;
      delete role1 from UA;
      i--;
      continue;
    else
      if role2 is a subset of role1 then
        delete role2 from CBA;
        delete role2 from UA;
        j--;
      else
        delete role1 from CBA;
        delete role1 from UA;
        i--;
        continue;
  return CBA, UA;

Function removeSubsetRoles traverses the whole UA matrix to make sure that no columns in UA share the same values. Algorithm U1 calls this function to remove the subset roles; however, this step may be redundant, because the original CBA and UA do not contain any subset roles. In other words, function removeSubsetRoles has already been done in the original role mining, and in the updating algorithm we should focus on the newly mined roles and do not need to traverse the whole UA matrix again.

Towards this issue, Algorithm U2 is derived from U1 by reducing the scale of the input of the function removeSubsetRoles. Algorithm U2 adds a new variable checklist to mark which new roles should be taken into consideration in the process of removing subset roles. In Line 4, besides executing what function addUser does, addUser* also adds the indexes of the roles that match with the user to the checklist. In Line 21, the index of the newly mined role is added to the checklist. Then in Line 23, instead of traversing all columns in UA, removeSubsetRoles* only check the columns that the checklist marks and the time complexity is therefore reduced.

Algorithm U2

Input: UCB, UA, CBA, ΔUCB
Output: UCB, UA, CBA
initialize checklist;
for i=0 to size(ΔUCB)-1 do
  user = ΔUCB.getRow(i);
  UA.addUser*(user, CBA, checklist);
  rpArray = rolePattern(user, CBA);
  for j=0 to size(rpArray)-1 do
    rp = rpArray[j];
    position = find(rp);
    initialize userlist;
    F = subtraction(UCB, rp, position);
    for k=0 to size(F)-1 do
      if F.getRow(k) is all zero then
        userlist.add(1);
      else
        userlist.add(0);
      userlist.add(1);
    num = numberOfOnes(userlist);
    if num >= MINU then
      UA.addRole(userlist);
      CBA.addRole(rp);
      checklist.add(size(CBA)-1);
      UCB.addRow(user);
    removeSubsetRoles*(CBA, UA, checklist);
return CBA, UA, UCB;

5. EXPERIMENT ANALYSIS

We conducted a series of experiments to evaluate the performance and accuracy of the proposed algorithms. All algorithms were developed in Java and carried out on a
Table 1 Comparisons among algorithms U2, O1 and C1

<table>
<thead>
<tr>
<th>N(ou)</th>
<th>N(nu)</th>
<th>N(b)</th>
<th>N(c)</th>
<th>Algorithm U2</th>
<th>Algorithm O1</th>
<th>Algorithm C1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Time(Sec.)</td>
<td>Match rate</td>
<td>Time(Sec.)</td>
</tr>
<tr>
<td>500</td>
<td>100</td>
<td>5</td>
<td>5</td>
<td>0.167</td>
<td>1.00</td>
<td>0.027</td>
</tr>
<tr>
<td>500</td>
<td>200</td>
<td>5</td>
<td>5</td>
<td>0.357</td>
<td>1.00</td>
<td>0.084</td>
</tr>
<tr>
<td>500</td>
<td>300</td>
<td>5</td>
<td>5</td>
<td>0.857</td>
<td>1.00</td>
<td>0.184</td>
</tr>
<tr>
<td>500</td>
<td>400</td>
<td>5</td>
<td>5</td>
<td>1.352</td>
<td>1.00</td>
<td>0.328</td>
</tr>
<tr>
<td>500</td>
<td>500</td>
<td>5</td>
<td>5</td>
<td>1.808</td>
<td>1.00</td>
<td>0.623</td>
</tr>
</tbody>
</table>

N(ou): The number of original users; N(nu): The number of new users; N(b): The number of behaviors; N(c): The number of contexts.

As is shown in Figure 4, the execution time of U2 is far less than U1 under different settings of new users, which suggests that the optimization steps of removing subset roles are effective.

Figure 4. Comparison between algorithms U1 and U2

5.1 Comparisons of execution time and match rate

We firstly conducted a series of experiments to compare the execution time and the match rate (the ratio of the number of mined roles to that of all possible roles) among the algorithm that re-mines the whole updated dataset (Algorithm C1), the updating algorithm (Algorithm O1) proposed in (Wong, 2014) and the updating algorithm proposed in this paper (Algorithm U2). The UCB matrices with different sizes are randomly generated. The number of original users is fixed as 500 and the number of new users is increased from 100 to 500.

As shown in Table 1, the execution time of C1 is much longer than those of U2 and O1, which implies that updating algorithms are far more efficient than re-mining the whole updated dataset. The results of the match rate demonstrate that U2 is completeness-guaranteed, while O1 is not. Although the completeness has brought additional complexity and decreased the execution time of U2 compared with algorithm O1, U2 is still much more efficient than C1.

5.2 Execution time of algorithms U1 and U2

Both algorithms U1 and U2 are completeness-guaranteed, while U2 adopted several steps to improve the efficiency of the process in removing subset roles. We conducted experiments to compare the execution time of algorithms U1 and U2 and determine whether the optimization steps are effective.

5.3 Scalability of algorithm U2

The objective of this group of experiments is to analyze the scalability of Algorithm U2 according to its execution time. We analyze it by changing one parameter each time. Considering the fact that the sparsity of the real input data is usually very high, i.e., the frequency of behavior 0 occurs more likely in the UCB matrix, therefore, we further control the percentage of the occurrence of 0 generated in UCB, apart from generating the input data in a completely random manner.

5.3.1 Scalability of U2 when the number of new users changes

It is obvious that the execution time of U2 increases with the increase of the number of new users. In this experiment we fix the number of original users to 500, and the numbers of new users are increased from 1000 to 2000 with step 200. Both the number of contexts and the number of behaviors are set to 5. In order to evaluate the performance of U2
under different data sparsity, we set the percentage of the occurrence of 0 generated in UCB to 30%, 50%, and 70%, besides the completely randomly generated dataset (the percentage of 0 in UCB is 1/6 in this case).

As is shown in Figure 5, there is an almost linear relationship between the number of new users and the execution time of U2. It can also be seen from Figure 5 that as the percentage of behavior 0 in UCB increases, the execution time of U2 decreases significantly.

5.3.2 Scalability of U2 when the number of contexts changes

In this experiment, we evaluate the influence of parameter context on the execution time of U2. The numbers of original users, new users and behaviors are all set to 5, while the number of contexts is increased from 5 to 15. Similar to the experiments mentioned in Subsection 5.3.1, we also control the percentage of the occurrence of 0 generated in UCB.

As can be seen from Figure 6, there is no significant correlation between the number of behaviors and the execution time of U2. The number of behaviors determines how many behaviors can be chosen for users. When the number of behaviors grows larger, the possibility for different users to share a same behavior will decrease, making it harder to constitute roles. This will only influence the number of extracted roles, while the execution time of U2 is not significantly influenced. Similar to the previous experiments, as the percentage of behavior 0 in UCB increases, the execution time of U2 also decreases significantly.

6. CONCLUSIONS

As the input of role mining algorithm is constantly changing, we consider the cases of updating the results of role mining when some
users are deleted and added to the UCB matrix. Experiments show that compared with existing solutions, the proposed algorithms can guarantee the completeness of extracted results while keeping the extraction efficiency.

In the future, we plan to further consider the update of extracted roles when the contexts within the user behavior patterns are changed.

7. ACKNOWLEDGMENT

The work is supported by the National Basic Research Program of China (No. 2014CB340404), and the National Natural Science Foundation of China under Nos. 61672387 and 61373037.

8. REFERENCES


Authors

Zejin Zhu received the B.E. in Information Security from Wuhan University in 2016. He is currently a master student in Computer School, Wuhan University, China. His current research interests include information security and services computing.

Jian Wang received the Ph.D. degree in State Key Lab of Software Engineering, School of Computer, Wuhan University, China in 2008. He is a lecturer in State Key Lab of Software Engineering, Computer School, Wuhan University, China. His current research interests include software engineering and services computing.