Enforcing Cardinality Constraint in Temporal RBAC

Sohail Rajdev[1,†] and Barsha Mitra[2,‡]
[1]*Microsoft, India*
[2]*Department of CSIS, BITS Pilani, Hyderabad Campus, Hyderabad, India*

Keywords: Temporal RBAC, Temporal Role Mining, Cardinality Constraint, Minimization.

Abstract: The temporal extensions of the Role-Based Access Control (RBAC) model imposes duration constraints on the availability of roles by allowing periodic enabling and disabling of roles. For deploying these models, a set of roles having associated time constraints are required. Such type of roles are termed as *temporal roles* and the process of creating them is referred to as *Temporal Role Mining*. In many real-life scenarios, simply imposing time constraints on role availability may not be sufficient. The system administrator may need to ensure different types of constraints reflecting specific organizational policies. In this paper, we propose a cardinality constraint which restricts the maximum number of temporal roles that a user can activate in a particular time interval. We name this constraint as *Temporal Role Assignment Constraint (TRAC)*. We formally define the problem of mining a minimal set of temporal roles in presence of TRAC as the *TRAC Temporal Role Mining Problem (TRAC-TRMP)* and propose an algorithm for solving it. We also present the experimental results reflecting the performance of our proposed approach.

1 INTRODUCTION

Role-Based Access Control (RBAC) model (Sandhu et al., 1996) has been effectively deployed over the years for protecting resources from unauthorized accesses and thus preventing security breaches. For implementing RBAC, a set of roles is required which can be created by an approach known as *role mining* (Ene et al., 2008). RBAC though quite effective, fails to restrict the accessibility of the permissions present in a role to a specific time duration. To handle such requirements, temporally extended RBAC models like the *Temporal Role-Based Access Control (TRBAC)* (Bertino et al., 2001) and the *Generalized Temporal Role-Based Access Control (GTRBAC)* (Joshi et al., 2005) models were proposed. These models allow the roles to be enabled for fixed time intervals during which the permissions associated with the role are available. Moreover, the GTRBAC model introduces the concept of role activation which corresponds to the actual utilization of the permissions by the users. The role activation duration is either equal to or a subset of the enabling duration.

To deploy TRBAC or GTRBAC, a set of roles having appropriate duration constraints are required. Such roles are named as *temporal roles* and the process of generating them is termed as *temporal role mining* (Mitra et al., 2013). Temporal role mining takes a *Temporal User-Permission Assignement (TUPA)* matrix (Mitra et al., 2013) as input. The problem of mining a minimal set of temporal roles has been named as the *Temporal Role Mining Problem (TRMP)* (Mitra et al., 2013). TRMP is an NP-complete problem. Heuristic solutions for solving it has been proposed in (Mitra et al., 2013) and (Mitra et al., 2016). TRMP aims to derive an exact solution where the output perfectly matches the input. Instead, an inexact solution can also be computed such that after assigning the temporal roles to the users, one or more of them may not acquire certain permissions that were originally assigned in the input TUPA. The TRMP variant that generates such an inexact solution has been named as the *Generalized Temporal Role Mining Problem (GTRMP)* (Mitra et al., 2015) and has been shown to be an NP-complete problem. The authors have proposed a greedy algorithm for solving GTRMP in (Mitra et al., 2015). Other than the number of temporal roles, other optimization criteria have also been considered like the *Cumulative Overhead of Temporal Roles and Permissions (CO-TRAP)* (Mi-

[†]This work was carried out by Sohail Rajdev as an undergraduate student of the Dept. of EEE of BITS Pilani, Hyderabad Campus.
[‡]Corresponding Author.

Rajdev, S. and Mitra, B.
Enforcing Cardinality Constraint in Temporal RBAC.
DOI: 10.5220/0010501506390646
In *Proceedings of the 18th International Conference on Security and Cryptography (SECRYPT 2021)*, pages 639-646
ISBN: 978-989-758-524-1

tra et al., 2016) and a *Weighted Structural Complexity (WSC)* (Stoller and Bui, 2017).

Several cardinality constraints have been proposed for the RBAC model like the maximum number of roles that can be assigned to a user (Role-Usage cardinality constraint), the maximum number of roles to which a permission can belong (Permission-Usage cardinality constraint), the maximum number of users to whom a role can be assigned (User-Distribution cardinality constraint)and the maximum number of permissions that can be present in a role (Permission-Distribution cardinality constraint). A number of role mining algorithms have been proposed that consider these constraints such as the ones presented in (Blundo and Cimato, 2012), (Carlo et al., 2018), (Harika et al., 2015), (Hingankar and Sural, 2011), (John et al., 2012), (Lu et al., 2013), (Blundo et al., 2020). The most recent work (Blundo et al., 2020) consider the Permission-Usage cardinality constraint and Permission-Distribution cardinality constraint. Some of these works consider a single cardinality constraint and some of them consider multiple constraints simultaneously. The authors of these papers propose different variants of the role mining problem in presence of one or more of these cardinality constraints and have proven them to be NP-complete. The TRBAC model does not support any cardinality constraints. The GTRBAC model, however, accounts for a number of cardinality constraints like limiting the total number of times a temporal role can be activated as well as the number of simultaneous activations of a role.

In recent years, the Attribute-Based Access Control (ABAC) model (Hu et al., 2015) has become quite popular. This model takes into consideration the attributes of the subjects and the objects and several environmental conditions for granting or denying access. Several algorithms have been proposed for the deployment of the ABAC model like (Xu and Stoller, 2015), (Das et al., 2019), (Batra et al., 2021), (Gupta et al., 2021).

To the best of our knowledge, none of the approaches for temporal role mining consider any cardinality constraint. Moreover, the cardinality constraints for GTRBAC do not consider the workload assigned to the individual users in a certain time interval. In this paper, we propose a new cardinality constraint which limits the number of temporal roles that can be activated by a user in a specific time interval. This constraint will help to uniformly distribute the workload among the users. In addition to this, the proposed constraint is also meaningful from a security point of view. If a user is restricted to activating only a pre-defined number of roles during a certain time interval, then the user can use only a subset of the total set of permissions available to her. By restricting the permissions that can be used during a particular time duration, a user can be prevented from carrying out any malicious activity.

Though the concept of role activation is associated with GTRBAC, the manner in which we handle this constraint makes it applicable for TRBAC as well. We name the proposed cardinality constraint as *Temporal Role Assignment Constraint (TRAC).* We formally define the problem of mining a minimal set of temporal roles in presence of TRAC as the *TRAC Temporal Role Mining Problem (TRAC-TRMP).* We analyze the computational complexity of TRAC-TRMP and present a heuristic algorithm named as *TRAC-Miner* for solving it. Experimental results carried out on synthetic datasets reflect the performance of the proposed approach in terms of the number of roles and the overall execution time.

The rest of the paper is organized as follows. Section 2 discusses some preliminary concepts related to TRBAC, GTRBAC and temporal role mining. Section 3 presents the proposed cardinality constraint TRAC, the associated problem definition for TRAC-TRMP and its complexity analysis. The approach for solving TRAC-TRMP is described in Section 4. Experimental results are presented in Section 5 and Section 6 concludes the paper along with insights into future research work.

2 BACKGROUND

In this section, we discuss some of the basic concepts related to the temporal extensions of RBAC and temporal role mining.

2.1 Temporal RBAC Models

In order to associate time dependent enabling constraints on the availability of roles, two temporal extensions of RBAC were proposed, namely, TRBAC (Bertino et al., 2001) and GTRBAC (Joshi et al., 2005). TRBAC associates duration constraints with roles by allowing them to be enabled for certain time intervals and disabling them for the remaining time intervals. Such roles are referred to as *temporal roles* (Mitra et al., 2013). The enabling duration corresponding to each role is specified in a Role Enabling Base (REB) using a construct known as periodic expression (Bertino et al., 2001). Apart from this, the TRBAC model also introduces the concepts of runtime requests, role triggers and blocked events. However, in the current context, we do not discuss about

these since they are not directly relevant.

The GTRBAC model (Joshi et al., 2005) extends the TRBAC model by introducing temporal constraints on enabling of roles, user-role and role-permission assignments, activation constraints, constraint enabling expressions, temporal role hierarchies and temporal separation of duty constraints. Moreover, the GTRBAC model differentiates between the enabling and the activation of a role. Let a temporal role r be enabled for a set of time intervals T and is assigned to users u and u'. For a time interval $t \in T$, if atleast any one of u and u' activates r, i.e., uses the permissions included in r, then r is said to be activated in t.

Different types of constraints can be imposed on temporal role activation. One such type of constraint is known as *cardinality constraint*. Cardinality constraints on role activation can be of 2 types - total number of activations of a temporal role and maximum number of concurrent activations of a temporal role. The former one specifies the number of times a role can be activated and the latter one restricts the number of concurrent activations of a role.

2.2 Temporal Role Mining

Temporal role mining (Mitra et al., 2013) is the process of creating a set of temporal roles. These roles are essential for the deployment of a temporally extended RBAC model. Temporal role mining takes as input a set of user-permission assignments each of which describes the set of time intervals for which a permission is assigned to a user. Such kind of user-permission assignments are referred to as *temporal user-permission assignments* (Mitra et al., 2013). Mitra et al. have proposed a representation for depicting the temporal user-permission assignments which is referred to as the *Temporal User-Permission Assignment (TUPA) matrix* (Mitra et al., 2013). Each row of the TUPA corresponds to a user and each column corresponds to a permission. If user u_x is assigned permission p_y for a set of time intervals $\mathcal{T}_{xy}$, then the cell of TUPA present in the x-th row and y-th column contains $\mathcal{T}_{xy}$. If however, u_x is not assigned p_y, then the cell contains a ϕ. An example TUPA matrix for 4 users and 5 permissions is shown in Table 1. In this matrix, for the sake of brevity, we have shown only a single interval instead of a set of time intervals for each user-permission assignment. The output of temporal role mining consists of a set of temporal roles, a user-role assignment (UA) matrix, a role-permission assignment (PA) matrix and an REB. The problem of finding a minimal set of temporal roles, a UA, a PA and an REB from an input TUPA has been defined as the *Temporal Role Mining Problem (TRMP)*. Several other variants of TRMP have been defined such as the *Generalized Temporal Role Mining Problem (GTRMP)* (Mitra et al., 2015), *Cumulative Overhead of Temporal Roles And Permissions Minimization Problem (CO-TRAPMP)* (Mitra et al., 2016) and the variant that minimizes a weighted structural complexity (Stoller and Bui, 2017). However, to the best of our knowledge, none of the problems consider any cardinality constraints.

3 MINING TEMPORAL ROLES IN PRESENCE OF CARDINALITY CONSTRAINT

In this section, we introduce the concept of performing temporal role mining in presence of cardinality constraint and formally define the corresponding TRMP variant. We also analyze the computational complexity of the problem.

3.1 Constrained Temporal Role Mining

The cardinality constraints that we have discussed in Sub-section 2.1 ensure that resources are accessible to the requesting users in a fair manner. However, these constraints may not always ensure a uniform distribution of responsibilities among the users. In any organization, it is really important to make sure that some users are not over burdened with responsibilities while others are assigned quite a lesser amount of workload. In other words, to make the work environment user-friendly, it is essential to regulate the number of temporal roles a user can activate simultaneously. This is required to make sure that a user does not juggle too many responsibilities at the same time. Hence, in this paper, we propose a new cardinality constraint which restricts the number of temporal roles that can be activated by each user in a given time interval. We name this constraint as the *Temporal Role Assignment Constraint (TRAC)*. TRAC ensures that in a particular set of time intervals, no user is allowed activate more than a pre-defined number of temporal roles. Consequently, at any point of time, no user is over burdened with too much responsibilities.

In order to enforce TRAC, it is essential to make sure that in a certain time interval, a user does not activate more number of roles than the permissible limit. Ensuring this in real-life scenarios incurs a lot of overhead and effort. Therefore, instead of trying to limit the number of role activations, we can limit the number of roles enabled for a specific set of time intervals

Table 1: Example TUPA Matrix.

	p_1	p_2	p_3	p_4	p_5
u_1	8 am - 9 am	ϕ	7 am - 10 am	8 am - 9 am	7 am - 10 am
u_2	ϕ	4 am - 5 am	8 am - 9 am	4 am - 5 am	7 am - 10 am
u_3	7 am - 10 am	4 am - 5 am	8 am - 9 am	7 am - 10 am	8 am - 9 am
u_4	4 am - 5 am	4 am - 5 am	7 am - 10 am	ϕ	7 am - 10 am

that are assigned to a user. This effectively will also limit the number of roles that a user can activate in a certain time interval. Therefore, for enforcing TRAC, we shall ensure that the number of temporal roles that are assigned to each user for a specific time interval is restricted to a pre-defined value. As a result, TRAC can be seamlessly integrated into TRBAC as well as GTRBAC since TRAC enforcement is not dependent only on role activations. In addition to uniform workload distribution, TRAC will also impose a restriction on the permissions that become available to the users during a specific time duration. By restricting the usable permissions for the users in this manner, it is possible to ensure that not too many permissions become available to the users, thereby preventing the users from carrying out any malicious activity.

Suppose an organization *ABC* has a temporal RBAC (TRBAC or GTRBAC) model deployed where there are a total of 6 roles, r_1, r_2, r_3, r_4, r_5 and r_6 and 3 distinct sets of time intervals T_1, T_2 and T_3. r_1 and r_2 are enabled for T_1, r_3 and r_4 are enabled for T_2 and r_5 and r_6 are enabled for T_3. If the value of TRAC is 1, then a user u will be assigned either one of r_1 and r_2, r_3 and r_4 and r_5 and r_6. As a result, the total number of roles assigned a particular user is also restricted.

3.2 Problem Definition

We name the temporal role mining problem variant that ensures the enforcement of TRAC as the *TRAC Temporal Role Mining Problem (TRAC-TRMP)*. TRAC-TRMP takes a TUPA matrix and an integer value, say k, specifying the value of the proposed cardinality constraint, as inputs. It generates as output a set of temporal roles TR, a UA matrix, a PA matrix and an REB by minimizing the total number of temporal roles and at the same time, ensures that no user is assigned more than k temporal roles enabled for the same set of time intervals. Moreover, the temporal user-permission assignments obtained by combining the UA, PA and REB match exactly with those depicted in the input TUPA. The problem definition is as follows:

TRAC-TRMP. *Given a TUPA matrix and a positive integer k as inputs, find a set of temporal roles TR, a user-role assignment matrix UA, a role-permission assignment matrix PA and a role enabling base REB such that $|TR|$ is minimized, no user is assigned more than k temporal roles that are enabled for the same set of time intervals and the output exactly matches the input.*

Lu et al. have proposed a constrained role mining problem variant known as the *User-Oriented Exact RMP* (Lu et al., 2015) that minimizes the total number of roles and makes sure that no user is assigned more than a pre-specified number of roles. User-Oriented Exact RMP has been shown to be an NP-hard problem. TRAC-TRMP can be shown to be an NP-complete problem by proving that given a solution for TRAC-TRMP, it is polynomial time verifiable and showing that a known NP-hard problem User-Oriented Exact RMP can be reduced to TRAC-TRMP in polynomial time by assuming that only a single set of time intervals is present in the TUPA and the value of the constraint to be a very large positive integer.

4 HEURISTIC ALGORITHM FOR SOLVING TRAC-TRMP

Since TRAC-TRMP is an NP-complete problem, we propose a heuristic algorithm for solving it. Our method makes use of the role mining algorithm proposed in (Lu et al., 2015) for solving User-Oriented Exact RMP. This algorithm minimizes the total number of roles and ensures that no user is assigned more than k roles. We first discuss the steps of the approach proposed in (Lu et al., 2015). The steps are:.

1. The users with the same set of permissions are removed from the input UPA matrix.
2. The users with exclusive permissions are identified. Exclusive permissions are the permissions which are given to only one user. If any user is found having an exclusive permission, a single role for the corresponding permission set is created and assigned to him/her.
3. The remaining user-permission assignments in the UPA matrix are used to create candidate roles for the subsequent steps. A candidate role is created from each row of the UPA.

4. Role selection is done as per a greedy choice. The role which covers the maximum number of users among all the candidate roles is selected.
5. The selected role is assigned to the remaining users according to the following conditions:
   i. If a user is assigned (k - 1) roles after assigning the selected role, the uncovered permissions of the user are checked. A temporary role to cover the set of uncovered permissions is created. It is checked if this role can be reused, i.e., whether this role can be given to some other user.
      a. If the temporary role can be reused, it is assigned to the corresponding users and added to the set of final roles. The UPA matrix is updated by removing the covered permission assignments. The control goes back to Step 3.
      b. If the role cannot be reused, it is discarded and all the roles previously given to the user are revoked. Now, a new single role covering all the permissions of the user is created and assigned to him/her. The UPA matrix is updated by removing the covered permission assignments and control goes back to Step 3.
   ii. If a user has been assigned less than (k - 1) roles after assigning the selected role, the corresponding user-permission assignments are removed from that row of the UPA matrix and the role selection is continued.
6. Steps 3 to 5 are repeated until all user-permission assignments of the UPA have been covered.

The above algorithm uses a strategy of updating the UPA after a role is selected and after each updation, a new set of candidate roles is created. Thus, this approach uses a dynamic set of candidate roles instead of a static one. In the rest of the paper, we will refer to this algorithm as *User-Oriented Miner*.

We name our proposed approach as *TRAC-Miner*. It takes as inputs a TUPA matrix and a value of the constraint expressed as a positive integer. TRAC-Miner works in 2 phases. The first phase identifies the unique time interval sets present in the TUPA and segregates out the temporal information corresponding to each identified set of time intervals. The segregation is done by creating a UPA for each distinct time interval set. After segregation, User-Oriented Miner is applied to each of the individual UPAs along with the value of TRAC and a set of roles, a UA, a PA and an REB are obtained. These roles are then input to the second phase of the algorithm which merges the roles in order to reduce the size of the role set. The final output consists of a set of temporal roles, a UA, a PA and an REB with the UA satisfying the value of TRAC. We next discuss each of the phases in detail.

4.1 Phase 1: Temporal Information Segregation and Mining

We name the first phase of our algorithm as *Temporal Information Segregator and Miner (TIS-Miner*. This phase takes the TUPA and the value of TRAC as inputs. First, the redundancy in the TUPA is removed by deleting the duplicate rows. Then, by scanning the TUPA, all the distinct sets of time intervals are identified. Corresponding to each time interval set, a UPA matrix having the same number of users and permissions as the TUPA is created. We call each such UPA as *Interval-UPA (INT-UPA)*. In order to distinguish the INT-UPAs from one another, INT-UPA created for a time interval set T_l is denoted as $INT\text{-}UPA_{T_l}$. For filling up the cells of $INT\text{-}UPA_{T_l}$, each corresponding cell of the TUPA is scanned and the steps mentioned below are followed.

1. If cell (i, j) of TUPA contains ϕ, then cell (i, j) of $INT\text{-}UPA_{T_l}$ contains zero.
2. If cell (i, j) of TUPA contains T_l, then cell (i, j) of $INT\text{-}UPA_{T_l}$ contains one.
3. If cell (i, j) of TUPA contains a time interval set T_m such that T_l is a subset of T_m, then cell (i, j) of $INT\text{-}UPA_{T_l}$ contains one.
4. If cell (i, j) of TUPA contains a time interval set T_n such that T_l is neither equal to nor a subset of T_n, then cell (i, j) of $INT\text{-}UPA_{T_l}$ contains zero.

After all the INT-UPAs are obtained, the User-Oriented Miner is applied to each INT-UPA to generate a corresponding set of roles, a UA, a PA and an REB. It may be noted here that we use a slightly modified version of User-Oriented Miner in the first phase. Since we remove the redundancy in the TUPA before creating the individual UPAs, we do not apply Step 1 of User-Oriented Miner to the separate UPAs. The roles that are computed from each INT-UPA are enabled for the same time interval set and the REB thus obtained contains the same enabling duration for all the roles. The procedure of the first phase is shown in Algorithm 1.

4.2 Phase 2: Merging Temporal Roles

The sets of temporal roles, UAs, PAs and REBs created in the first phase are input to the second phase. This phase checks the compatibility every pair of roles in order to determine whether the two roles can be merged into a single role. The compatibility checking is done according to the conditions mentioned as follows.

Algorithm 1: TIS-Miner.

Require: **INPUT:** p x q TUPA, **OUTPUT:** sets of temporal roles, UAs, PAs, REBs, S_T: set of all unique time interval sets
Require: $TUPA(i, j)$: cell (i, j) of TUPA
Require: $INT\text{-}UPA(i, j)_{T_l}$: cell (i, j) of INT-UPA for time interval set T_l
Require: R_l: set of roles obtained from $INT\text{-}UPA_{T_l}$
Require: UA_l: UA matrix obtained from $INT\text{-}UPA_{T_l}$
Require: PA_l: PA matrix obtained from $INT\text{-}UPA_{T_l}$
Require: REB_l: REB obtained from $INT\text{-}UPA_{T_l}$
Remove duplicate rows from TUPA
$S_T \leftarrow \phi$
for $i \leftarrow 1$ to p **do**
  for $j \leftarrow 1$ to q **do**
    if $TUPA(i, j) = T$ and $T \notin S_T$ **then**
      $S_T \leftarrow S_T \cup \{T\}$
    end if
  end for
end for
for $l \leftarrow 1$ to $|S_T|$ **do**
  for $i \leftarrow 1$ to p **do**
    for $j \leftarrow 1$ to q **do**
      if $TUPA(i, j) = \phi$ **then**
        $INT\text{-}UPA(i, j)_{T_l} = 0$
      else if $TUPA(i, j) = T_m$ and $T_l \subseteq T_m$ **then**
        $INT\text{-}UPA(i, j)_{T_l} = 1$
      else if $TUPA(i, j) = T_n$ and $T_l \not\subseteq T_n$ **then**
        $INT\text{-}UPA(i, j)_{T_l} = 0$
      end if
    end for
  end for
end for
for $l \leftarrow 1$ to $|S_T|$ **do**
  Apply User-Oriented Miner to $INT\text{-}UPA_{T_l}$ and obtain R_l, UA_l, PA_l and REB_l
end for

1. If two temporal roles are assigned to the same set of users, have the same permission set and are enabled for overlapping or consecutive time interval sets, a new temporal role is created which has the same user and permission sets and enabling duration as the union of time interval sets of the original roles.
2. If two temporal roles are enabled for the same time interval set and have the same user set, a new temporal role is created with the same user set and time interval set and with permission set as the union of permission sets of the original roles.
3. If two temporal roles are enabled for the same time interval set and have the same permission set, a new temporal role is created with the same time interval and permission sets and with user set as the union of user set of individual roles.

In each of the above cases, the new role is added to the final set of temporal roles and the two individual roles are removed from the set. This process of merging of temporal roles was introduced in (Mitra et al., 2013).

4.3 Illustrative Example

We explain the working of TRAC-Miner using the TUPA matrix of Table 1. The TUPA does not contain any duplicate rows. 3 distinct time intervals are present in the TUPA - 8 am to 9 am, 7 am to 10 am and 4 am to 5 am. Time interval 8 am to 9 am is completely contained inside the interval 7 am to 10 am and the interval 4 am to 5 am is disjoint from the other two. Thus, three INT-UPAs, $INT\text{-}UPA_{8\ am-9\ am}$, $INT\text{-}UPA_{7\ am-10\ am}$ and $INT\text{-}UPA_{4\ am-5\ am}$ are created and are shown in Tables 2, 3 and 4 respectively.

We assume the value of TRAC to be 2. First, we process $INT\text{-}UPA_{8\ am-9\ am}$. This UPA does not contain any permission which is exclusively given to only one user. The candidate roles that are created from $INT\text{-}UPA_{8\ am-9\ am}$ are $\{p_1, p_3, p_4, p_5\}$ and $\{p_3, p_5\}$. The first candidate role can be assigned to two users and the second candidate role can be assigned to four users. The second one is selected as per the greedy choice. This role we shall refer to as r_1 and is assigned to all four users. After these role assignments, $INT\text{-}UPA_{8\ am-9\ am}$ is updated by removing the covered user-permission assignments. The updated $INT\text{-}UPA_{8\ am-9\ am}$ contains only four ones in the cells (u_1, p_1), (u_1, p_4), (u_3, p_1) and (u_3, p_4). Only one candidate role $\{p_1, p_4\}$ is created and is assigned to u_1 and u_3. We refer to this role as r_2.

Next, we process $INT\text{-}UPA_{7\ am-10\ am}$. In this UPA, p_1 and p_4 are exclusively given to u_3. Therefore, a role $\{p_1, p_4\}$ is created and is assigned to u_3. We refer to this role as r_3. The modified $INT\text{-}UPA_{7\ am-10\ am}$ now contains ones in cells (u_1, p_3), (u_1, p_5), (u_2, p_5), (u_4, p_3) and (u_4, p_5). From this modified UPA, two candidate roles can be created - $\{p_3, p_5\}$ and $\{p_5\}$. As per the greedy choice, the second candidate role is selected as it can be assigned to three users - u_1, u_2 and u_4. We refer to this role as r_4. After this role assignments, in the modified $INT\text{-}UPA_{7\ am-10\ am}$, the cells (u_1, p_3) and (u_4, p_3) contain one. From this, only a single candidate role $\{p_3\}$ is created and is assigned to u_1 and u_4. This role is referred to as r_5.

While processing $INT\text{-}UPA_{4\ am-5\ am}$, it is found that p_1 and p_4 are exclusively assigned to u_4 and u_2 respectively. Thus, two roles $r_6 = \{p_1, p_2\}$ and $r_7 = \{p_2, p_4\}$ are created, given to u_4 and u_2 respectively and the INT-UPA is updated. After the updation, only one permission assignment (u_3, p_2) remains uncovered. So, a role r_8 containing p_2 is created and given to u_3. Thus, the temporal roles obtained are as follows

Table 2: $INT\text{-}UPA_{8\ am-9\ am}$.

	p_1	p_2	p_3	p_4	p_5
u_1	1	0	1	1	1
u_2	0	0	1	0	1
u_3	1	0	1	1	1
u_4	0	0	1	0	1

Table 3: $INT\text{-}UPA_{7\ am-10\ am}$.

	p_1	p_2	p_3	p_4	p_5
u_1	0	0	1	0	1
u_2	0	0	0	0	1
u_3	1	0	0	1	0
u_4	0	0	1	0	1

Table 4: $INT\text{-}UPA_{4\ am-5\ am}$.

	p_1	p_2	p_3	p_4	p_5
u_1	0	0	0	0	0
u_2	0	1	0	1	0
u_3	0	1	0	0	0
u_4	1	1	0	0	0

Table 5: Experimental Results for TRAC-Miner.

TRAC Value	**Dataset (No. of Users x No. of Permissions)**					
	20 x 20		**50 x 50**		**100 x 100**	
	Roles	*Time (ms.)*	*Roles*	*Time (ms.)*	*Roles*	*Time (ms.)*
1	50.4	5.095	208.7	57.850	490.5	273.919
2	48.6	7.469	205.5	153.031	490.1	2231.087
3	49	7.685	205.6	208.221	490.4	3316.691
4	49.6	7.987	204.4	230.684	490.5	3595.979
5	49.6	8.074	204.5	225.124	490.5	3667.727
6	49.6	8.278	204.4	224.390	490.5	3638.085
7	49.6	7.847	204.4	225.117	490.5	3839.822
8	49.6	7.366	204.4	224.866	490.5	3757.121
9	49.6	7.984	204.4	226.600	490.5	3662.941
10	49.6	7.448	204.4	227.881	490.5	3672.839
11	49.6	7.798	204.4	224.221	490.5	3674.013
12	49.6	7.276	204.4	223.821	490.5	3743.931
13	49.6	7.348	204.4	224.335	490.5	3654.852
14	49.6	7.658	204.4	225.088	490.5	3726.158
15	49.6	8.153	204.4	224.392	490.5	3652.173
16	49.6	7.160	204.4	224.393	490.5	3628.246
17	49.6	8.679	204.4	227.264	490.5	3722.950
18	49.6	8.311	204.4	227.258	490.5	3863.396
19	49.6	7.725	204.4	228.005	490.5	3842.728
20	49.6	8.200	204.4	226.234	490.5	4133.534

- r_1 = ($\{u_1, u_2, u_3, u_4\}$, $\{p_3, p_5\}$, {8 am-9 am}), r_2 = ($\{u_1, u_3\}$, $\{p_1, p_4\}$, {8 am-9 am}), r_3 = ($\{u_3\}$, $\{p_1, p_4\}$, {7 am-10 am}), r_4 = ($\{u_1, u_2, u_4\}$, $\{p_5\}$, {7 am-10 am}), r_5 = ($\{u_1, u_4\}$, $\{p_3\}$, {7 am-10 am}), r_6 = ($\{u_4\}$, $\{p_1, p_2\}$, {4 am-5 am}), r_7 = ($\{u_2\}$, $\{p_2, p_4\}$, {4 am-5 am}) and r_8 = ($\{u_3\}$, $\{p_2\}$, {4 am-5 am}). The merging step does not reduce the size of the role set any further.

5 EXPERIMENTAL RESULTS

For evaluating the performance of our proposed approach, we have synthetically generated three types of TUPA matrices containing 20 users and 20 permissions, 50 users and 50 permissions and 100 users and 100 permissions. All the matrices have a density of 20%, i.e., 20% of all the cells of the TUPA contain non-null entries. We have added the temporal component by associating a time interval with each user-permission assignment chosen from a set of five equi-probable time intervals. The following relationships exist among the time intervals - disjoint, contained, overlapping and consecutive. Since the TUPA matrices are randomly generated, we have created 10 matrices for each type of TUPA. We have varied the value of TRAC from 1 to 20. The algorithm has been implemented using Python and the experiments were carried out on a laptop running macOS 10 and having 2.9 GHz i5 processor and 8 GB RAM. In Table 5, we report the average number of temporal roles and the average execution in milliseconds for the different datasets. The average is computed over the 10 samples for each TUPA type and TRAC value combination.

As can be observed from the table, the number of temporal roles is higher for lower values of the constraint. This is because lower values of constraint imply stricter restrictions which in turn means more number of roles need to be created to satisfy the permission requirements of the user. The execution time is lower for values 1 and 2 of TRAC and is higher for the subsequent values. The lower execution time for constraint values 1 and 2 can be attributed to the fact that role creation for each INT-UPA does not go through all the steps of User-Oriented Miner, but rather through only a few of them. Both the num-

ber of roles and the execution time become constant after a certain value of the constraint which is 6 for our datasets. The reason behind this is, at high values of TRAC, the scenario becomes equivalent to role mining without any constraint enforcement. Also, the number of roles and execution time increases as the size of TUPA increases.

6 CONCLUSION AND FUTURE WORK

In this paper, we have formally defined the problem of mining a minimal set of temporal roles in presence of a cardinality constraint which restricts the maximum number of roles that each user can activate in a specific set of time intervals. We have shown this problem to be NP-complete and have proposed a heuristic algorithm to solve it. The proposed approach works in two phases, the first phase using an existing constrained role mining algorithm to mine the time interval specific UPAs and the second phase merging the temporal roles obtained from the first phase.

In future, we intend to design temporal role mining methods capable of enforcing the various cardinality constraints proposed for the GTRBAC model either in isolation or in combination. Also, it will be interesting to investigate how role mining metrics other than the number of temporal roles can be minimized in presence of these constraints. Moreover, the enforcement of cardinality constraints in presence of an administrative temporal RBAC model can be a possible future direction of research.

REFERENCES

Batra, G., Atluri, V., Vaidya, J., and Sural, S. (2021). Incremental maintenance of abac policies. In *11th ACM Conference on Data and Application Security and Privacy*, page 185 – 196.

Bertino, E., Bonatti, P. A., and Ferrari, E. (2001). TRBAC: A Temporal Role-Based Access Control Model. *ACM Trans. on Info. and Sys. Security*, 4(3):191–233.

Blundo, C. and Cimato, S. (2012). Constrained role mining. In *Proc. of 8th Int. Workshop on Security and Trust Management*, pages 289–304.

Blundo, C., Cimato, S., and Siniscalchi, L. (2020). Managing constraints in role based access control. *IEEE Access*, 8:140497–140511.

Carlo, B., Stelvio, C., and Luisa, S. (2018). Postprocessing in constrained role mining. In *Int. Conf. on Intelligent Data Engineering and Automated Learning*, pages 204–214.

Das, S., Sural, S., Vaidya, J., Atluri, V., and Rigoll, G. (2019). Vismap: Visual mining of attribute-based access control policies. In *Int. Conf. on Information Systems Security*, pages 79–98.

Ene, A., Horne, W., Milosavljevic, N., Rao, P., Schreiber, R., and Tarjan, R. E. (2008). Fast exact and heuristic methods for role minimization problems. In *Proc. of 13th ACM Symposium on Access Control Models and Technologies*, pages 1–10.

Gupta, E., Sural, S., Vaidya, J., and Atluri, V. (2021). Attribute-based access control for nosql databases. In *11th ACM Conference on Data and Application Security and Privacy*, pages 317 – 319.

Harika, P., Nagajyothi, M., John, J. C., Sural, S., Vaidya, J., and Atluri, V. (2015). Meeting cardinality constraints in role mining. *IEEE Trans. on Dependable and Secure Computing*, 12(1):71–84.

Hingankar, M. and Sural, S. (2011). Towards role mining with restricted user-role assignment. In *Proc. of 2nd Int. Conf. on Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology*, pages 1–5.

Hu, V. C., Kuhn, D. R., and Ferraiolo, D. F. (2015). Attribute-Based Access Control. *Computer (IEEE)*, 48(2):85–88.

John, J. C., Sural, S., Atluri, V., and Vaidya, J. (2012). Role mining under role-usage cardinality constraint. In *Proc. of 27th Int. Info. Security and Privacy Conf.*, pages 150–161.

Joshi, J. B. D., Bertino, E., Latif, U., and Ghafoor, A. (2005). A Generalized Temporal Role-Based Access Control Model. *IEEE Trans. on Knowledge and Data Engg.*, 17(1):4–23.

Lu, H., Hong, Y., Yang, Y., Duan, L., and Badar, N. (2013). Towards user-oriented RBAC model. In *Proc. of 27th Int. Conf. on Data and Applications Security and Privacy*, pages 81–96.

Lu, H., Hong, Y., Yang, Y., Duan, L., and Badar, N. (2015). Towards user-oriented RBAC model. *J. of Comp. Security*, 23(1):107–129.

Mitra, B., Sural, S., Atluri, V., and Vaidya, J. (2013). Toward mining of temporal roles. In *Proc. of 27th Conf. on Data and Applications Security and Privacy*, pages 65–80.

Mitra, B., Sural, S., Atluri, V., and Vaidya, J. (2015). The generalized temporal role mining problem. *J. of Comp. Security*, 23(1):31–58.

Mitra, B., Sural, S., Vaidya, J., and Atluri, V. (2016). Mining temporal roles using many-valued concepts. *Computers & Security*, 60:79 – 94.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. (1996). Role-Based Access Control Models. *IEEE Computer*, 29(2):38–47.

Stoller, S. and Bui, T. (2017). Mining hierarchical temporal roles with multiple metrics. *J. of Comp. Security*, 26(1):121–142.

Xu, Z. and Stoller, S. (2015). Mining Attribute-Based Access Control policies. *IEEE Transactions on Dependable and Secure Computing*, 12(5):533–545.