Meeting Cardinality Constraints in Role Mining

Pullamsetty Harika, Marreddy Nagajyothi, John C. John, Shamik Sural, Jaideep Vaidya, and Vijayalakshmi Atluri

Abstract—Role mining is a critical step for organizations that migrate from traditional access control mechanisms to role based access control (RBAC). Additional constraints may be imposed while generating roles from a given user-permission assignment relation. In this paper we consider two such constraints which are the dual of each other. A role-usage cardinality constraint limits the maximum number of roles any user can have. Its dual, the permission-distribution cardinality constraint, limits the maximum number of roles to which a permission can belong. These two constraints impose mutually contradictory requirements on user to role and role to permission assignments. An attempt to satisfy one of the constraints may result in a violation of the other. We show that the constrained role mining problem is NP-Complete and present heuristic solutions. Two distinct frameworks are presented in this paper. In the first approach, roles are initially mined without taking the constraints into account. The user-role and role-permission assignments are then checked for constraint violation in a post-processing step, and appropriately re-assigned, if necessary. In the second approach, constraints are enforced during the process of role mining. The methods are first applied on problems that consider the two constraints individually, and then with both considered together. Both methods are evaluated over a number of real-world data sets.

Index Terms—RBAC, role mining, cardinality constraint, concurrent framework, post-processing framework

1 Introduction

In traditional access control mechanisms, a user accesses a resource through direct permission given on that resource. In organizations with tens of thousands of users and permissions, the number of user-permission assignments becomes very large, making security administration quite challenging. Over the last few years, there is an increasing trend of using role based access control (RBAC) [11], [12]. In RBAC, permissions are assigned to roles. Users obtain permissions by acquiring the required roles. Since the number of roles is significantly smaller than the number of permissions, RBAC makes security administration more manageable and flexible.

Defining suitable roles is, however, essential for organizations that desire to migrate from traditional access control mechanisms to RBAC. Role engineering is the process of defining appropriate roles. Top down, bottom up or hybrid approaches can be followed for role engineering [9], [16]. In the top down approach, roles are defined based on the different business processes and job functions in the organization. This approach is difficult to implement when large number of business processes and job functions are involved. Also, it is not amenable to automation. On the other hand, a bottom-up approach uses the existing user-permission assignments to formulate roles in an automated way. The user-permission assignment relation UPA is represented as a binary matrix denoted by $M(UPA)$, where rows represent users, columns represent permissions, and each cell value (0 or 1) indicates the permission status of the corresponding user. An automated procedure for formulating roles from the UPA relation is known as role mining [7], [13], [17].

In role mining, roles are generated as a set of permissions and they are appropriately assigned to users. The output of the role mining process is represented in the form of two binary matrices, $M(UA)$ (Matrix representation of user to role assignment UA) and $M(PA)$ (Matrix representation of permission to role assignment PA). The primary goal is to obtain a correct decomposition, i.e., $M(UA) \otimes M(PA) = M(UPA)$ where $\otimes$ denotes Boolean matrix multiplication [7], [8]. In the rest of the paper, we use UA (respectively PA) to denote both the user-assignment (respectively permission-assignment) relation as well as its matrix representation. Beyond correct decomposition, it is often required to obtain a decomposition with the least number of roles. Minimizing the number of roles derived from a given UPA, thereby forming an optimal decomposition into UA and PA is known as the basic role mining problem (RMP) [7], which has been shown to be NP-hard.

The importance of incorporating various constraints in RBAC has been articulated in a number of seminal papers [11], [12]. Constraints like separation of duty, pre-requisite and cardinality constraints [11] are essential in implementing organizational policies on the security of a system. Usually, every cardinality constraint imposed on the UA relation has a dual constraint for the PA relation [11]. A restriction is often imposed on the maximum number of roles that can be assigned to any user, either for enforcing the principle of least privilege or for balanced work distribution. The constraint is termed as the *role-usage cardinality constraint* [15].

The problem of role mining under role-usage cardinality constraint has been addressed in [15] using optimal Boolean matrix decomposition [8]. The work in [19] proposes a role mining algorithm that restricts the number of

- *P. Harika, M. Nagajyothi, J.C. John, and S. Sural are with the School of Information Technology, Indian Institute of Technology, Kharagpur, India. E-mail: {pharika, marreddyn, johncjohn, shamik}@sit.iitkgp.ernet.in.*
- *J. Vaidya and V. Atluri are with the Department of MSIS, Rutgers University, New Jersey. E-mail: jsvaidya@business.rutgers.edu, atluri@rutgers.edu.*

Manuscript received 3 May 2013; revised 18 Jan. 2014; accepted 23 Jan. 2014. Date of publication 27 Feb. 2014; date of current version 16 Jan. 2015.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TDSC.2014.2309117

permissions in a role. However, it is ineffective in controlling the distribution of powerful permissions since there is no restriction on the number of roles to which a permission can belong. Such a constraint, which is the dual of the role-usage constraint, is denoted as the *permission-distribution cardinality constraint*.

All prior work so far only considers role mining with a single constraint at a time. Organizations may also impose multiple constraints on roles simultaneously. Our work extends the current state-of-the-art by incorporating both RBAC constraints discussed above together. Note that, in certain situations, enforcing one constraint can lead to creation of new roles, which in turn violates the other constraint. Thus, enforcing one constraint may preclude enforcement of the other. While this paper primarily focuses on the problem of role mining in the simultaneous presence of two cardinality constraints, we also include brief discussions on the individual constraints in order to introduce the design principles of our algorithm.

Our main contribution is two alternative frameworks for role mining in the presence of role-usage and permission-distribution cardinality constraints. In the first framework, called the *post-processing framework*, roles are initially mined without considering the constraints, using any known role mining algorithm. Once the UA and PA decomposition is obtained, it is checked whether each of the constraints is satisfied. If not, we try to fix the cases where the constraints are violated. Care is taken so that this post-processing step does not introduce further violations. On the other hand, in the second framework, called the *concurrent processing framework*, we impose the constraints during the process of role mining.

The rest of this paper is organized as follows. In Section 2, we formally introduce the different problems addressed in this paper. Sections 3 and 4 describe the post processing framework and the concurrent processing framework, respectively. In Section 5, we experimentally validate our approaches on real data sets and compare against alternatives. Section 6 discusses how the proposed algorithms can complement existing approaches in role mining. We review related work in Section 7. Finally, Section 8 concludes the paper.

2 Constrained Role Mining Problem

We address three problems on constrained role mining. These are: role mining under role-usage cardinality constraint, role mining under permission-distribution cardinality constraint and role mining under simultaneous application of both. We now formally define the problems and study their complexity.

The same notations as used in [8] and [15] are also used here. Let there be n users and m permissions with q roles, the Boolean matrices UA, PA and UPA can be represented as $C_{n\times q}$, $R_{q\times m}$ and $X_{n\times m}$, respectively. Let c_{ij} be the UA matrix entry for user i $(i = 1, \dots, n)$ and role j $(j = 1, \dots, q)$, let r_{jt} be the PA matrix entry for role j $(j = 1, \dots, q)$ and permission t $(t = 1, \dots, m)$ and let x_{it} be the UPA matrix entry for user i and permission t. Then the following conditions must be satisfied for any consistent decomposition [8], [15]: $\sum_{j=1}^{q} c_{ij}r_{jt} \geq 1$ for $x_{it} = 1$ and $\sum_{j=1}^{q} c_{ij}r_{jt} = 0$ for $x_{it} = 0$.

We now formally define the problem of role mining under the constraint that there is an upper limit on the number of roles to which a user can belong. We assume that the constraint value is the same for all users.

Definition 1. *Role-usage cardinality constraint problem (RUP). Given $X_{n\times m}$, find a consistent decomposition $C_{n\times q}$ and $R_{q\times m}$ which minimizes the value of q under the condition $\sum_{j=1}^{q} c_{ij} \leq MRC_{user}$ for all $1 \leq i \leq n$, where MRC_{user} (maximum role constraint on user) is the maximum number of roles assigned to any user.*

We next define the problem of role mining under the constraint that there is an upper limit on the number of roles to which a permission can be assigned. We assume that the constraint value is the same for all permissions.

Definition 2. *Permission-distribution cardinality constraint problem (PDP). Given $X_{n\times m}$, find a consistent decomposition $C_{n\times q}$ and $R_{q\times m}$ which minimizes the value of q under the condition $\sum_{j=1}^{q} r_{jt} \leq MRC_{perm}$ for all $1 \leq t \leq m$, where MRC_{perm} (maximum role constraint on permission) is the maximum number of roles to which any permission can be assigned.*

Finally, we define the problem of role mining when the two constraints described above are applied simultaneously. We assume that the role-usage cardinality constraint value is the same for all users and the permission-distribution cardinality value is the same for all permissions, though the two values need not be the same.

Definition 3. *Multiple cardinality constraint problem (MCP). Given $X_{n\times m}$, find a correct decomposition $C_{n\times q}$ and $R_{q\times m}$ which minimizes the value of q under the conditions $\sum_{j=1}^{q} c_{ij} \leq MRC_{user}$ for all $1 \leq i \leq n$, and $\sum_{j=1}^{q} r_{jt} \leq MRC_{perm}$ for all $1 \leq t \leq m$.*

We next define a decision version of this problem.

Definition 4. *Decision multiple cardinality constraint problem (Decision MCP). Given $X_{n\times m}$, $MRC_{user} > 0$, $MRC_{permission} > 0$ and $k > 0$, is there a set of decompositions $C_{n\times q}$ and $R_{q\times m}$ such that $q \leq k$ and $\sum_{j=1}^{q} c_{ij} \leq MRC_{user}$ for all $1 \leq i \leq n$, and $\sum_{j=1}^{q} r_{jt} \leq MRC_{permission}$ for all $1 \leq t \leq m$?*

RUP, PDP and MCP are all NP-complete. The proofs for RUP and PDP are given in the supplement. Below, we give the proof for MCP. For ease of understanding, we first state the decision version of the Basic (i.e., Unconstrained) Role Mining Problem (Basic RMP) introduced in [7], [8].

Definition 5. *Decision role mining problem (Decision RMP). Given $X_{n\times m}$ and $k > 0$, is there a set of decompositions $C_{n\times q}$ and $R_{q\times m}$ such that $q \leq k$?*

Decision RMP is known to be NP-complete [7].

Theorem 1. *Decision-MCP is NP-complete.*

Proof. Decision RMP can be reduced to Decision MCP by defining $MRC_{user} \geq k$ and $MRC_{permission} \geq k$. Since it is a direct one-to-one mapping, the reduction takes polynomial time. The matrices X, C and R together form a certificate, which can be verified in polynomial time. Since, Decision RMP is NP-complete and it can be polynomially reduced to Decision MCP, Decision MCP is also NP-complete. □

Having defined the three problems, we next proceed to present two different approaches for solving each.

3 Post Processing Framework

In this framework, we start by considering an un-constrained decomposition of a given UPA into UA and PA, which has been obtained by applying any of the existing role mining techniques.

The UA and PA are first checked for violation of the given cardinality constraint(s). If there is no violation, they are simply returned as a valid solution. Otherwise, a post-processing step is used to fix the cases of constraint violation. Depending on the constraint being handled (i.e., role-usage, permission-distribution or both), different algorithms are used as described below. The general principle followed is that violations are not allowed to further propagate as constraints are fixed.

An intermediate step can be introduced to refine the UA and PA generated by the unconstrained role mining algorithm before post-processing. This is useful when roles with skewed distributions of users and permissions are generated by the unconstrained algorithm, e.g., roles that contain only one permission or one user. There may also be redundant roles and redundant user-role assignments. In this work, we do not consider any such intermediate phase. However, our algorithm can seamlessly handle situations when such a step is introduced for refinement of user-role and role-permission assignments following unconstrained role mining.

Note that when the constraints are considered one at a time, the corresponding algorithms are guaranteed to return a UA and PA satisfying the given constraint and consistent with the original UPA. However, when both kinds of constraints are simultaneously considered, the algorithm may terminate without finding any consistent decomposition that satisfies all constraints.

3.1 Fixing Role-Usage Cardinality Constraint

We denote the value of the constraint as MRC_{user}. $UserRoleCount[u]$ is the number of roles possessed by user u and $RoleUserCount[r]$ is the number of users belonging to role r. Users whose number of roles exceeds MRC_{user} are first identified from the UA matrix. For each violating user, the algorithm creates a new role by merging some of his existing roles, removes assignments to the merged roles, and adds assignments to the newly created role. K roles out of the roles currently possessed by the user are chosen for merging where $K = UserRoleCount[u] - (MRC_{user} - 1)$. In other words, $MRC_{user} - 1$ number of roles of the user are retained and the permissions of the remaining roles are merged to form a single role. Those K roles of the current user are chosen which are currently assigned to the maximum number of users in the UA matrix, i.e., the roles having the top K maximum $RoleUserCount$ values are chosen. This particular heuristic is used since the new role thus formed can be used to replace the set of merged roles for all the concerned users. As a result, for some of these users, the desired constraint can be fixed without forming new roles, with a possible reduction in the total number of roles finally generated.

The steps of the algorithm are summarized in the *Fix_Role_Usage_Constraint* algorithm (Algorithm 1). Merging of roles and creation of new roles are done in Lines 6-8. The newly created role is added to both UA and PA matrices (Lines 9-23).

Algorithm 1 Fix_Role_Usage_Constraint

1: Required: UA and PA matrices, MRC_{user}
2: Compute $UserRoleCount[]$, $RoleUserCount[]$
3: Compute $ViolationCount$: No. of users in UA with $UserRoleCount[] > MRC_{user}$
4: **while** $ViolationCount > 0$ **do**
5: Choose a violating user u based on a heuristic
6: $K = UserRoleCount[u] - (MRC_{user} - 1)$
7: Choose K roles of u with highest $RoleUserCount$ values to form set S
8: Merge the permissions of the roles in S to form a new role $newrole$ {Add $newrole$ to both UA and PA matrices}
9: **for** each user $i \in U$ **do**
10: **if** $UA[i][r] = 1, \forall r \in S$ **then**
11: $UA[i][r] = 0, \forall r \in S$
12: $UA[i][newrole] = 1$
13: **else**
14: $UA[i][newrole] = 0$
15: **end if**
16: **end for**
17: **for** each permission $p \in P$ **do**
18: **if** $\exists r \in S$ such that $PA[r][p] = 1$ **then**
19: $PA[newrole][p] = 1$
20: **else**
21: $PA[newrole][p] = 0$
22: **end if**
23: **end for**
24: Update $ViolationCount$, $UserRoleCount[u]$ for all $u \in U$, $RoleUserCount[r]$ for all $r \in R$
25: **end while**

3.2 Fixing Permission-Distribution Cardinality

This algorithm is a dual of the *Fix_Role_Usage_Constraint* algorithm. The value of the constraint is denoted by MRC_{perm}. $PermRoleCount[p]$ is the number of roles to which permission p belongs and $RolePermCount[r]$ is the number of permissions assigned to role r.

Analogous to *Fix_Role_Usage_Constraint*, in this algorithm, each of the *permissions* violating the constraint in the PA matrix is selected one at a time. The roles having the top K (where $K = PermRoleCount[p] - (MRC_{perm} - 1)$) maximum $RolePermCount$ values are chosen. However, at the time of forming a new role, the *intersection* of the permissions of the chosen K roles is considered. Unlike the role-usage cardinality constraint case, only the users who have *any* of the selected set of roles can be assigned to the newly created role.

3.3 Simultaneous Fixing of Cardinality Constraints

We next study the problem of fixing the role-usage and permission-distribution cardinality constraints applied simultaneously on the UA and PA decomposition of a given UPA matrix. The proposed algorithm combines the algorithms for fixing individual constraints in a meaningful way. However, the process of merging a number of roles to form a new role and assigning the same to the respective users and permissions may lead to assignment of more number of roles to a permission, resulting in a new violation in permission-distribution constraint. Similarly, intersecting a number of roles to form a new role and assigning the same to respective permissions and users may lead to a new violation in role-usage constraint. In the proposed algorithm, we prevent such proliferation of violations beyond the ones present in the given UA and PA matrices in order to ensure that the algorithm always terminates.

It may be noted that, two situations could occur where the proposed algorithm does not give a solution (i.e., final UA and PA matrices) in which both the constraints are fixed. In the first case, the constraint values could be such that no valid decomposition of the given UPA matrix into UA and PA matrices exists for which the constraints can be satisfied. The second situation is where such a solution exists, yet the algorithm fails to provide the same. This could occur due to the fact that we start with a given decomposition and only try to fix the constraint violations without making global changes. The chosen heuristic for selecting the next violating user or permission attempts to reduce this scenario.

The sets of users and permissions violating the respective MRC_{user} and MRC_{perm} constraints are first determined. In each step of the algorithm, either a violating user or a violating permission is selected following a greedy choice. K number or roles possessed by u or p are chosen. If a violating user is chosen, $K = UserRoleCount[u] - (MRC_{user} - 1)$ whereas if a violating permission is chosen, $K = PermRoleCount[p] - (MRC_{perm} - 1)$. In this work, we studied the effect of four different heuristics, namely, (i) choosing the user or permission with the minimum number of violations (Min), (ii) choosing the user or permission with the maximum number of violations (Max), (iii) sequentially choosing users, then permissions (UP) and (iv) sequentially choosing permissions, then users (PU).

A new role r could be created by merging (if a violating user is chosen in the current step) or intersection (if a violating permission is chosen in the current step) of the K roles as done in the single constraint algorithms. However, merger/intersecting leads to an increase in the role counts of the permissions/users belonging to r, which, in turn, may cause a violation of the cardinality constraint on permission/user. Hence, the roles have to be selected in such a way that their merger/intersection operation does not lead to any new violation of constraint. In order to achieve this, a role set RU containing roles that would not cause any new violation on the permission side by a single merger of roles and re-assignment on the user side is maintained. Similarly, another role set RI containing roles that would not cause any violation on the user side due to a single intersection and re-assignment on the permission side is maintained. The problem of propagation of constraint violation is handled by selecting the K roles possessed by u from the set RU or by selecting K roles to which p is assigned from the set RI. Role selection is done similar to that for single constraints.

The detailed steps of the algorithm are given in Algorithm 2. Initially, the sets RU and RI are computed. In each step of the iteration, either a user or a permission violating their respective constraints is selected using a greedy choice. The selected user or permission undergoes the step of role merger/intersection in which the top K number of roles are chosen as mentioned in the previous sections. The variable *ExceptionCount* represents the number of constraints that could not be fixed so far. It is incremented by one every time an attempt to fix a user or a permission fails. Its value is reset to 0 whenever a violation of user or permission constraint is fixed. This is done to handle situations in which fixing one user or permission violation through creation of a new role could change the role allocation of a user or permission for which it was earlier determined that the constraint violation could not be fixed. Hence, the algorithm re-considers these previously failed cases. The algorithm terminates when all the constraint violations are fixed or if it is determined that a total of *ExceptionCount* of constraint violations cannot be fixed.

Algorithm 2 Fix_Role_Usage_and_Permission_Distribution_Constraints

Required: UA and PA matrices, MRC_{user}, MRC_{perm}
Compute $UserRoleCount[]$, $PermRoleCount[]$, $RoleUserCount[]$, $RolePermCount[]$, RU, RI
$ExceptionCount \leftarrow 0$
Compute $UserViolationCount$: No. of users in UA with $UserRoleCount > MRC_{user}$, $PermViolationCount$: No. of permissions in PA with $PermRoleCount > MRC_{perm}$
while $ExceptionCount < (UserViolationCount + PermViolationCount)$ **do**
  Choose a violating user u or a violating permission p based on a heuristic {Excludes a user or permission whose violation has already been fixed.}
  if a user is chosen **then**
    $K = UserRoleCount[u] - (MRC_{user} - 1)$
    Choose K roles of u from RU with highest $RoleUserCount$ values to form set S
    Merge the permissions of the roles in S to form a new role $newrole$
  else
    $K = PermRoleCount[p] - (MRC_{perm} - 1)$
    Choose K roles of p from RI with highest $RolePermCount$ values to form set S.
    Intersect the permissions of the roles in S to form a new role $newrole$
  end if
  if new role formed **then**
    Add $newrole$ to both UA and PA matrices
    Update $UserRoleCount[u]$ for all $u \in U$, $PermRoleCount[p]$ for all $p \in P$
    Update RU, RI and for all $r \in R$ $RoleUserCount[r]$, $RolePermCount[r]$
    Update $UserViolationCount$ and $PermViolationCount$ {The number of violations gets reduced}
    $ExceptionCount = 0$
  else
    $ExceptionCount{+}{+}$
  end if
end while
if $UserViolationCount \neq 0$ OR $PermViolationCount \neq 0$ **then**
  "The given set of constraints cannot be fixed."
end if

4 Concurrent Processing Framework

In the concurrent processing framework, a given UPA is decomposed into UA and PA while minimizing the number of roles and considering the constraints $during$ the process of decomposition itself. As a result, the obtained UA and PA have the constraints satisfied by construction. The basic approach for decomposition could be based on any of the existing algorithms for unconstrained role mining like [7], [8]. In this paper, we use a constrained version of the minimum biclique cover (MBC) based approach proposed in [10]. We first consider the role-usage and permission-distribution constraints individually and then consider them simultaneously.

4.1 Enforcing Role-Usage Cardinality Constraint

We now present our proposed approach for handling role-usage cardinality constraint based on a bipartite graph representation of the user-permission assignment.

The work in [2], [10] maps a UPA matrix into an undirected bipartite graph $G(V, E)$ in which the vertex set V is partitioned into two disjoint subsets U and P, where elements of U represent users and the elements of P represent permissions. The edge set E consists of pairs (u, p) where $u \in U$ and $p \in P$ only if user u is granted permission p in the given UPA matrix. Then the users and permissions included in a role form a biclique in G. The basic role minimization problem can, therefore, be mapped to the problem of finding a minimum biclique cover of the edges of the bipartite graph so constructed. Selection of the next biclique is based on a greedy heuristic. The work in [2], [10] pointed out that by selecting vertices with the minimum number of uncovered incident edges, one can get better result.

To enforce role-usage cardinality constraint, we count the number of roles to which a user u belongs at any iteration (denoted as $UserRoleCount[u]$). Our goal is to limit $UserRoleCount[u], \forall u$ to MRC_{user}, the cardinality constraint value. The approach selects only user vertices rather than both user and permission vertices as done in [10]. In each step, attempt is made to cover the most number of possible uncovered incident edges of the selected vertex.

The steps of the *Enforce_Role_Usage_Constraint* are given in Algorithm 3. Here $V[u]$ represents the set of end vertices (permissions) of incident edges of user u, $UC[u]$ represents the set of end vertices (permissions) of uncovered edges of user u, P represents the set of selected permissions and U represents the set of selected users. The algorithm works in two phases. In Phase 1 (Lines 3-20), only users with uncovered incident edges whose $UserRoleCount < MRC_{user} - 1$ are taken into consideration. Among these users, the algorithm selects a user u based on the chosen heuristic (Line 5). It then increments $UserRoleCount[u]$ by 1 (Line 6) and sets $UC[u]$ to P (Line 7). Next, it finds all the other users v for which vertices in $V[v]$ are supersets of U (Line 9 or Line 13). The user v can be added into the set U if either $UserRoleCount[v]$ is less than $MRC_{user} - 1$ and a permission of at least one element of $UC[v]$ belongs to P (Line 9), or if $UserRoleCount[v]$ is equal to $MRC_{user} - 1$ and $UC[v] \subseteq P$ (Line 13). It adds all such v into the set U (Line 10 or Line 14) and increments $UserRoleCount[v]$ by 1 for all v (Line 11 or Line 15). A biclique is then formed with sets U and P (Line 19). The process is repeated until all the users with uncovered incident edges and $UserRoleCount < MRC_{user} - 1$ are considered.

After Phase 1, some users with $UserRoleCount = MRC_{user} - 1$ may still be left with uncovered incident edges. In this case, Phase 2 is executed (Lines 21-33). Here, one of the pending users u with the maximum number of uncovered incident edges is selected. Other users v are found if $P \subseteq V[v]$ and $UC[v] \subseteq P$ and $UserRoleCount[v] = MRC_{user} - 1$ (Line 27). Bicliques are formed as done in Phase 1 (Lines 28-32). The steps are repeated till all the pending users have been assigned to bicliques to cover all of their incident edges.

4.2 Enforcing Permission-Distribution Cardinality

The *Enforce_Permission_Distribution_Constraint* algorithm enforces permission-distribution cardinality constraint. With every permission p, we associate a count denoted as $PermRoleCount[p]$, which represents the number of roles to which a permission p belongs at any step of the iteration. The goal of the algorithm is to limit $PermRoleCount[p], \forall p$ to MRC_{perm}, the maximum number of roles to which a permission can belong. It also runs in two phases, which are similar to that of Algorithm 3. However, only permission vertices are selected as the starting vertex in each iteration rather than user vertices.

Algorithm 3 Enforce_Role_Usage_Constraint

1: Set $UserRoleCount[u] = 0$, for all users u
2: Determine $UC[u]$, for all users u
{PHASE 1}
3: **while** there is at least one user u with uncovered incident edges AND $UserRoleCount[u] < MRC_{user} - 1$ **do**
4: Set $U = \phi$, $P = \phi$
5: Select the next user u using a suitable heuristic and add u into U
6: $UserRoleCount[u] = UserRoleCount[u] + 1$
7: Set $P = UC[u]$
8: **for** each user $v \neq u$ **do**
9: **if** $P \subseteq V[v]$ AND at least one element of $UC[v]$ is an element of P AND $UserRoleCount[v] < MRC_{user} - 1$ **then**
10: Add v into U
11: $UserRoleCount[v] = UserRoleCount[v] + 1$
12: **else**
13: **if** $P \subseteq V[v]$ AND $UC[v] \subseteq P$ AND $UserRoleCount[v] = MRC_{user} - 1$ **then**
14: Add v into U
15: $UserRoleCount[v] = UserRoleCount[v] + 1$
16: **end if**
17: **end if**
18: **end for**
19: Form a biclique with U and P
20: **end while**
{PHASE 2}
21: **while** there is at least one user u with uncovered incident edges AND $RoleCount[u] = MRC_{user} - 1$ **do**
22: Set $U = \phi$, $P = \phi$
23: Select the user u with maximum number of uncovered incident edges and add u into U
24: $UserRoleCount[u] = UserRoleCount[u] + 1$
25: Set $P = UC[u]$
26: **for** each user $v \neq u$ **do**
27: **if** $P \subseteq V[v]$ AND $UC[v] \subseteq P$ AND $UserRoleCount[v] = MRC_{user} - 1$ **then**
28: Add v into U
29: $UserRoleCount[v] = UserRoleCount[v] + 1$
30: **end if**
31: **end for**
32: Form a biclique with U and P
33: **end while**

4.3 Simultaneous Constraint Enforcement

The algorithm for enforcing both role-usage and permission-distribution cardinality constraints effectively combines the two algorithms presented in the last two sections. At each step of Phase 1, either a user or a permission vertex can be selected using a suitable heuristic as described below. An end vertex is selected only if its current $RoleCount$ ($UserRoleCount$ or $PermRoleCount$ depending on whether the chosen vertex is a user vertex or a permission vertex) value is less than $(MRC - 1)$ (Here MRC to be considered is either MRC_{user} or MRC_{perm} depending on whether the chosen vertex represents a user or a permission).

In Phase 2, in each iteration, a user or a permission is selected that has the maximum number of uncovered incident edges. A new role with the selected vertex is formed if and only if all of its end vertices have $RoleCount$ values less than or equal to $(MRC - 1)$. The algorithm terminates either with every edge covered using at least one biclique with no vertex assigned to more than MRC number of bicliques or if one or more vertex exists whose incident edges cannot be covered by bicliques without causing the vertex to belong to more than MRC number of bicliques.

In order to study the effectiveness of the algorithm, i.e., its ability to determine valid user-role and role-permission assignments while honoring the given constraints over a wide range of constraint values, a number of heuristics for choosing the next vertex for biclique formation (Line 7 of Algorithm 4) were tried. These are (i) Node with the minimum number of uncovered incident edges giving priority to user (NU), i.e., if there exist a user and a permission with the same minimum number of uncovered edges, choosing a user vertex is preferred over a permission vertex. (ii) Node with the minimum number of uncovered incident edges giving priority to permission (NP), i.e., if there exist a user and a permission with the same minimum number of uncovered edges, choosing a permission vertex is preferred over a user vertex. (iii) Choosing the node with the maximum number of roles that can yet be assigned (XR), i.e., a user or a permission is chosen that has the maximum number of roles left to reach the constraint value (equivalently, the vertex with the minimum role count). In case of conflict, the one with the minimum number of uncovered edges is chosen. (iv) Choosing the node with the minimum number of roles left (NR), i.e., a user or a permission is chosen that has the minimum number of roles left to reach the constraint value. In case of conflict, the one with the minimum number of uncovered edges is chosen.

Algorithm 4 Enforce_Role_Usage_and_Permission_Distribution_Constraints

1: Required: MRC_{user}, MRC_{perm}, UPA Matrix
2: Set $UserRoleCount[u] = 0, \forall u \in U$
3: Set $PermRoleCount[p] = 0, \forall p \in P$
4: U represents the set of selected users and P represents the set of selected permissions to form a role
{PHASE 1}
5: **for** each user u with uncovered incident edges AND $UserRoleCount[u] < MRC_{user} - 1$ OR Permission p with uncovered incident edges AND $PermRoleCount[p] < MRC_{perm} - 1$ **do**
6: Set $U = \phi$, $P = \phi$
7: Select a vertex v using a suitable heuristic
8: **if** the selected vertex is a user **then**
9: Call $Form\_Role$ procedure (Algorithm 5)
10: **else**
11: Call $Dual\ of\ Form\_Role$ procedure
12: **end if**
13: **end for**
{PHASE 2}
14: **for** each user u with uncovered incident edges AND $UserRoleCount[u] = MRC_{user} - 1$ OR permission p with uncovered incident edges AND $PermRoleCount[p] = MRC_{perm} - 1$ **do**
15: Set $U = \phi$, $P = \phi$
16: Select the vertex v with the maximum number of uncovered incident edges
17: **if** the selected vertex is a user **then**
18: Set $P = UC[v]$
19: **if** $PermRoleCount[p] \leq MRC_{perm} - 1, \forall p \in P$ **then**
20: Call $Form\_Role$ procedure (Algorithm 5)
21: **else**
22: Call $Dual\ of\ Form\_Role$ procedure
23: **end if**
24: **end if**
25: **end for**
26: **if** there is at least one vertex with uncovered edges **then**
27: "The given set of constraints cannot be enforced"
28: **end if**

Two other heuristics, which are variants of (iii) and (iv) above, were also considered, where in case of conflict, the vertex with the maximum number of uncovered edges was chosen rather than the one with the minimum. However, these did not show any improvement.

The steps are summarized in Algorithm 4. Here MRC_{perm} and MRC_{user} represent the maximum number of roles to which a permission or a user can belong. $UC[u]$ and $UC[p]$ represent the set of uncovered end vertices of the respective user or permission. $V[u]$ and V[p] represent the set of incident end vertices of the respective user and permission. If a user vertex is selected, a new role is formed using the *Form_Role* procedure of Algorithm 5. On the other hand, if a permission vertex is selected, the new role is formed using a dual of the *Form_Role* procedure (not separately shown here).

Algorithm 5 Form_Role procedure

Add v to U
$UserRoleCount[v] = UserRoleCount[v] + 1$
for each $p \in UC[v]$ AND $PermRoleCount[p] < MRCperm - 1$ **do**
Add p to P
$PermRoleCount[p] = PermRoleCount[p] + 1$
end for
for each user $u \neq v$ **do**
if $P \in V[u]$ AND at least one element of $UC[u]$ is an element of P AND $UserRoleCount[u] < MRC_{user} - 1$ **then**
Add u into U
$UserRoleCount[u] = UserRoleCount[u] + 1$
else
if $P \subseteq V[u]$ AND $UC[u] \subseteq P$ AND $UserRoleCount[u] = MRC_{user} - 1$ **then**
Add u into U
$UserRoleCount[u] = UserRoleCount[u] + 1$
end if
end if
end for
Form a biclique with U and P

5 Experimental Evaluation

We now describe the experimental evaluation. The algorithms proposed in this paper have been implemented in C on a 3.1 GHz Intel i5-2400 CPU having 4GB RAM. Experiments were carried out on nine real-world data sets [10], [14], as shown in Table 1. It may be noted that these data sets have been widely used in the literature for studying the performance of unconstrained role mining algorithms. Use of the same data sets facilitates evaluation of the impact of various constraints on the number of roles generated by our proposed approaches.

Each data set represents a UPA matrix. In the concurrent processing framework, the UPA matrix serves as the input to the algorithm. In the post processing framework, an unconstrained decomposition of the UPA matrix into UA and PA matrices is required as the input. We apply the minimum biclique cover based algorithm [10] to decompose the given UPA. The number of roles obtained in this unconstrained decomposition are also shown in Table 1, which match with that reported in the literature [10]. The corresponding execution time is also included.

TABLE 1
Data Set Description

Data set	Users	Permissions	Roles	Execution time (s)
Americas-large	3485	10127	421	94.360
Americas-small	3477	1587	211	3.640
Apj	2044	1164	456	3.640
Customer	10961	284	276	2.100
Domino	79	231	20	0.002
EMEA	35	3046	34	0.012
Firewall1	365	709	69	0.066
Firewall2	325	590	10	0.012
Healthcare	46	46	15	0.001

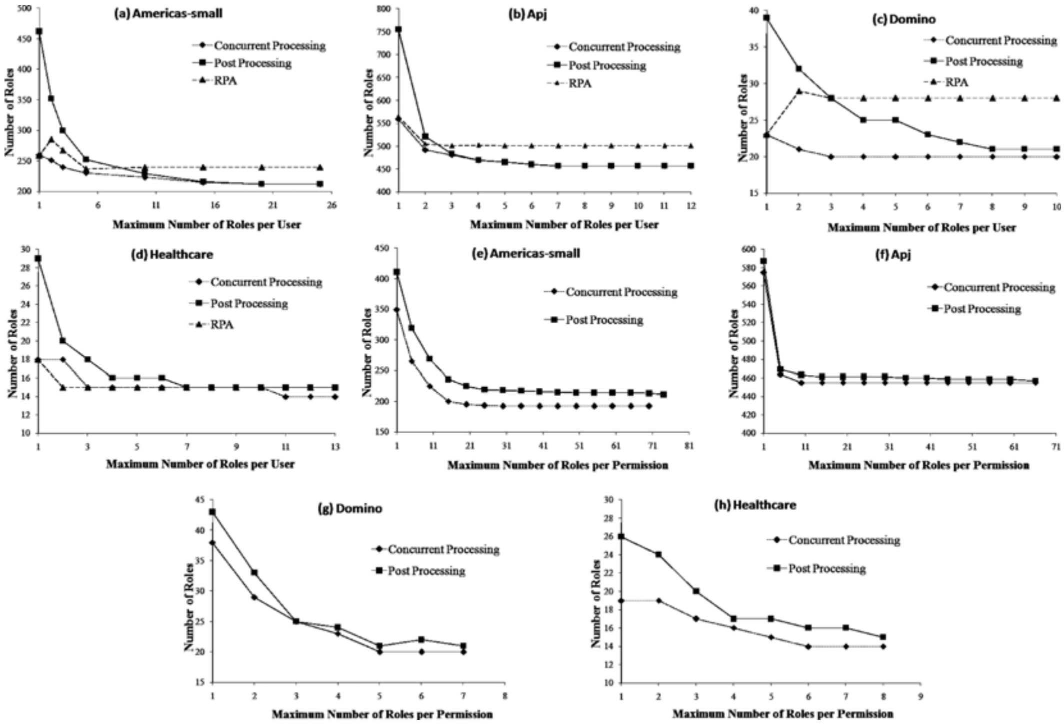


Fig. 1. Effect of role-usage (a-d) and permission-distribution (e-h) constraints on the number of generated roles.

We first study the impact of a single constraint on the number of roles generated using the two approaches (Section 5.1). This is followed by Section 5.2 in which a comparative study is made between the four heuristics of each of the two proposed approaches in terms of the number of roles generated. A comparison of the execution time of the two best approaches, one from post-processing and one from the concurrent framework, is also included in this section. Finally, in Section 5.3, we do a comparative study with the graph optimization based algorithm proposed in [9].

5.1 Role-Usage Cardinality Constraint

Figs. 1a, 1b, 1c, and 1d show the variation in the number of roles generated for the Americas-small, Apj, Domino and Healthcare data sets by the post-processing and the concurrent processing frameworks as the value of the cardinality constraint MRC_{user} is varied. For comparison, the results of a recently proposed algorithm named as role priority based approach (RPA) [15] based on Boolean matrix decomposition [8] have also been shown in the same set of figures. We start with a constraint value of 1 and increase it till there is no further change in the number of roles generated. This corresponds to the point at which the constraint value equals the maximum number of roles possessed by any user in the unconstrained decomposition of the respective UPA.

The results show that the number of roles generated by both pre-processing and concurrent approaches increase as the constraint value is lowered. This is intuitively expected since for a lower value of the number of roles allowed per user, more permissions of each user need to be combined together to generate unique user specific roles. There is little scope for sharing the same role across multiple users, thereby increasing the overall total number of roles required for a consistent decomposition.

Another important observation from the figures is that, the concurrent processing framework usually generates less number of roles than the post-processing framework. This is because the post-processing framework can fix the violations by only re-assigning the permissions of the violating users. New roles are formed without removing the existing ones while fixing the constraint unless there are other users sharing all the permissions of a role newly created. There is no flexibility to make global changes. On the other hand, the concurrent framework at each step enforces the constraints by making a greedy choice among all the users remaining to be assigned to roles. As a result, the number of generated roles is less. It is also seen from Figs. 1a, 1b, 1c, and 1d that both post-processing and concurrent processing approaches proposed in this paper generate less number of roles compared to RPA for all the four data sets.

The impact of the MRC_{perm} value on the number of roles generated for role mining under permission-distribution cardinality constraint is shown in Figs. 1e, 1f, 1g, and 1h. The results follow a similar trend as in the case of role-usage cardinality constraint.

5.2 Results in the Presence of Two Constraints

We now present results when both Role-usage and Permission-distribution constraints are specified simultaneously. Since the two constraints impose mutually contradictory requirements, certain combinations of MRC_{user} and MRC_{perm} can result in no valid decomposition being found satisfying both constraints. However, anti-monotone property holds meaning that if for a given combination of MRC_{user} and MRC_{perm}, no solution can be found, for any combination with lower values of either or both of the

TABLE 2
Number of Roles Generated for Post-Processing and Concurrent Approaches Using Different Heuristics

(a) Americas-large

MRC_{perm}	MRC_{user}																															
	6								5								4								3							
	Min	Max	UP	PU	NU	NP	XR	NR	Min	Max	UP	PU	NU	NP	XR	NR	Min	Max	UP	PU	NU	NP	XR	NR	Min	Max	UP	PU	NU	NP	XR	NR
144	421	421	421	421	423	421	435	521	422	422	422	422	423	421	435	513	424	424	424	424	424	421	435	503	x	x	x	x	420	417	437	489
140	423	422	423	423	423	421	435	521	424	423	424	424	423	421	435	513	426	425	426	426	424	421	435	503	x	x	x	x	420	417	437	489
130	423	422	423	423	423	421	435	521	424	423	424	424	423	421	435	513	426	425	426	426	424	421	435	503	x	x	x	x	420	417	437	489
120	426	422	424	424	424	422	435	521	427	423	425	425	424	422	435	513	429	425	427	427	425	422	435	503	x	x	x	x	x	x	437	489
110	430	422	426	426	424	421	435	521	431	423	427	427	424	421	x	513	433	425	429	429	425	421	x	503	x	x	x	x	x	x	x	489
100	434	427	430	430	427	423	435	521	435	428	431	431	426	422	x	513	437	431	433	433	426	420	x	503	x	x	x	x	x	x	x	x

(b) Apj

MRC_{perm}	MRC_{user}																															
	11								10								8								6							
	Min	Max	UP	PU	NU	NP	XR	NR	Min	Max	UP	PU	NU	NP	XR	NR	Min	Max	UP	PU	NU	NP	XR	NR	Min	Max	UP	PU	NU	NP	XR	NR
67	456	456	456	456	456	457	454	484	457	457	457	457	456	457	454	484	459	457	458	458	458	462	454	484	463	461	465	465	461	x	460	484
65	457	457	457	457	456	457	454	484	458	458	458	458	456	x	454	484	460	458	459	459	458	x	454	484	464	462	466	466	x	x	460	484
55	460	458	459	459	456	457	455	484	461	459	460	460	456	x	455	484	463	461	463	461	x	x	x	484	x	x	471	468	x	x	x	484
45	463	459	460	460	x	x	x	484	464	460	461	461	x	x	x	484	467	464	466	462	x	x	x	484	x	x	x	x	x	x	x	484
35	463	460	460	460	x	x	x	484	464	461	462	462	x	x	x	484	468	x	x	x	x	x	x	484	x	x	x	x	x	x	x	484
25	463	460	461	461	x	x	x	484	464	461	462	462	x	x	x	484	468	x	x	x	x	x	x	484	x	x	x	x	x	x	x	484
15	463	x	x	x	x	x	x	484	464	x	x	x	x	x	x	484	x	x	x	x	x	x	x	484	x	x	x	x	x	x	x	484
5	x	x	x	x	x	x	x	482	x	x	x	x	x	x	x	482	x	x	x	x	x	x	x	482	x	x	x	x	x	x	x	x

(c) Firewall1

MRC_{perm}	MRC_{user}																															
	21								16								12								8							
	Min	Max	UP	PU	NU	NP	XR	NR	Min	Max	UP	PU	NU	NP	XR	NR	Min	Max	UP	PU	NU	NP	XR	NR	Min	Max	UP	PU	NU	NP	XR	NR
26	69	69	69	69	69	69	65	69	70	70	70	70	69	69	65	69	71	71	71	71	69	69	66	69	75	72	73	73	x	x	70	69
25	70	70	70	70	70	70	65	69	71	71	71	71	70	70	65	69	72	72	72	72	70	70	66	69	76	76	74	74	x	x	70	69
21	71	70	71	71	69	69	65	69	72	71	72	72	69	69	65	69	73	72	73	73	69	69	66	69	78	x	75	75	x	x	70	69
17	75	72	73	73	69	69	65	69	76	73	74	74	69	69	65	69	77	74	75	75	69	69	66	69	x	x	x	x	x	x	x	69
13	76	73	75	75	69	69	65	69	77	74	76	76	69	69	65	69	78	75	77	77	69	69	66	69	x	x	x	x	x	x	x	69
9	76	74	76	76	69	69	65	69	77	75	77	77	69	69	65	69	78	x	80	81	69	69	66	69	x	x	x	x	x	x	x	69

(d) Firewall2

MRC_{perm}	MRC_{user}																															
	9								8								7								6							
	Min	Max	UP	PU	NU	NP	XR	NR	Min	Max	UP	PU	NU	NP	XR	NR	Min	Max	UP	PU	NU	NP	XR	NR	Min	Max	UP	PU	NU	NP	XR	NR
3	10	10	10	10	10	10	10	12	11	11	11	11	10	10	10	12	11	11	11	11	10	10	10	12	x	x	x	x	x	x	x	12
2	x	x	x	x	10	10	10	11	x	x	x	x	10	10	10	11	x	x	x	x	x	x	10	11	x	x	x	x	x	x	x	11

Post-processing heuristics: Min, Max, UP and PU; Concurrent heuristics: NU, NP, XR and NR.

constraints, no solution would exist. Similarly, monotonicity property also holds, i.e., if a solution exists for a given combination of MRC_{user} and MRC_{perm}, a solution would exist for any combination with higher values of either or both of the constraints.

In order to decide which combinations of MRC_{user} and MRC_{perm} should be considered for experiments, from the unconstrained decomposition of the UPA, the maximum number of roles any user had in the resulting UA and the maximum number of roles to which any permission belonged in the PA were determined. If these values are considered to be the values of MRC_{user} and MRC_{perm}, respectively, then by construction, a valid solution is known to exist for the given UPA. The constraint values were incrementally lowered starting from this initial combination. We stop when no solution is found for a given combination. By the anti-monotone property mentioned above, no solution is expected to exist for smaller values of the constraints.

5.2.1 Variation in the Number of Roles Generated

Table 2 shows the number of roles generated using the post-processing and the concurrent processing frameworks on four of the nine real world data sets. Since the EMEA data set has at most one role for each user and the customer data set has at most one role for each permission, these two data sets cannot be meaningfully used for studying the presence of the two constraints together and hence, are excluded from this set of results. Due to space limitation, we do not report the results for Americas-small, Domino and Healthcare. The complete results are given in the supplemental text. For the post-processing framework, we show results for the four heuristics mentioned in Section 3.3 and used in Algorithm 2, namely, *Min*, *Max*, *UP* and *PU*. Similarly, for the concurrent framework, results are presented for the four heuristics mentioned in Section 4.3 and used in Algorithm 4, namely, *NU*, *NP*, *XR* and *NR*. The top left set of values of the number of roles in Tables 2a, 2b, 2c, and 2d correspond to the cardinality constraint values for which there is no constraint violation for the users as well as permissions. The fact that these values match closely with those reported in Table 1 in the column named "Roles" shows that the proposed method works correctly and the constraint handling step built into the role mining algorithm does not affect the performance of the algorithm when users do not specify any constraint (equivalent to specifying high values of the constraints).

MRC_{user} is varied along the columns and MRC_{perm} is varied along the rows in the tables. A "$\times$" in any cell indicates that no valid decomposition could be obtained for that combination of constraint values using the corresponding algorithm.

In Tables 2a, 2b, 2c, and 2d, it is observed that, for any given data set and a chosen heuristic, the number of roles increases or remains unchanged in the post-processing framework as the values of the two constraints are reduced. While fixing one violation at a time in each step of the iteration, the four heuristics choose the next violation in different ways. Since there is no backtracking, the algorithm after fixing some of the violations could be left with a subset of violations that cannot be fixed. A different sequence of choices in the earlier stages (by a different heuristic) might not have yielded such a state and all the violations could have been fixed when the algorithm terminated. It is seen from the results presented in the tables that the *Min* heuristic, i.e., choosing the user or permission with the minimum number of violations at any iteration step of Algorithm 2, has better performance as compared to the other heuristics in terms of the ability to fix constraints for smaller values of MRC_{user} and MRC_{perm}. The number of roles generated by this heuristic is also comparable with the other ones.

Similarly, Tables 2a, 2b, 2c, and 2d also show that the number of roles usually increases or remains unchanged in the concurrent framework as the values of the two constraints are reduced. It is also seen from the tables that the three heuristics *NU*, *NP* and *XR* have similar performance in terms of the number of roles generated as well as the ability to handle lower values of constraints. Although the number of roles generated by the *NR* heuristic is higher than the rest, it has much better performance in terms of the ability to fix constraints for smaller values of MRC_{user} and MRC_{perm}. For example, for the Firewall1 and Firewall2 data sets, it can enforce constraints for all the combinations shown in the tables. For Americas-large and Apj, it can handle all the combinations except the last one in each.

It is further observed that the three heuristics *NU*, *NP* and *XR* of the concurrent approach, in general, have better performance compared to all the four heuristics of the post-processing approach in terms of the number of roles generated. This is similar to the observation made in the previous section where only one constraint was considered at a time. Further, the post-processing approach fails to provide a solution to certain combinations of constraints for which the solution exists as is evidenced by the results from the concurrent approach. For example, none of the heuristics of the post-processing approach can provide a solution for the Americas-large data set for all values of MRC_{perm} when $MRC_{user} = 3$. All the heuristics of the concurrent approach can provide a solution till $MRC_{perm} = 130$ for $MRC_{user} = 3$. The poorer performance of the post-processing approach is due to the fact that it starts with the given UA and PA matrix and only tries to fix the violations without bringing in any new violation.

Also note that in the concurrent processing case, sometimes the number of roles decreases even though the constraint values are reduced. The original UPA matrices for all the data sets were analyzed with respect to the distribution of the number of permissions over users and that of users over permissions. It was observed that, there are clusters of such coverage numbers. For example, in the Americas-large data set, there are 20 permissions each of which has 2,804 users. There is an abrupt drop after this with the permission having the next highest number of users has 178 users. There are three permissions with 178, 165 and 164 users, followed by two with 162 users each. This is once again followed by 11 permissions each with 161 users. When there is a tie, the concurrent processing algorithm chooses one of the alternatives at random, which causes occasional decrease in the number of roles even if the values of the constraints are reduced. Since, such types of abruptness do not exist in case of the post-processing approach which starts with the UA and PA matrices already obtained from an initial decomposition, it does not show such unexpected variations.

TABLE 3
Execution Time

(a) Americas-large (in sec.)

MRC_{perm}	MRC_{user} 6		5		4		3	
	p	c	p	c	p	c	p	c
144	0.06	194.9	0.12	188.5	0.23	188.2	x	180.6
140	0.17	196.0	0.23	187.6	0.34	185.0	x	181.1
130	0.18	195.4	0.23	187.4	0.35	185.7	x	179.9
120	0.35	195.9	0.4	189.4	0.51	185.5	x	182.8
110	0.59	194.5	0.63	190.7	0.74	187.0	x	181.6
100	0.81	198.1	0.85	189.4	0.95	184.6	x	x

(b) Apj (in sec.)

MRC_{perm}	MRC_{user} 11		10		8		6	
	p	c	p	c	p	c	p	c
67	0.02	11.8	0.03	11.9	0.07	11.9	0.12	11.9
65	0.04	12.0	0.05	11.7	0.08	11.7	0.14	11.9
55	0.09	12.1	0.09	11.7	0.13	11.7	x	11.7
45	0.13	11.9	0.14	11.7	0.19	11.7	x	11.7
35	0.13	12.0	0.14	11.6	0.2	11.7	x	11.7
25	0.13	12.0	0.14	11.7	0.2	11.6	x	11.7
15	0.13	11.9	0.14	11.7	x	11.7	x	11.7
5	x	12.0	x	11.9	x	11.7	x	11.6

(c) Firewall1 (in msec.)

MRC_{perm}	MRC_{user} 21		16		12		8	
	p	c	p	c	p	c	p	c
26	1.22	192.7	2.52	192.3	3.77	193.2	8.70	192.2
25	2.46	193.6	3.80	187.7	5.01	186.8	9.95	187.3
21	3.81	193.4	2.55	186.4	6.23	188.3	8.77	187.5
17	8.86	192.5	10.2	179.8	11.4	188.8	x	190.1
13	10.2	193.1	11.6	193.5	12.3	193.4	x	186.9
9	10.1	193.0	8.83	189.8	12.6	187.8	x	187.9

(d) Firewall2 (in msec.)

MRC_{perm}	MRC_{user} 9		8		7		6	
	p	c	p	c	p	c	p	c
3	0.14	31.4	0.37	31.5	0.38	31.4	x	31.5
2	x	30.7	x	28.0	x	28.0	x	27.7

Based on the above observations, we select the *Min* heuristic of the post-processing approach and the *NR* heuristic of the concurrent approach for further evaluation in the next sections.

5.2.2 Variation in Execution Time

In Tables 3a, 3b, 3c, and 3d, we show the variation in execution time required in the post-processing and concurrent processing frameworks for the data sets mentioned above. In all following tables *P* denotes post-processing approach using the *Min* heuristic and *C* denotes concurrent approach using the *NR* heuristic.

The results show that the execution time in the post-processing approach increases slowly with lower constraint values. The execution time for the concurrent approach, on the other hand, remains more or less constant. Note that the concurrent processing approach takes more time than the post-processing approach for all the data sets. Even if the basic unconstrained decomposition time shown in Table 1 is added to the time for fixing the constraints in the post-processing approach, the total time is less than the concurrent processing time. However, as observed from the results above, the concurrent processing approach works for tighter (smaller) values of the constraints where the post-processing approach fails. As an example, for the Apj data set, with $MRC_{user} = 8$ and $MRC_{perm} = 25$, the post-processing approach takes a total of $3.64 + 0.2 = 3.84$ seconds for generating the roles, while the concurrent processing approach takes 11.6 seconds. The number of roles generated are 468 and 484, respectively. However, as the constraint on permission-distribution is reduced to 15, the post-processing approach fails to fix the same. On the other

TABLE 4
Comparative Performance of Post-Processing (P), Concurrent (C), Post Processing Graph Optimization (PGO), and Concurrent Graph Optimization (CGO) in Terms of Number of Roles Generated / the WSC Metric

(a) Americas-large

MRC_{perm}	MRC_{user} 6				5				4				3			
	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO
144	421/106560	521/70943	1051/105794	1501/38708	422/106484	513/82103	1051/105794	1501/38708	424/106431	503/80001	1051/105794	1501/38708	x/x	489/83038	1051/105794	1501/38708
140	423/105161	521/70943	1082/103284	1501/38708	424/105085	513/82103	1082/103284	1501/38708	426/105032	503/80001	1082/103284	1501/38708	x/x	489/83038	1082/103284	1501/38708
130	423/104264	521/70943	1086/92290	1501/38708	424/104188	513/82103	1086/92290	1501/38708	426/104135	503/80001	1086/92290	1501/38708	x/x	489/83038	1086/92290	1501/38708
120	426/101422	521/70943	1156/85756	1501/38708	427/101346	513/82103	1156/85756	1501/38708	429/100517	503/80001	1156/85756	1501/38708	x/x	489/83038	1156/85756	1501/38708
110	430/100289	521/70943	1216/78975	1501/38708	431/100213	513/82103	1216/78975	1501/38708	433/100160	503/80001	1216/78975	1501/38708	x/x	489/83038	1216/78975	1501/38708
100	434/100515	521/70943	1246/74818	1501/38708	435/100439	513/82103	1246/74818	1501/38708	437/100152	503/80001	1246/74818	1501/38708	x/x	x/x	1246/74818	1501/38708

(b) Apj

MRC_{perm}	MRC_{user} 11				10				8				6			
	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO
67	456/6188	484/5047	1538/8398	484/4097	457/6071	484/5047	1538/8398	484/4097	459/5983	484/5047	1538/8398	484/4097	463/5971	484/5046	1538/8398	484/4097
65	457/6161	484/5047	1536/8388	484/4097	458/6071	484/5047	1536/8388	484/4097	460/5995	484/5047	1536/8388	484/4097	464/5908	484/5046	1536/8388	484/4097
55	460/6076	484/5047	1534/8354	484/4097	461/5988	484/5047	1534/8354	484/4097	463/5974	484/5047	1534/8354	484/4097	x/x	484/5046	1534/8354	484/4097
45	463/5991	484/5047	1532/8320	484/4097	464/5991	484/5047	1532/8320	484/4097	467/5991	484/5047	1532/8320	484/4097	x/x	484/5046	1532/8320	484/4097
35	463/5953	484/5047	888/6237	484/4097	464/5937	484/5047	888/6237	484/4097	468/5933	484/5047	888/6237	484/4097	x/x	484/5046	888/6237	484/4097
25	463/5874	484/5047	529/5267	484/4097	464/5859	484/5047	529/5267	484/4097	468/5831	484/5047	529/5267	484/4097	x/x	484/5046	529/5267	484/4097
15	463/5781	484/5047	519/5247	484/4097	464/5765	484/5047	519/5247	484/4097	x/x	484/5047	519/5247	484/4097	x/x	484/5046	519/5247	484/4097
5	x/x	482/5347	489/5179	484/4092	x/x	482/5347	489/5179	484/4092	x/x	482/5347	489/5179	484/4092	x/x	x/x	489/5179	484/4092

(c) Firewall1

MRC_{perm}	MRC_{user} 21				16				12				8			
	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO
26	69/6239	69/2688	116/5351	86/1684	70/6266	69/2688	116/5351	86/1684	71/6078	69/2688	116/5351	86/1684	75/5413	69/2688	116/5351	86/1684
25	70/6028	69/2688	116/5343	86/1684	71/6055	69/2688	116/5343	86/1684	72/5869	69/2688	116/5343	86/1684	76/5253	69/2688	116/5343	86/1684
21	71/5553	69/2688	113/4852	86/1684	72/5580	69/2688	113/4852	86/1684	73/5701	69/2688	113/4852	86/1684	78/5705	69/2688	113/4852	86/1684
17	75/5171	69/2688	110/4520	86/1684	76/5178	69/2688	110/4520	86/1684	77/5225	69/2688	110/4520	86/1684	x/x	69/2688	110/4520	86/1684
13	76/4887	69/2688	106/4188	86/1684	77/4914	69/2688	106/4188	86/1684	78/4933	69/2688	106/4188	86/1684	x/x	69/2688	106/4188	86/1684
9	76/4605	69/2688	102/3856	86/1684	77/4612	69/2688	102/3856	86/1684	78/4633	69/2688	102/3856	86/1684	x/x	69/2688	102/3856	86/1684

(d) Firewall2

MRC_{perm}	MRC_{user} 9				8				7				6			
	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO
3	10/1858	12/1803	14/1742	12/1212	11/1780	12/1803	14/1742	12/1212	11/1713	12/1803	14/1742	12/1212	x/x	12/1803	14/1742	12/1212
2	x/x	11/1808	13/1739	14/1216	x/x	11/1808	13/1739	14/1216	x/x	11/1808	13/1739	14/1216	x/x	11/1808	13/1739	14/1216

hand, the concurrent approach gives a valid solution with 484 roles needing 11.7 seconds.

Note that the time taken by the post-processing approach to fix the constraint violations depends on the number of violations present in the initial decomposition provided as input. For example, for Firewall1, an initial decomposition with the minimum biclique cover based approach [10] contains five users having more than eight roles and 79 permissions belonging to more than nine roles.

The number of constraint violations would depend on the value of the constraint chosen. Thus, in the case where $MRC_{user} = 8$ and $MRC_{perm} = 9$ (which corresponds to the bottom-right combinations of Tables 2c and 3c), the total number of violations is 84. A different unconstrained role mining algorithm would generate UA and PA with different number of violating users and permissions. The execution time for the post-processing approach would accordingly vary.

5.3 Comparison with Graph Optimization Algorithm

As mentioned in Section 3, the initial decomposition for the post-processing approach can be made using any existing role mining algorithm. For the results presented above, we had considered the minimum biclique cover based approach proposed in [10] for generating the UA and PA matrices. This was done to provide a meaningful comparison with the concurrent approach, which is also based on the same algorithm. In this section, we perform a comparative analysis between the minimum biclique cover based post-processing and concurrent approaches with another pair of post-processing and concurrent approaches that are based on the graph optimization algorithm presented in [9]. This algorithm forms role hierarchies by splitting and merging of roles.

It may be noted that, besides using the number of roles as the objective function to minimize, recent research in role mining has focused on optimizing other metrics like the weighted structural complexity (WSC) [13]. The WSC metric is a weighted sum of the number of roles, user-role assignments, role-permission assignments, elements in the role hierarchy relation and the number of direct user to permission assignments.

In Tables 4a, 4b, 4c, and 4d, we show the variation in the number of generated roles as well as the WSC metric value achieved by the *Min* heuristic of the proposed post-processing approach (denoted as *P* in the table), *NR* heuristic of the proposed concurrent approach (denoted as *C* in the table), graph optimization based concurrent approach (*CGO*) and graph optimization based post-processing approach (*PGO*). Thus, the first two values giving the number of roles for each MRC_{user}-MRC_{perm} combination in these tables are taken from the respective columns of Tables 2a, 2b, 2c, and 2d.

PGO initially generates the UA, PA and RA (role-to-role relationship representing role hierarchy) matrices from a given UPA matrix. Every user is assigned to a single role but through role hierarchy he can obtain all the required permissions. It then fixes the permission-distribution constraint violation by revising the RA through merging/splitting of roles. Since a user is directly assigned to one role only, the role-usage constraint has no effect on PGO. Hence, the two-constraint problem reduces to a single constraint problem in this case. The concurrent graph optimization approach (CGO), on the other hand, enforces the constraints at the time of forming the set of roles and the role hierarchy. Here also, the obtained decomposition contains only one role per user and hence, the role-usage constraint has no effect.

From the tables, it is observed that both PGO and CGO can produce a solution to the constrained role mining problem for all values of the constraints. This is expected because, each user is only assigned to one role directly and the problem essentially reduces to a single constraint problem. All the necessary permissions of a user are made available through the role hierarchy, which is not affected by the cardinality constraint. However, the number of roles generated for both PGO and CGO is much higher compared to the proposed post-processing and concurrent approaches. This is due to the formation of a large number of roles through repeated splitting and merging in the graph optimization approach. Also, in *PGO*, the number of roles generated remains unchanged for all the data sets except for Firewall2 even when the cardinality constraint on permission is changed (No change across columns in the same row for either CGO or PGO is expected since these approaches always generate one role per user and, therefore, variation in MRC_{user} has no effect on the number of roles). This is because the maximum number of roles to which any permission belongs in the initial unconstrained decomposition of the UPA matrices using the graph optimization method for the various data sets are as follows: 35 for Americas-large, 6 for Apj, 6 for Firewall1, and 3 for Firewall2. The MRC_{perm} values considered in the tables are above these except for Apj and Firewall2. Even for Apj, it is close to the smallest MRC_{perm} value considered, i.e., 5 as a result of which there was no change in the number of roles generated. For Firewall2, however, a change in the generated number of roles is observed as the value of MRC_{perm} is changed from 3 to 2.

The WSC values show the same type of variation as observed for the number of roles, except that a minor variation in WSC is observed for the Apj data set as the value of MRC_{perm} is changed from 15 to 5. Other than PGO, the proposed concurrent approach has the smallest value of WSC for three of the data sets (i.e., all the data sets except Firewall2) for all combinations of MRC_{user} and MRC_{perm}. However, although PGO has lower WSC values, the number of roles generated is much higher.

For CGO, no definite pattern is observed as the value of MRC_{perm} is changed for the different data sets. In this approach, whether the number of roles would increase or decrease with changing value of the constraint depends on the data set under consideration and the order in which roles are chosen for merging/splitting. As has been noted in [9] as well, depending on the scenario, the number of roles can either increase or decrease or remain unchanged. The number of roles and corresponding role hierarchy after mining may be different even for the same data set if the order of role selection is different.

The comparative results of Table 4 further show that the final number of roles generated by the post-processing algorithm would depend on the design of the initial role mining algorithm. For example, the graph optimization based algorithm used here for comparison assigns each user to only one role directly. Hence, the results differ significantly from that obtained for the minimum biclique cover based approach. In either case, the proposed post-processing framework can be used for role mining under role-usage and permission-distribution cardinality constraints.

Overall, it is observed that the proposed concurrent approach has lower WSC values compared to the post-processing approach and it can also satisfy more number of constraints. The number of roles generated is also much lower than the CGO and PGO approaches. However, the time taken by the proposed post-processing approach is much lower as reported in Table 3.

Hence, it can be concluded that a meaningful strategy for meeting cardinality constraints in role mining would be to do an initial unconstrained decomposition of a given UPA into UA and PA, and then follow a post-processing approach to fix the violations for the given pair of MRC_{user} and MRC_{perm} constraints. If no valid solution can be generated while trying to fix the violations, the concurrent approach should be tried on the initial UPA itself. The concurrent processing heuristics should be tried in the order NP, NU, XR and NR.

6 Discussions

We now consider other existing approaches to role engineering and discuss how our proposed approaches can complement those efforts.

6.1 Meeting Other Constraints

Besides considering MRC_{user} and MRC_{perm}, one could also consider their dual constraints, namely, the maximum number of users who can be a member of any role (denoted by MUC_{role}) and the maximum number of permissions that a role can have (denoted by MPC_{role}). MRC_{user} ensures uniform distribution of responsibilities and MRC_{perm} allows the privileged permissions to belong only to certain roles. Enforcing MUC_{role} is meaningful when certain roles are made available only to a limited number of users. Permissions that can be obtained through these roles are then available only to a small set of users. The fourth constraint MPC_{role} facilitates uniform distribution of permissions among the roles. Thus, cardinality constraints can enforce various restrictions on user-role and role-permission assignment relations. Such types of restrictions are relevant from an organizational point of view since they reflect different organizational policies.

The problem formulation as expressed through Definitions 1, 2 and 3 of Section 2 can be extended to reflect all four cardinality constraints as follows:

Definition 6. *All cardinality constraint problem (ACP). Given $X_{n\times m}$, find a correct decomposition $C_{n\times q}$ and $R_{q\times m}$ which minimizes the value of q under the conditions $\sum_{j=1}^{q} c_{ij} \leq MRC_{user}$ for all $1 \leq i \leq n$, $\sum_{j=1}^{q} r_{jt} \leq MRC_{perm}$ for all $1 \leq t \leq m$, $\sum_{i=1}^{n} c_{ij} \leq MUC_{role}$ for all $1 \leq j \leq q$ and $\sum_{t=1}^{m} r_{jt} \leq MPC_{role}$ for all $1 \leq j \leq q$.*

The three problems addressed in our current work are special cases of this formulation with MUC_{role} and MPC_{role} set to very large values. Developing suitable heuristics when all the four cardinality constraints are present is left as a future extension of this work.

Besides satisfying the cardinality constraints in the basic role mining problem [7] as discussed in this paper, one can also consider constraints for the $\delta - approx$ and the $min - noise$ role mining problems introduced in [7]. For a constrained version of $\delta - approx$ RMP, the post-processing approach will need to consider two situations. If the UA and PA matrices are $\delta_1 - consistent$ [7] with the given UPA, where $\delta_1 < \delta$, δ being the upper bound on the number of mismatches allowed, updating role assignments of users or role belongingness of permissions so that the newly created mismatches do not exceed $\delta - \delta_1$ can be first tried. However, if the number of mismatches already present equals δ, no further error can be allowed and the post-processing step would be similar to that presented in Section 3. In the concurrent processing approach, Phase 1 would be similar to that proposed in Section 4. In Phase 2, an additional stopping criterion will need to be used to terminate the iterations if the error falls below δ.

For constrained $min - noise$ RMP, the number of roles is fixed and under closed world assumption, the post-processing approach first needs to remove the extra roles from the violating users and permissions. For a violating user, only those roles should be removed that contain the least number of permissions. Similarly, for the violating permissions, those roles should be removed that affect the least number of users. This effectively removes a number of user-permission assignments which should then be re-introduced with the help of the existing roles without causing new constraint violations. The concurrent approach, on the other hand, would be a single phase algorithm in which the roles are formed without causing violation. The user-permission assignments missed would count towards noise, which needs to be minimized through a suitable heuristic.

Similar to the graph optimization approach [9], Frank et al. [28] propose a disjoint decomposition model which aims to reduce complexity while maintaining flexibility in the system. The model allows a user to belong to only one business role and a permission to belong to exactly one technical role. This reduces the number of user-role assignments and permission-role assignments to one. A technical role can be shared by any number of business roles and many users can acquire the same business role. However, this leads to fewer number of permissions included in each role resulting in a substantial increase in the total number of roles. The results of constrained role mining for disjoint decomposition are likely to be similar to that presented for [9].

Other than cardinality constraints, pre-requisite constraints and separation of duty constraints are also important in enforcing organizational policies through RBAC, and have been considered in previous work [8], [23]. There is no work, however, that considers the cardinality constraints and the separation of duty constraints together. Role mining in the presence of separation of duty constraints is likely to affect and be affected by the presence of cardinality constraints. We plan to study this in the future.

6.2 Cardinality Constraints in Hybrid Role Mining

In many business scenarios, a more interactive role engineering approach is preferred in which the effort is to formulate meaningful roles by studying the organizational role structure [24], [25], [26]. Hybrid role mining combines both top-down and bottom-up approaches.

The proposed post-processing and concurrent approaches can be extended to handle cardinality constraints in hybrid role mining. A consistent RBAC state is the input for the post-processing algorithm which might either be an output of any unconstrained role mining algorithm or the output of a hybrid role mining method or may even be the result of a top down role engineering process. Instead of fully automating the post-processing framework, the steps that cause new roles to be formed (Line Nos. 9-10 and 13-14 of Algorithm 2) can take input from the user. This would lead to the formation of more semantically meaningful roles. The concurrent approaches dealing with multiple cardinality constraints can also be modified for a hybrid approach. Now, the heuristics used for selecting vertices will have to be suitably changed and instead of merely considering minimum or maximum number of edges to be included in the next biclique (Line No. 7 of Algorithm 4), the decision should be guided by the organizational structure.

6.3 Cardinality in Cost Based Optimization and Machine Learning Based Approaches

It is possible to formulate the problem of role mining in presence of multiple cardinality constraints using the cost-driven approach reported in [29]. Suitable penalty weights can be defined to account for constraint violations. It will also provide flexibility by allowing for imposition of soft constraints. For example, if the total cost comprises of, among others, a cost due to the number of roles and another due to violation of a constraint, relative weights can determine whether small violations would be allowed if there is a substantial reduction in the number of roles. The WSC metric used in [24] can be enhanced to include a suitably weighted sum of the number of constraint violations.

It may be noted that the number of roles generated will depend on the optimization criterion chosen during role mining. For example, if the goal is to reduce the WSC metric, the number of roles itself might not get minimized. However, the proposed post-processing and concurrent frameworks can be used for any optimization criterion with appropriate modification in the greedy heuristic used for forming the next role at each step of the iteration. The ability of the concurrent processing approach to handle relatively smaller (tighter) values of constraints as compared to the post-processing approach is expected to be the same independent of the criteria.

Frank et al. in [27] show that the role mining process can be formulated as multi-assignment clustering of Boolean data. In multi-assignment clustering, an object may belong to more than one cluster at the same time. In role mining, multi-assignments exist in the user-role assignment as well as in the role-permission assignment. The approaches can suitably be modified to enforce upper bounds on the number of assignments during the clustering process based on cardinality constraints.

7 Related Work

A number of different algorithms have been proposed in the past few years for role mining. Vaidya et al. [7] first formalized the basic role mining problem. They show that RMP and its variants such as δ-approx RMP and Min-noise RMP are NP-complete problems and adapt solutions to the database tiling problem [5] to this area. Vaidya et al. [4] proposes a technique called Complete Miner, which uses subset enumeration over all users to generate candidate roles. FastMiner [4] limits the intersections to only pairs of users. Lu et al. [8] use optimal boolean matrix decomposition to decompose the UPA into UA and PA. In [21], Uzun et al. propose optimization models for the extended role mining problem which allow negative assignments in roles.

A role mining tool named ORCA has been proposed by Schlegelmilch and Steffens [3]. The work in [9] proposes a graph optimization method to find role hierarchies in RBAC. In [10], Alina et al. propose a bipartite graph representation of the UPA matrix for deriving the minimum number of roles, which is equivalent to finding the minimum biclique cover of the graph. In [14], Molloy et al. propose a comprehensive framework for evaluating different role mining algorithms. The work in [20] proposes a generic process model for role mining that divides the entire task of role mining into several sub-phases. Zhang et al. use permission set pattern data mining for finding a set of practical and useful role hierarchies [6]. The work in [22] proposes a six step methodology to make role mining practical and usable.

So far, there is little work on constrained role mining. Kumar et al. [19] and Blundo and Cimato [1] propose a constrained permission set approach for role mining, which restricts the maximum number of permissions in a role. A biclique cover based approach has been proposed by Hingankar and Sural [2] that limits the maximum number of users for a role. Recently, the authors of [15] have proposed two approaches for role mining under role usage constraint restricting the maximum number of roles for a user. However, so far, no work has been reported on role mining with multiple constraints. The work presented here is the first such approach. The notion of a post-processing and a concurrent framework is also being proposed for the first time in this paper.

8 Conclusions and Future Work

We have proposed a post-processing and a concurrent framework for role mining with role-usage and permission-distribution cardinality constraints considered separately and together. The proposed approaches have been tested with real data sets. While the concurrent framework can generate a solution for certain combinations of constraint values in which the post-processing approach fails, it is less efficient. The proposed frameworks are generic and can be extended to include various other algorithms for role mining as well.

In the future, we plan to consider other cardinality constraints in this framework like the number of users a role can have or the number of permissions a role can have. Additionally, the framework can be enriched by considering other constraints like pre-requisite constraints, separation of duty and role hierarchy during role mining besides the use of cardinality constraints.

References

[1] C. Blundo, S. Cimato, "Constrained Role Mining" ArXiv e-prints, Mar. 2012.

[2] M. Hingankar and S. Sural, "Towards Role Mining with Restricted User-Role Assignment," *Proc. Wireless VITAE Conf.*, pp. 1-5, 2011.

[3] J. Schlegelmilch and S. Ulrike, "Role Mining with ORCA," *Proc. 10th ACM Symp. Access Control Models and Technologies (SACMAT)*, pp. 168-176, 2005.

[4] J. Vaidya, V. Atluri, and J. Warner, "Role Miner: Mining Roles Using Subset Enumeration," *Proc. 13th ACM Conf. Computer and Comm. Security (CCS)*, pp. 144-153, 2006.

[5] F. Geerts, B. Goethals, and T. Mielikinen, "Tiling Databases," *Proc. Seventh Int'l Conf. Discovery Science*, pp. 278-289, Springer, 2004.

[6] D. Zhang, R. Kotagiri, E. Tim, and Y. Trevor, "Permission Set Mining: Discovering Practical and Useful Roles," *Proc. Ann. Computer Security Applications Conf. (ACSAC)*, pp. 247-256, 2008.

[7] J. Vaidya, V. Atluri, and Q. Guo, "The Role Mining Problem: Finding a Minimal Descriptive Set of Roles," *Proc. 12th ACM Symp. Access Control Models and Technologies (SACMAT)*, pp. 175-184, 2007.

[8] H. Lu, J. Vaidya, and V. Atluri, "Optimal Boolean Matrix Decomposition: Application to Role Engineering," *Proc. IEEE 24th Int'l Conf. Data Eng. (ICDE)*, pp. 297-306, 2008.

[9] D. Zhang, R. Kotagiri, and E. Tim, "Role Engineering Using Graph Optimization," *Proc. 12th ACM Symp. Access Control Models and Technologies (SACMAT)*, pp. 139-144, 2007.

[10] E. Alina, H. William, M. Nikola, R. Prasad, R. Schreiber, and R.E. Tarjan, "Fast Exact and Heuristic Methods for Role Minimization Problems," *Proc. 13th ACM Symp. Access Control Models and Technologies (SACMAT)*, pp. 1-10, 2008.

[11] 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, Feb. 1996.

[12] D.F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, and R. Chandramouli, "Proposed NIST Standard for Role-Based Access Control," *ACM Trans. Information and System Security*, vol. 4, no. 3, pp. 224-274, 2001.

[13] I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S.B. Calo, and J. Lobo, "Mining Roles with Multiple Objectives," *ACM Trans. Information and System Security*, vol. 13, no. 4, article 36, 2010.

[14] I. Molloy, N. Li, T. Li, Z. Mao, Q. Wang, and J. Lobo, "Evaluating Role Mining Algorithms," *Proc. 14th ACM Symp. Access Control Models and Technologies (SACMAT)*, pp. 95-104, 2009.

[15] J.C. John, S. Sural, V. Atluri, and J. Vaidya, "Role Mining under Role-Usage Cardinality Constraint," *27th IFIP Int'l Information Security and Privacy Conf. (SEC)*, 2012.

[16] E.J. Coyne, "Role Engineering," *Proc. First ACM Workshop Role-Based Access Control*, pp. 15-16, 1996.

[17] M. Kuhlmann, D. Shobat, and G. Schimpf, "Role Mining-Revealing Business Roles for Security Administration Using Data Mining Technologies," *Proc. Eighth ACM Symp. Access Control Models and Technologies (SACMAT)*, pp. 179-186, 2003.

[18] H. Fleischner, E. Mujuni, D. Paulusma, and S. Szeider, "Covering Graphs with Few Complete Bipartite Subgraphs," *Proc. 27th Int'l Conf. Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pp. 340-351, 2007.

[19] R. Kumar, S. Sural, and A. Gupta, "Mining RBAC Roles under Cardinality Constraint," *Proc. Sixth Int'l Conf. Information Systems Security (ICISS '10)*, pp. 171-185, 2010.

[20] L. Fuchs and S. Meier, "The Role Mining Process Model - Underlining the Need for a Comprehensive Research Perspective," *Proc. Sixth Int'l Conf. Availability, Reliability and Security (ARES)*, pp. 35-42, 2011.

[21] E. Uzun, V. Atluri, H. Lu, and J. Vaidya, "An Optimization Model for the Extended Role Mining Problem," *Proc. 25th IFIP 25th Ann. IFIP WG 11.3 Conf. Data and Applications Security and Privacy (DBSEC)*, pp. 76-89, 2011.

[22] N.V. Verde, J. Vaidya, V. Atluri, and A. Colantonio, "Role Engineering: From Theory to Practice," *Proc. Second ACM Conf. Data and Application Security and Privacy (CODASPY)*, pp. 181-192, 2012.

[23] H. Lu, J. Vaidya, V. Atluri, and Y. Hong, "Constraint-Aware Role Mining via Extended Boolean Matrix Decomposition," *IEEE Trans. Dependable and Security Computing*, vol. 9, no. 5, pp. 655-669, Sept./Oct. 2012.

[24] I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. Calo, and J. Lobo, "Mining Roles with Semantic Meanings," *Proc. 13th ACM Symp. Access Control Models and Technologies (SACMAT)*, pp. 21-30, 2008.

[25] A. Colantonio, R. Di Pietro, A. Ocello, and N.V. Verde, "A Formal Framework to Elicit Roles with Business Meaning in RBAC Systems," *Proc. 14th ACM Symp. Access Control Models and Technologies (SACMAT)*, pp. 85-94, 2009.

[26] M. Frank, A.P. Streich, D. Basin, and J.M. Buhmann, "A Probabilistic Approach to Hybrid Role Mining," *Proc. 16th ACM Conf. Computer and Comm. Security (CCS)*, pp. 101-111, 2009.

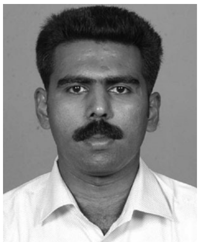
[27] M. Frank, A.P. Streich, D. Basin, and J.M. Buhmann, "Multi-Assignment Clustering for Boolean Data," *J. Machine Learning Research*, vol. 13, pp. 459-489, Feb. 2012.

[28] M. Frank, J.M. Buhman, and D. Basin, "Role Mining with Probabilistic Models," *ACM Trans. Information and System Security*, vol. 15, article 15, Apr. 2013.

[29] A. Colantonio, R. Di Pietro, and A. Ocello, "A Cost-Driven Approach to Role Engineering," *Proc. ACM Symp. Applied Computing (SAC)*, pp. 2129-2136, 2008.

Pullamsetty Harika received the BTech degree in computer science and engineering from JNTU and the MTech degree in information technology from IIT, Kharagpur. Currently, she is a software development engineer in Amazon. Her research interests include access control and role mining.

Marreddy Nagajyothi received the BTech degree in computer science and engineering from Acharya Nagarjuna University and the MTech degree in information technology from IIT, Kharagpur. Currently, she is a software development engineer in Amazon. Her research interests include role mining and access control.



John C. John received the MTech degree from IIT, Kharagpur, in 2012, and is currently working toward the PhD degree. He is an assistant professor in the Department of Computer Applications at the Rajiv Gandhi Institute of Technology, India. His research interests include access control, distributed systems, and data mining.

Shamik Sural received the PhD degree from Jadavpur University. He is a professor at the School of Information Technology, IIT Kharagpur. He received the Alexander von Humboldt Fellowship for Experienced Researchers. He has published more than 150 research papers. His research interests include computer security, data mining, and database systems.

Jaideep Vaidya received the PhD degree in computer science from Purdue University. He is an associate professor in the MSIS Department at Rutgers University. His general area of research is in data mining, data management, security, and privacy. He has published more than 100 technical papers in peer-reviewed journals and conference proceedings, and has received three best paper awards.

Vijayalakshmi Atluri is a professor of computer information systems in the MSIS Department at Rutgers University. Her research interests include information security, privacy, databases, and workflow management. She has published more than 150 papers in journals and conferences. She received the National Science Foundation CAREER Award, and the Rutgers University Research Award for untenured faculty.

Supplemental Material

In the supplemental material, we first give the remaining proofs of NP-completeness. We then provide a detailed illustrative example for both the post-processing and concurrent approaches. Finally, we provide the complete experimental evaluation with results over all datasets.

1 PROOF OF NP-COMPLETENESS

Definition 1: Decision Role-Usage Cardinality Constraint Problem (Decision RUP). Given $X_{n\times m}$, $MRC_{user} > 0$ and $k > 0$, is there a set of decompositions $C_{n\times q}$ and $R_{q\times m}$ such that $q \leq k$ and $\sum_{j=1}^{q} c_{ij} \leq MRC_{user}$ for all $1 \leq i \leq n$?

Theorem 1: Decision RUP is NP-Complete.

Proof: Decision RMP is known to be NP-Complete. This problem can be reduced to Decision RUP by defining $MRC_{user} \geq k$. Since it is a direct one-to-one mapping, the reduction can be done in polynomial time. Hence, decision RUP is NP-hard. The matrices X, C and R together form a certificate, which can be verified in polynomial time. Hence, Decision RUP is in NP. Thus, Decision RUP is NP-Complete. □

Definition 2: Decision Permission-Distribution Cardinality Constraint Problem (Decision PDP).

Given $X_{n\times m}$, $MRC_{permission} > 0$ and $k > 0$, is there a set of decompositions $C_{n\times q}$ and $R_{q\times m}$ such that $q \leq k$ and $\sum_{j=1}^{q} r_{jt} \leq MRC_{permission}$ for all $1 \leq t \leq m$?

Theorem 2: Decision PDP is NP-Complete

Proof: The proof is similar to that for Theorem 1. □

TABLE 1: Original UPA matrix

	p1	p2	p3	p4	p5	p6	p7	p8
u1	1	0	1	0	1	0	1	0
u2	1	1	1	1	1	0	1	0
u3	1	1	1	1	1	1	0	1
u4	0	1	0	1	1	0	0	1

TABLE 2: UA generated from the UPA of Table 1

	r1	r2	r3	r4	r5	r6
u1	0	1	0	1	0	0
u2	0	1	0	1	1	0
u3	1	0	0	1	1	1
u4	0	0	1	0	1	0

TABLE 3: PA generated from the UPA of Table 1

	p1	p2	p3	p4	p5	p6	p7	p8
r1	0	0	0	0	1	1	0	0
r2	0	0	0	0	1	0	1	0
r3	0	0	0	0	1	0	0	1
r4	1	0	1	0	0	0	0	0
r5	0	1	0	1	0	0	0	0
r6	0	0	0	0	0	1	0	1

TABLE 4: Intermediate UA after fixing u4 constraint

	r1	r2	r3	r4	r5	r6	r7
u1	0	1	0	1	0	0	0
u2	0	1	0	0	0	0	1
u3	1	0	0	0	0	1	1
u4	0	0	1	0	1	0	0

TABLE 5: Intermediate PA after fixing u4 constraint

	p1	p2	p3	p4	p5	p6	p7	p8
r1	0	0	0	0	1	1	0	0
r2	0	0	0	0	1	0	1	0
r3	0	0	0	0	1	0	0	1
r4	1	0	1	0	0	0	0	0
r5	0	1	0	1	0	0	0	0
r6	0	0	0	0	0	1	0	1
r7	1	1	1	1	0	0	0	0

2 ILLUSTRATIVE EXAMPLE

2.1 Illustrative Example for Post Processing

We now illustrate the working of the post-processing framework with both constraints applied simultaneously. Consider the UPA in Table 1 and its corresponding decomposition into UA (Table 2) and PA (Table 3).

Let MRC_{user}=3 and MRC_{perm}=2. From Table 2, UserRoleCount[]=2,3,4,2 and PermRoleCount[]=1,1,1,1,3,2,1,2. u4 is the violating user and p5 is the violating permission. RoleUserCount[]=1,2,1,3,3,1, RolePermCount[]=2,2,2,2,2,2, RU=r4,r5, RI=r2,r3. Hence, UserViolationCount=1, PermViolationCount=1.

Considering u4, $UserRoleCount[u4] - (MRC_{user} - 1)$=4-(3-1)=2. Hence, 2 roles are to be merged. The role set of u4 consists of r1,r4,r5,r6. However, r1 and r6 cannot be used for merging as they are not in RU. Hence, a new role is formed by merging r4 and r5. The intermediate UA and PA matrices after fixing the first violation are shown in Tables 4 and 5, respectively.

At this stage, UserRoleCount[]=2,2,3,2. With MRC_{user}=3, there is no more violating user left. The updated value of PermRoleCount[]=2,2,2,2,3,2,1,2. RoleUserCount[]=1,2,1,1,1,1,2, RolePermCount[]=2,2,2,2,2,2,4, RU=ϕ, RI=r2,r3,r4,r5. For MRC_{perm}=2, p5 is still a violating permission. So, a new role has to be formed through intersection of $PermRoleCount[p5] - (MRC_{perm} - 1)$=3-(2-1)=2 roles. Now, the role set of p5 is r1,r2,r3. However, r1 cannot be used for intersection as it is not in RI. So, the new role is formed using r2,r3. After adding this new role, the final UA and PA are shown in Tables 6 and 7, respectively. Now, UserRoleCount[]=3,3,3,3 and PermRoleCount[]=2,2,2,2,2,2,1,2. For MRC_{user}=3

TABLE 6: UA after fixing constraint on u4 and p5

	r1	r2	r3	r4	r5	r6	r7	r8
u1	0	1	0	1	0	0	0	1
u2	0	1	0	0	0	0	1	1
u3	1	0	0	0	0	1	1	0
u4	0	0	1	0	1	0	0	1

TABLE 7: PA after fixing constraint on u4 and p5

	p1	p2	p3	p4	p5	p6	p7	p8
r1	0	0	0	0	1	1	0	0
r2	0	0	0	0	0	0	1	0
r3	0	0	0	0	0	0	0	1
r4	1	0	1	0	0	0	0	0
r5	0	1	0	1	0	0	0	0
r6	0	0	0	0	0	1	0	1
r7	1	1	1	1	0	0	0	0
r8	0	0	0	0	1	0	0	0

and MRC_{perm}=2, there are no more violating users or permissions. Hence, the algorithm terminates giving UA and PA matrices of Tables 6 and 7 as the final output.

2.2 Illustration of Concurrent Processing

We consider the same UPA as shown in Table 1 to illustrate how the two constraints are enforced together in concurrent processing. Out of the various heuristics suggested in the previous sub-section, we consider heuristic (iv), i.e., choosing the vertex (user or permission as the case may be) with the minimum number of roles that can yet be assigned.

Similar to the illustrative example on post-processing framework, let $MRC_{user} = 3$ and $MRC_{perm} = 2$. Initially, role count for each user and each permission is 0. Among all the user and permission vertices, p6 has the minimum number of uncovered edges. So we select p6 which is assigned to user u3. Along with p6, the permissions possessed by u3 are considered. A role is formed with u3 and p1,p2,p3,p4,p5,p6,p8. p7 is next selected and a role is formed with u1,u2 and p1,p3,p7. As no other vertex satisfies the criteria of Phase 1, the algorithm enters Phase 2. p5 is first selected and a role is formed with u1,u2,u4 and p5. Repeating the same procedure, further roles are formed with u2,u4 and p2,p4 as well as with u4 and p8. The final UA and PA matrices are shown in Tables 8 and 9, respectively.

It is observed from the examples of post-processing and concurrent processing frameworks that the number of roles formed in the concurrent approach is less than that obtained in the post-processing approach. This is since the concurrent approach creates roles *ab initio*, the post-processing approach can only fix the given constraints starting from an already given decomposition.

3 Experimental Evaluation

We now describe the experimental evaluation. The algorithms proposed in this paper have been implemented in C on a 3.1GHz Intel i5-2400 CPU having 4GB RAM. Experiments were carried out on nine real-world data sets [1], namely, Americas-large, Americas-small, Apj, Customer, Domino, EMEA, Firewall1, Firewall2 and Healthcare as shown in Table 10. It may be noted that

TABLE 8: UA generated from the UPA of Table 1 with both cardinality constraint values of 3

	r1	r2	r3	r4	r5
u1	0	1	1	0	0
u2	0	1	1	1	0
u3	1	0	0	0	0
u4	0	0	1	1	1

TABLE 9: PA generated from the UPA of Table 1 with both cardinality constraint values of 3

	p1	p2	p3	p4	p5	p6	p7	p8
r1	1	1	1	1	1	1	0	1
r2	1	0	1	0	0	0	1	0
r3	0	0	0	0	1	0	0	0
r4	0	1	0	1	0	0	0	0
r5	0	0	0	0	0	0	0	1

these data sets have been widely used in the literature for studying the performance of various role mining algorithms [2],[1]. Each data set represents a UPA matrix. In the concurrent processing framework, the UPA matrix serves as the input to the algorithm. In the post processing framework, an unconstrained decomposition of the UPA matrix into UA and PA matrices is required as the input. We apply the Minimum Biclique Cover (MBC) based algorithm [2] to decompose the given UPA. The number of roles obtained in this unconstrained decomposition are also shown in Table 10, which quite well match with that reported in the literature [2]. The corresponding execution time required is also included in the table.

We first study the impact of a single constraint on the number of roles generated using the two approaches (Sub-section 3.1). This is followed by Sub-section 3.2 in which a comparative study is made between the four heuristics of each of the two proposed approaches in terms of the number of roles generated. A comparison of the execution time of the two best approaches, one from post-processing and one from the concurrent framework, is also included in this sub-section. Finally, in Sub-section 3.3, we do a comparative study with the graph optimization based algorithm proposed in [3].

3.1 Role-usage Cardinality Constraint

Figures 1(a)-(d) show the variation in the number of roles generated for the Americas-small, Apj, Domino and Healthcare data sets by the post-processing and the concurrent processing frameworks as the value of the cardinality constraint MRC_{user} is varied. For comparison, the results of a recently proposed algorithm named

TABLE 10: Data set description

Data set	Users	Permissions	Roles	Execution time (in sec.)
Americas-large	3485	10127	421	94.360
Americas-small	3477	1587	211	3.640
Apj	2044	1164	456	3.640
Customer	10961	284	276	2.100
Domino	79	231	20	0.002
EMEA	35	3046	34	0.012
Firewall1	365	709	69	0.066
Firewall2	325	590	10	0.012
Healthcare	46	46	15	0.001

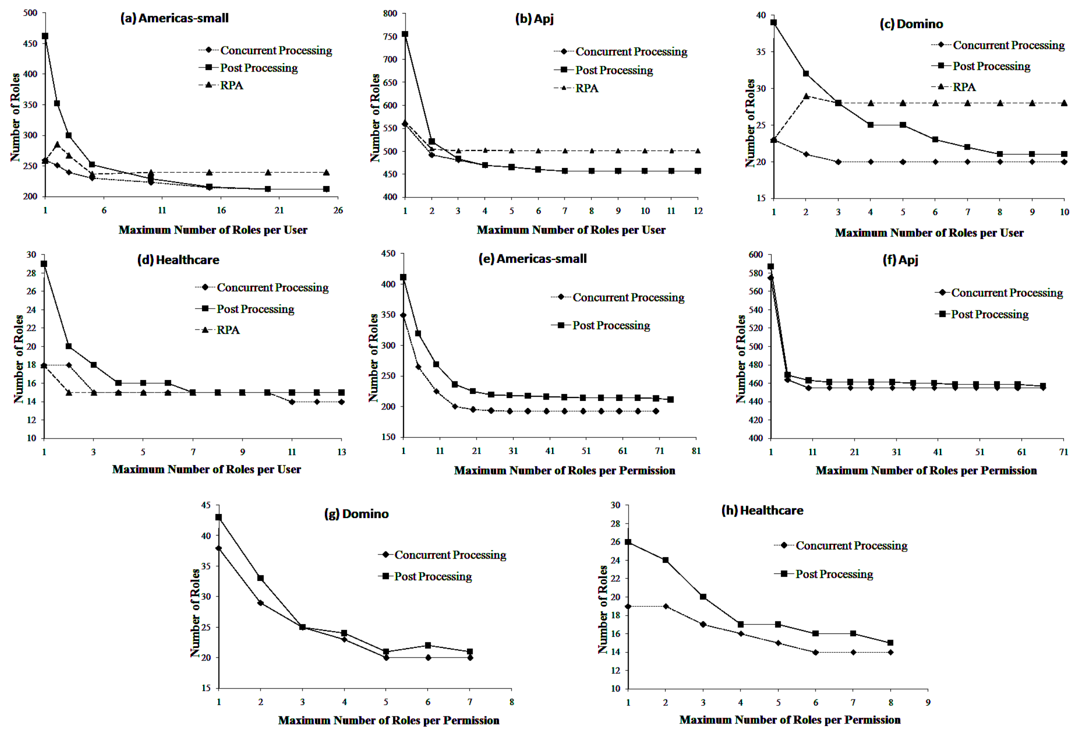


Fig. 1: Effect of role-usage (a-d) and permission-distribution (e-h) constraints on the number of generated roles for different data sets.

as Role Priority based Approach (RPA) [4] based on Boolean matrix decomposition [5] have also been shown in the same set of figures. We start with a constraint value of 1 and increase it till there is no further change in the number of roles generated. This corresponds to the point in which the constraint value becomes the same as the maximum number of roles possessed by any user in the unconstrained decomposition of the respective UPA.

The results show that the number of roles generated by both pre-processing and concurrent approaches increase as the value of the constraint is lowered. This is intuitively expected since for a lower value of the number of roles allowed per user, more number of permissions of each user need to be combined together to generate unique user specific roles. There is little scope for sharing the same role across multiple users, thereby increasing the overall total number of roles required for a consistent decomposition.

Another important observation from the figures is that, the concurrent processing framework usually generates less number of roles than the post-processing framework. The reason is that, the post-processing framework can fix the violations by only re-assigning the permissions of the violating users. New roles are formed without removing the existing ones while fixing the constraint unless there are other users sharing all the permissions of a role newly created. It does not have the flexibility of making global changes. On the other hand, the concurrent framework at each step enforces the constraints by making a greedy choice among all the users remaining to be assigned to roles. As a result, the number of generated roles is less. It is also seen from Figures 1(a)-(d) that both post-processing and concurrent processing approaches proposed in this paper generate less number of roles compared to RPA for all the four data sets.

The impact of the MRC_{perm} value on the number of roles generated for role mining under permission-distribution cardinality constraint is shown in Figures 1(e)-(h). The results follow a similar trend as in the case of role-usage cardinality constraint.

3.2 Results of Role Mining in the Presence of Two Constraints Simultaneously

In this sub-section, we present results of our experiments when both Role-usage and Permission-distribution constraints are specified simultaneously. Since the two constraints impose mutually contradictory requirements, certain combinations of MRC_{user} and MRC_{perm} can result in no valid decomposition being found that satisfies both the constraints. However, anti-monotone property holds meaning that if for a given combination of

TABLE 11: Number of roles generated for post-processing approach using different heuristics

(a) Americas-large

MRC_{perm}	MRC_{user} 6				5				4				3			
	Min	Max	UP	PU	Min	Max	UP	PU	Min	Max	UP	PU	Min	Max	UP	PU
144	421	421	421	421	422	422	422	422	424	424	424	424	x	x	x	x
140	423	422	423	423	424	423	424	424	426	425	426	426	x	x	x	x
130	423	422	423	423	424	423	424	424	426	425	426	426	x	x	x	x
120	426	422	424	424	427	423	425	425	429	425	427	427	x	x	x	x
110	430	422	426	426	431	423	427	427	433	425	429	429	x	x	x	x
100	434	427	430	430	435	428	431	431	437	431	433	433	x	x	x	x

(b) Americas-small

MRC_{perm}	MRC_{user} 22				16				12				8			
	Min	Max	UP	PU	Min	Max	UP	PU	Min	Max	UP	PU	Min	Max	UP	PU
75	211	211	211	211	217	215	216	216	227	226	223	223	245	238	243	243
70	215	212	213	213	221	216	218	218	231	227	225	225	250	239	247	245
60	216	212	214	214	222	216	219	219	232	227	226	226	252	242	248	246
50	216	212	214	214	222	216	219	219	232	227	226	226	252	246	248	247
40	219	216	216	216	225	220	221	221	234	231	228	228	x	x	x	x
30	221	218	218	218	227	222	223	223	236	234	230	230	x	x	x	x
20	235	225	225	225	245	x	237	239	x	x	x	x	x	x	x	x

(c) Apj

MRC_{perm}	MRC_{user} 11				10				8				6			
	Min	Max	UP	PU	Min	Max	UP	PU	Min	Max	UP	PU	Min	Max	UP	PU
67	456	456	456	456	457	457	457	457	459	457	458	458	463	461	465	465
65	457	457	457	457	458	458	458	458	460	458	459	459	464	462	466	466
55	460	458	459	459	461	459	460	460	463	461	463	461	x	x	471	468
45	463	459	460	460	464	460	461	461	467	464	466	462	x	x	x	x
35	463	460	460	460	464	461	462	462	468	x	x	x	x	x	x	x
25	463	460	461	461	464	461	462	462	468	x	x	x	x	x	x	x
15	463	x	x	x	464	x	x	x	x	x	x	x	x	x	x	x
5	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

(d) Domino

MRC_{perm}	MRC_{user} 11				9				7				6			
	Min	Max	UP	PU	Min	Max	UP	PU	Min	Max	UP	PU	Min	Max	UP	PU
8	20	20	20	20	21	21	21	21	23	22	23	23	23	23	23	23
7	21	21	21	21	22	22	22	22	24	23	24	24	x	x	x	x
6	22	21	22	22	23	22	23	23	25	x	25	x	x	x	x	x
5	23	22	22	22	x	23	23	23	x	x	x	x	x	x	x	x
4	25	24	25	25	x	x	x	x	x	x	x	x	x	x	x	x

(e) Firewall1

MRC_{perm}	MRC_{user} 21				16				12				8			
	Min	Max	UP	PU	Min	Max	UP	PU	Min	Max	UP	PU	Min	Max	UP	PU
26	69	69	69	69	70	70	70	70	71	71	71	71	75	72	73	73
25	70	70	70	70	71	71	71	71	72	72	72	72	76	76	74	74
21	71	70	71	71	72	71	72	72	73	72	73	73	78	x	75	75
17	75	72	73	73	76	73	74	74	77	74	75	75	x	x	x	x
13	76	73	75	75	77	74	76	76	78	75	77	77	x	x	x	x
9	76	74	76	76	77	75	77	77	78	x	80	81	x	x	x	x

(f) Firewall2

MRC_{perm}	MRC_{user} 9				8				7				6			
	Min	Max	UP	PU	Min	Max	UP	PU	Min	Max	UP	PU	Min	Max	UP	PU
3	10	10	10	10	11	11	11	11	11	11	11	11	x	x	x	x
2	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

(g) Healthcare

MRC_{perm}	MRC_{user} 7				6				5				4			
	Min	Max	UP	PU	Min	Max	UP	PU	Min	Max	UP	PU	Min	Max	UP	PU
9	15	15	15	15	16	16	16	16	16	16	16	16	17	16	16	16
8	16	16	16	16	17	17	17	17	17	17	17	17	18	17	17	17
7	16	16	16	16	17	17	17	17	17	17	17	17	18	18	18	18
6	17	17	17	17	18	18	18	18	18	18	18	18	x	x	x	x
5	18	18	18	18	19	19	19	19	x	x	x	x	x	x	x	x
4	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

MRC_{user} and MRC_{perm}, no solution can be found, for any combination with lower values of either or both of the constraints, no solution would exist. Similarly, monotonicity property also holds, i.e., if a solution exists for a given combination of MRC_{user} and MRC_{perm}, a solution would exist for any combination with higher values of either or both of the constraints.

In order to decide which combinations of MRC_{user} and MRC_{perm} should be considered for experiments, from the unconstrained decomposition of the UPA, the maximum number of roles any user had in the resulting UA and the maximum number of roles to which any permission belonged in the PA were determined. If these values are considered to be the values of MRC_{user} and MRC_{perm}, respectively, then by construction, a valid solution is known to exist for the given UPA. The constraint values were lowered starting from this initial combination and experiments were carried out. We stop when it is found that no solution exists for a given combination. By the anti-monotone property mentioned above, no solution is expected to exist for smaller values of the constraints.

3.2.1 Variation in the Number of Roles Generated

Tables 11 and 12 respectively show the number of roles generated using the post-processing and the concurrent processing frameworks on seven of the nine real world data sets. Since the EMEA data set has at most one role

TABLE 12: Number of roles generated for concurrent approach using different heuristics

(a) Americas-large

MRC_{perm}	MRC_{user} 6				5				4				3			
	NU	NP	XR	NR	NU	NP	XR	NR	NU	NP	XR	NR	NU	NP	XR	NR
144	423	421	435	521	423	421	435	513	424	421	435	503	420	417	437	489
140	423	421	435	521	423	421	435	513	424	421	435	503	420	417	437	489
130	423	421	435	521	423	421	435	513	424	421	435	503	420	417	437	489
120	424	422	435	521	424	422	435	513	425	422	435	503	x	x	437	489
110	424	421	435	521	424	421	x	513	425	421	x	503	x	x	x	489
100	427	423	435	521	426	422	x	513	426	420	x	503	x	x	x	x

(b) Americas-small

MRC_{perm}	MRC_{user} 22				16				12				8			
	NU	NP	XR	NR	NU	NP	XR	NR	NU	NP	XR	NR	NU	NP	XR	NR
75	213	212	200	245	215	217	201	245	222	219	204	255	230	227	218	254
70	213	212	200	245	215	217	201	245	222	219	204	255	230	227	218	254
60	213	212	200	245	215	217	201	245	222	219	204	255	230	227	218	254
50	213	212	200	245	215	217	201	245	222	219	204	255	x	x	218	254
40	213	212	200	245	215	217	201	245	222	219	204	255	x	x	218	254
30	213	212	200	245	215	215	201	245	x	x	x	255	x	x	x	254
20	217	217	202	245	219	220	203	245	x	x	x	255	x	x	x	x

(c) Apj

MRC_{perm}	MRC_{user} 11				10				8				6			
	NU	NP	XR	NR	NU	NP	XR	NR	NU	NP	XR	NR	NU	NP	XR	NR
67	456	457	454	484	456	457	454	484	458	462	454	484	461	x	460	484
65	456	457	454	484	456	x	454	484	458	x	454	484	x	x	460	484
55	456	457	455	484	456	x	455	484	x	x	x	484	x	x	x	484
45	x	x	x	484	x	x	x	484	x	x	x	484	x	x	x	484
35	x	x	x	484	x	x	x	484	x	x	x	484	x	x	x	484
25	x	x	x	484	x	x	x	484	x	x	x	484	x	x	x	484
15	x	x	x	484	x	x	x	484	x	x	x	484	x	x	x	484
5	x	x	x	482	x	x	x	482	x	x	x	482	x	x	x	x

(d) Domino

MRC_{perm}	MRC_{user} 11				9				7				6			
	NU	NP	XR	NR	NU	NP	XR	NR	NU	NP	XR	NR	NU	NP	XR	NR
8	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
7	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
6	20	20	21	20	20	20	21	20	20	20	21	20	20	20	21	20
5	20	20	21	20	20	20	21	20	20	20	21	20	20	20	21	20
4	x	24	24	21	x	x	x	21	x	x	x	21	x	x	x	21

(e) Firewall1

MRC_{perm}	MRC_{user} 21				16				12				8			
	NU	NP	XR	NR	NU	NP	XR	NR	NU	NP	XR	NR	NU	NP	XR	NR
26	69	69	65	69	69	69	65	69	69	69	66	69	x	x	70	69
25	70	70	65	69	70	70	65	69	70	70	66	69	x	x	70	69
21	69	69	65	69	69	69	65	69	69	69	66	69	x	x	70	69
17	69	69	65	69	69	69	65	69	69	69	66	69	x	x	x	69
13	69	69	65	69	69	69	65	69	69	69	66	69	x	x	x	69
9	69	69	65	69	69	69	65	69	69	69	66	69	x	x	x	69

(f) Firewall2

MRC_{perm}	MRC_{user} 9				8				7				6			
	NU	NP	XR	NR	NU	NP	XR	NR	NU	NP	XR	NR	NU	NP	XR	NR
3	10	10	10	12	10	10	10	12	10	10	10	12	x	x	x	12
2	10	10	10	11	10	10	10	11	x	x	10	11	x	x	x	11

(g) Healthcare

MRC_{perm}	MRC_{user} 7				6				5				4			
	NU	NP	XR	NR	NU	NP	XR	NR	NU	NP	XR	NR	NU	NP	XR	NR
9	15	15	14	14	15	15	14	14	15	15	14	14	14	14	14	14
8	15	15	14	14	15	15	14	14	x	x	14	14	x	x	14	14
7	15	15	14	14	15	15	14	14	x	x	14	14	x	x	14	14
6	15	15	14	14	15	15	14	14	x	x	14	14	x	x	14	14
5	15	15	14	15	15	15	x	15	x	x	x	15	x	x	x	x
4	15	15	x	x	x	x	x	x	x	x	x	x	x	x	x	x

for each user and the Customer data set has at most one role for each permission, these two data sets cannot be meaningfully used for studying the presence of the two constraints together and hence, are excluded from this set of results. For the post-processing framework, we show results for the four heuristics mentioned in Sub-section 3.3 and used in Algorithm 2, namely, *Min*, *Max*, *UP* and *PU*. Similarly, for the concurrent framework, results are presented for the four heuristics mentioned in Sub-section 4.3 and used in Algorithm 4, namely, *NU*, *NP*, *XR* and *NR*.

MRC_{user} is varied along the columns and MRC_{perm} is varied along the rows in the tables. A "$\times$" in any cell indicates that no valid decomposition could be obtained for that combination of constraint values using the corresponding algorithm.

In Tables 11(a)-(g), it is observed that, for any given data set and a chosen heuristic, the number of roles increases or remains unchanged in the post-processing framework as the values of the two constraints are reduced. While fixing one violation at a time in each step of the iteration, the four heuristics choose the next violation in different ways. Since there is no backtracking, the algorithm after fixing some of the violations could be left with a subset of violations that cannot be fixed. A different sequence of choices in the earlier stages (by a different heuristic) might not have yielded such a state and all the violations could have been fixed when the

TABLE 13: Execution time

(a) Americas-large (in sec.)

MRC_{perm}	MRC_{user} 6		5		4		3	
	p	c	p	c	p	c	p	c
144	0.06	194.9	0.12	188.5	0.23	188.2	x	180.6
140	0.17	196.0	0.23	187.6	0.34	185.0	x	181.1
130	0.18	195.4	0.23	187.4	0.35	185.7	x	179.9
120	0.35	195.9	0.4	189.4	0.51	185.5	x	182.8
110	0.59	194.5	0.63	190.7	0.74	187.0	x	181.6
100	0.81	198.1	0.85	189.4	0.95	184.6	x	x

(b) Americas-small (in sec.)

MRC_{perm}	MRC_{user} 22		16		12		8	
	p	c	p	c	p	c	p	c
75	0.02	14.4	0.07	14.3	0.18	14.9	0.35	15.0
70	0.06	14.4	0.11	13.9	0.22	14.5	0.41	14.5
60	0.07	14.5	0.13	13.9	0.23	14.6	0.44	14.6
50	0.07	14.4	0.13	14.0	0.24	14.8	0.44	14.6
40	0.1	14.3	0.16	13.9	0.25	14.5	x	14.5
30	0.12	14.4	0.19	14.0	0.28	14.5	x	14.5
20	0.3	14.40	0.38	13.96	x	14.58	x	x

(c) Apj (in sec.)

MRC_{perm}	MRC_{user} 11		10		8		6	
	p	c	p	c	p	c	p	c
67	0.02	11.8	0.03	11.9	0.07	11.9	0.12	11.9
65	0.04	12.0	0.05	11.7	0.08	11.7	0.14	11.9
55	0.09	12.1	0.09	11.7	0.13	11.7	x	11.7
45	0.13	11.9	0.14	11.7	0.19	11.7	x	11.7
35	0.13	12.0	0.14	11.6	0.2	11.7	x	11.7
25	0.13	12.0	0.14	11.7	0.2	11.6	x	11.7
15	0.13	11.9	0.14	11.7	x	11.7	x	11.7
5	x	12.0	x	11.9	x	11.7	x	11.6

(d) Domino (in msec.)

MRC_{perm}	MRC_{user} 11		9		7		6	
	p	c	p	c	p	c	p	c
8	0.12	5.58	0.25	4.09	0.51	4.21	0.5	4.13
7	0.25	4.11	0.38	4.14	0.65	4.98	x	4.08
6	0.38	4.85	0.51	4.47	0.75	5.21	x	4.61
5	0.48	5.27	x	6.75	x	7.76	x	4.43
4	0.71	4.78	x	4.81	x	6.54	x	7.52

(e) Firewall1 (in msec.)

MRC_{perm}	MRC_{user} 21		16		12		8	
	p	c	p	c	p	c	p	c
26	1.22	192.7	2.52	192.3	3.77	193.2	8.70	192.2
25	2.46	193.6	3.80	187.7	5.01	186.8	9.95	187.3
21	3.81	193.4	2.55	186.4	6.23	188.3	8.77	187.5
17	8.86	192.5	10.2	179.8	11.4	188.8	x	190.1
13	10.2	193.1	11.6	193.5	12.3	193.4	x	186.9
9	10.1	193.0	8.83	189.8	12.6	187.8	x	187.9

(f) Firewall2 (in msec.)

MRC_{perm}	MRC_{user} 9		8		7		6	
	p	c	p	c	p	c	p	c
3	0.14	31.4	0.37	31.5	0.38	31.4	x	31.5
2	x	30.7	x	28.0	x	28.0	x	27.7

(g) Healthcare (in msec.)

MRC_{perm}	MRC_{user} 7		6		5		4	
	p	c	p	c	p	c	p	c
9	0.04	0.88	0.09	0.86	0.09	0.86	0.13	0.90
8	0.09	0.87	0.13	0.87	0.14	0.87	0.18	0.90
7	0.09	0.86	0.14	0.87	0.14	0.87	0.18	0.81
6	0.13	0.86	0.19	0.87	0.18	0.86	x	0.91
5	0.17	1.10	0.23	1.09	x	1.10	x	x
4	x	x	x	x	x	x	x	x

algorithm terminated. It is seen from the results presented in the tables that the *Min* heuristic, i.e., choosing the user or permission with the minimum number of violations at any iteration step of Algorithm 2, has better performance as compared to the other heuristics in terms of the ability to fix constraints for smaller values of MRC_{user} and MRC_{perm}. The number of roles generated by this heuristic is also comparable with the other ones.

Similarly, from Tables 12(a)-(g), it is observed that the number of roles usually increases or remains unchanged in the concurrent framework as the values of the two constraints are reduced. It is also seen from the table that the three heuristics *NU*, *NP* and *XR* have similar performance in terms of the number of roles generated as well as the ability to handle lower values of constraints. Although the number of roles generated by the *NR* heuristic is higher than the rest, it has much better performance in terms of the ability to fix constraints for smaller values of MRC_{user} and MRC_{perm}. For example, for the Domino, Firewall1 and Firewall2 data sets, it can enforce constraints for all the combinations shown in the tables. For Americas-large, Americas-small and Apj, it can handle all the combinations except the last one.

It is further observed from Tables 11 and 12 that the three heuristics *NU*, *NP* and *XR* of the concurrent approach, in general, have better performance compared to all the four heuristics of the post-processing approach in terms of the number of roles generated. This is similar to the observation made in the previous subsection where only one constraint was considered at a time. Further, the post-processing approach fails to provide a solution to certain combinations of constraints for which the solution exists as is evidenced by the results from the concurrent approach. For example, none of the heuristics of the post-processing approach can provide a solution for the Americas-large data set for all values of MRC_{perm} when MRC_{user}=3. All the heuristics of the concurrent approach can provide a solution till MRC_{perm}=130 for MRC_{user}=3. The poorer performance of the post-processing approach is due to the fact that it starts with the given UA and PA matrix and only tries to fix the violations without bringing in any new violation.

It may also be noted that in Tables 12(a)-(g), sometimes the number of roles decreases even though the constraint values are reduced. The original UPA matrices for all the data sets were analyzed with respect to the distribution of the number of permissions over users and that of users over permissions. It was observed that, there are clusters of such coverage numbers. For example, in the Americas-large data set, there are 20 permissions each of which has 2804 users. There is an abrupt drop after this with the permission having the next highest number of users has 178 users. There are three permissions with 178, 165 and 164 users, followed by two with 162 users each. This is once again followed by 11 permissions each with 161 users. When there is a tie, the concurrent processing algorithm chooses one of the alternatives at random, which causes occasional decrease in the number of roles even if the values of the constraints are reduced. Since, such types of abruptness do not exist in case of the post-processing approach which starts with the UA and PA already obtained from an initial decomposition, Tables 11(a)-(g) do not show such unexpected variations.

Based on the above observations, we select the *Min* heuristic of the post-processing approach and the *NR* heuristic of the concurrent approach for further evaluation in the next sub-sections.

3.2.2 Variation in Execution Time

In Tables 13(a)-(g), we show the variation in execution time required in the post-processing and concurrent processing frameworks for the seven data sets mentioned above. In these tables and also in those of the next sub-section, P denotes post-processing approach using the *Min* heuristic and C denotes concurrent approach using the *NR* heuristic.

The results show that the execution time in the post-processing approach increases slowly with decrease in the values of the constraints. The execution time for the concurrent approach, on the other hand, remains more or less constant. It is also observed that the concurrent processing approach takes more time than the post-processing approach for all the data sets. Even if the basic unconstrained decomposition time shown in Table 10 is added to the time for fixing the constraints in the post-processing approach, the total time is less than the concurrent processing time. However, as observed from the results in the previous sub-section, the concurrent processing approach works for tighter (smaller) values of the constraints where the post-processing approach fails. As an example, for the Apj data set, with MRC_{user} = 8 and MRC_{perm} = 25, the post-processing approach takes a total of 3.64+0.2=3.84 seconds for generating the roles. For the same combination, the concurrent processing approach takes 11.6 seconds. The number of roles generated are 468 and 484, respectively. However, as the constraint on permission-distribution is reduced to 15, the post-processing approach fails to fix the same. On the other hand, the concurrent approach gives a valid solution with 484 roles needing 11.7 seconds.

It may be noted here that the time taken by the post-processing approach to fix the constraint violations depends on the number of violations present in the initial decomposition provided as input. For example, for the Americas-small dataset, an initial decomposition with the minimum biclique cover based approach [2] contains 98 users who have greater than 8 roles each and 55 permissions each of which belongs to more than 20 roles. For Firewall1, 5 users have 8 or more roles and 79 permissions belong to more than 9 roles. A different unconstrained role mining algorithm would generate UA and PA matrices with potentially different number of violating users and permissions. The execution time for the post-processing approach would accordingly vary.

3.3 Comparison with Graph Optimization Algorithm

As mentioned in Section 3, the initial decomposition for the post-processing approach can be made using any existing role mining algorithm. For the results reported in the last two sub-sections, we had considered the minimum biclique cover based approach proposed in [2] for generating the UA and PA matrices. This was done to provide a meaningful comparison with the concurrent approach, which is also based on the same algorithm. In this section, we perform a comparative analysis between the minimum biclique cover based post-processing and concurrent approaches with another pair of post-processing and concurrent approaches that are based on the graph optimization algorithm presented in [3]. This algorithm forms role hierarchies by splitting and merging of roles.

It may be noted that, besides using the number of roles as the objective function to minimize, recent research in role mining has focused on optimizing other metrics like the Weighted Structural Complexity (WSC) [6]. The WSC metric is a weighted sum of the number of roles, user-role assignments, role-permission assignments, elements in the role hierarchy relation and the number of direct user to permission assignments.

In Tables 14(a)-(g), we show the variation in the number of generated roles by the *Min* heuristic of the proposed post-processing approach (denoted as P in the table), *NR* heuristic of the proposed concurrent approach (denoted as C in the table), graph optimization based concurrent approach (*CGO*) and graph optimization based post-processing approach (*PGO*). Thus, the first two values for each MRC_{user}-MRC_{perm} combination in these tables are taken from the respective columns of Tables 11(a)-(g) and 12(a)-(g).

PGO initially generates the UA, PA and RA (role-to-role relationship representing role hierarchy) matrices from a given UPA matrix. Every user is assigned to a single role but through role hierarchy he can obtain all the required permissions. It then fixes the permission-distribution constraint violation by revising the RA through merging/splitting of roles. Since a user is directly assigned to one role only, the role-usage constraint has no effect on PGO. Hence, the two-constraint problem reduces to a single constraint problem in this case. The concurrent graph optimization approach (CGO), on the other hand, enforces the constraints at the time of forming the set of roles and the role hierarchy. Here also, the obtained decomposition contains only one role per user and hence, the role-usage constraint has no effect.

From the tables, it is observed that both PGO and CGO can produce a solution to the constrained role mining problem for all values of the constraints. This is expected because, each user is only assigned to one role directly and the problem essentially reduces to a single constraint problem. All the necessary permissions of a user are made available through the role hierarchy, which is not affected by the cardinality constraint. However, the number of roles generated for both PGO and CGO is much higher compared to the proposed post-processing and concurrent approaches. This is due to the formation of a large number of roles through repeated splitting and merging in the graph optimization approach. It is also observed that in PGO, the number of roles generated remains unchanged for all the data sets except for Firewall2 even when the cardinality constraint on permission is changed (No change across columns in the same row for either CGO or PGO is expected since these approaches always generate one role per user and, therefore, variation in the MRC_{user} value has no effect on the number of roles). This is because of the fact that the maximum number of roles to which any permission belongs in the initial unconstrained decomposition of the

TABLE 14: Comparative performance of the proposed approaches (Post-processing - P and Concurrent - C) with graph optimization based approaches (Post processing graph optimization - PGO and Concurrent graph optimization - CGO) in terms of the number of roles generated

(a) Americas-large

MRC_{perm}	MRC_{user}: 6				5				4				3			
	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO
144	421	521	1051	1501	422	513	1051	1501	424	503	1051	1501	x	489	1051	1501
140	423	521	1082	1501	424	513	1082	1501	426	503	1082	1501	x	489	1082	1501
130	423	521	1086	1501	424	513	1086	1501	426	503	1086	1501	x	489	1086	1501
120	426	521	1156	1501	427	513	1156	1501	429	503	1156	1501	x	489	1156	1501
110	430	521	1216	1501	431	513	1216	1501	433	503	1216	1501	x	489	1216	1501
100	434	521	1246	1501	435	513	1246	1501	437	503	1246	1501	x	x	1246	1501

(b) Americas-small

MRC_{perm}	MRC_{user}: 22				16				12				8			
	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO
75	211	245	469	382	217	245	469	382	227	255	469	382	245	254	469	382
70	215	245	464	382	221	245	464	382	231	255	464	382	250	254	464	382
60	216	245	464	382	222	245	464	382	232	255	464	382	252	254	464	382
50	216	245	464	382	222	245	464	382	232	255	464	382	252	254	464	382
40	219	245	464	382	225	245	464	382	234	255	464	382	x	254	464	382
30	221	245	464	382	227	245	464	382	236	255	464	382	x	254	464	382
20	235	245	464	382	245	245	464	382	x	255	464	382	x	x	464	382

(c) Apj

MRC_{perm}	MRC_{user}: 11				10				8				6			
	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO
67	456	484	1538	484	457	484	1538	484	459	484	1538	484	463	484	1538	484
65	457	484	1536	484	458	484	1536	484	460	484	1536	484	464	484	1536	484
55	460	484	1534	484	461	484	1534	484	463	484	1534	484	x	484	1534	484
45	463	484	1532	484	464	484	1532	484	467	484	1532	484	x	484	1532	484
35	463	484	888	484	464	484	888	484	468	484	888	484	x	484	888	484
25	463	484	529	484	464	484	529	484	468	484	529	484	x	484	529	484
15	463	484	519	484	464	484	519	484	x	484	519	484	x	484	519	484
5	x	482	489	484	x	482	489	484	x	482	489	484	x	x	489	484

(d) Domino

MRC_{perm}	MRC_{user}: 11				9				7				6			
	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO
8	20	20	32	32	21	20	32	32	23	20	32	32	23	20	32	32
7	21	20	31	32	22	20	31	32	24	20	31	32	x	20	31	32
6	22	20	28	32	23	20	28	32	25	20	28	32	x	20	31	32
5	23	20	27	32	x	20	27	32	x	20	27	32	x	20	27	32
4	25	21	27	32	x	21	27	32	x	21	27	32	x	21	27	32

(e) Firewall1

MRC_{perm}	MRC_{user}: 21				16				12				8			
	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO
26	69	69	116	86	70	69	116	86	71	69	116	86	75	69	116	86
25	70	69	116	86	71	69	116	86	72	69	116	86	76	69	116	86
21	71	69	113	86	72	69	113	86	73	69	113	86	78	69	113	86
17	75	69	110	86	76	69	110	86	77	69	110	86	x	69	110	86
13	76	69	106	86	77	69	106	86	78	69	106	86	x	69	106	86
9	76	69	102	86	77	69	102	86	78	69	102	86	x	69	102	86

(f) Firewall2

MRC_{perm}	MRC_{user}: 9				8				7				6			
	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO
3	10	12	14	12	11	12	14	12	11	12	14	12	x	12	14	12
2	x	11	13	14	x	11	13	14	x	11	13	14	x	11	13	14

(g) Healthcare

MRC_{perm}	MRC_{user}: 7				6				5				4			
	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO
9	15	14	27	22	16	14	27	22	16	14	27	22	17	14	27	22
8	16	14	26	22	17	14	26	22	17	14	26	22	18	14	26	22
7	16	14	25	22	17	14	25	22	17	14	25	22	18	14	25	22
6	17	14	24	22	18	14	24	22	18	14	24	22	x	14	24	22
5	18	15	23	22	19	15	23	22	x	15	23	22	x	x	23	22
4	x	x	23	22	x	x	23	22	x	x	23	22	x	x	23	22

UPA matrices using the graph optimization method for the various data sets are as follows: 35 for Americas-large, 18 for Americas-small, 6 for Apj, 4 for Domino, 6 for Firewall1, 3 for Firewall2 and 4 for Healthcare. The MRC_{perm} values considered in the tables are above these except for Apj and Firewall2. Even for Apj, it is close to the smallest MRC_{perm} value considered, i.e., 5 as a result of which there was no change in the number of roles generated. For Firewall2, however, a change in the generated number of roles is observed as the value of MRC_{perm} is changed from 3 to 2.

For CGO, no definite pattern is observed as the value of MRC_{perm} is changed for the different data sets. In this approach, whether the number of roles would increase or decrease with changing value of the constraint depends on the data set under consideration and the order in which roles are chosen for merging/splitting. As has been noted in [3] as well, depending on the scenario, the number of roles can either increase or decrease or remain unchanged. The number of roles and corresponding role hierarchy after mining may be different even for the same data set if the order in which the roles are selected are different.

Finally, in Tables 15(a)-(g), we show the variation in

TABLE 15: Comparative performance of the proposed approaches (Post-processing - P and Concurrent - C) with graph optimization based approaches (Post processing graph optimization - PGO and Concurrent graph optimization - CGO) in terms of WSC metric

(a) Americas-large

MRC_{perm}	MRC_{user} 6				5				4				3			
	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO
144	106560	70943	105794	38708	106484	82103	105794	38708	106431	80001	105794	38708	x	83038	105794	38708
140	105161	70943	103284	38708	105085	82103	103284	38708	105032	80001	103284	38708	x	83038	103284	38708
130	104264	70943	92290	38708	104188	82103	92290	38708	104135	80001	92290	38708	x	83038	92290	38708
120	101422	70943	85756	38708	101346	82103	85756	38708	100517	80001	85756	38708	x	83038	85756	38708
110	100289	70943	78975	38708	100213	82103	78975	38708	100160	80001	78975	38708	x	83038	78975	38708
100	100515	70943	74818	38708	100439	82103	74818	38708	100152	80001	74818	38708	x	x	74818	38708

(b) Americas-small

MRC_{perm}	MRC_{user} 22				16				12				8			
	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO
75	25088	14703	18386	8633	24461	14703	18366	8633	23929	15110	18366	8633	20628	15957	18366	8633
70	24011	14703	18386	8633	23384	14703	18366	8633	22852	15110	18366	8633	22765	15957	18366	8633
60	23570	14703	18366	8633	22943	14703	18366	8633	22411	15110	18366	8633	23089	15957	18366	8633
50	24398	14703	18366	8633	23771	14703	18366	8633	23239	15110	18366	8633	22952	15957	18366	8633
40	23464	14703	18366	8633	22837	14703	18366	8633	22433	15110	18366	8633	x	15957	18366	8633
30	23333	14703	18366	8633	22706	14703	18366	8633	223311	15110	18366	8633	x	15957	18366	8633
20	21724	14703	18366	8633	21191	14703	18366	8633	x	15110	18366	8633	x	x	18366	8633

(c) Apj

MRC_{perm}	MRC_{user} 11				10				8				6			
	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO
67	6188	5047	8398	4097	6071	5047	8398	4097	5983	5047	8398	4097	5971	5046	8398	4097
65	6161	5047	8388	4097	6071	5047	8388	4097	5995	5047	8388	4097	5908	5046	8388	4097
55	6076	5047	8354	4097	5988	5047	8354	4097	5974	5047	8354	4097	x	5046	8354	4097
45	5991	5047	8320	4097	5991	5047	8320	4097	5991	5047	8320	4097	x	5046	8320	4097
35	5953	5047	6237	4097	5937	5047	6237	4097	5933	5047	6237	4097	x	5046	6237	4097
25	5874	5047	5267	4097	5859	5047	5267	4097	5831	5047	5267	4097	x	5046	5267	4097
15	5781	5047	5247	4097	5765	5047	5247	4097	x	5047	5247	4097	x	5046	5247	4097
5	x	5347	5179	4092	x	5347	5179	4092	x	5347	5179	4092	x	x	5179	4092

(d) Domino

MRC_{perm}	MRC_{user} 11				9				7				6			
	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO
8	811	762	567	596	811	762	567	596	807	762	567	596	809	762	567	596
7	714	762	563	596	714	762	563	596	709	762	563	596	x	762	563	596
6	618	762	553	596	706	762	553	596	711	762	553	596	x	762	553	596
5	613	762	551	596	x	762	551	596	x	762	551	596	x	762	551	596
4	617	668	551	596	x	668	551	596	x	668	551	596	x	668	551	596

(e) Firewall1

MRC_{perm}	MRC_{user} 21				16				12				8			
	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO
26	6239	2688	5351	1684	6266	2688	5351	1684	6078	2688	5351	1684	5413	2688	5351	1684
25	6028	2688	5343	1684	6055	2688	5343	1684	5869	2688	5343	1684	5253	2688	5343	1684
21	5553	2688	4852	1684	5580	2688	4852	1684	5701	2688	4852	1684	5705	2688	4852	1684
17	5171	2688	4520	1684	5178	2688	4520	1684	5225	2688	4520	1684	x	2688	4520	1684
13	4887	2688	4188	1684	4914	2688	4188	1684	4933	2688	4188	1684	x	2688	4188	1684
9	4605	2688	3856	1684	4612	2688	3856	1684	4633	2688	3856	1684	x	2688	3856	1684

(f) Firewall2

MRC_{perm}	MRC_{user} 9				8				7				6			
	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO
3	1858	1803	1742	1212	1780	1803	1742	1212	1713	1803	1742	1212	x	1803	1742	1212
2	x	1808	1739	1216	x	1808	1739	1216	x	1808	1739	1216	x	1808	1739	1216

(g) Healthcare

MRC_{perm}	MRC_{user} 7				6				5				4			
	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO	P	C	CGO	PGO
9	480	316	409	168	461	316	409	168	448	316	409	168	437	316	409	168
8	455	316	388	168	436	316	388	168	423	316	388	168	418	316	388	168
7	438	316	385	168	426	316	385	168	406	316	385	168	402	316	385	168
6	424	316	382	168	412	316	382	168	399	316	382	168	x	316	382	168
5	407	324	379	168	403	324	379	168	x	324	379	168	x	321	379	168
4	x	x	378	168	x	x	378	168	x	x	378	168	x	x	378	168

the WSC metric for the same set of approaches as given in Tables 14(a)-(g). For PGO, the WSC values show the same type of variation as observed for the number of roles in Tables 14(a)-(g) except that a minor variation in WSC is observed for the Apj data set as the value of MRC_{perm} is changed from 15 to 5. Other than PGO, the proposed concurrent approach has the smallest value of WSC for five of the data sets (i.e., all the data sets except Domino and Firewall2) for all combinations of MRC_{user} and MRC_{perm}. However, although PGO has lower WSC values, the number of roles generated is much higher as observed from Tables 14(a)-(g).

Thus, comparing Tables 14 and 15, it is observed that the proposed concurrent approach has lower WSC values compared to the post-processing approach and it can also satisfy more number of constraints. The number of roles generated is also much lower than the CGO and PGO approaches. However, the time taken by the proposed post-processing approach is much lower as reported in Table 13.

Hence, it can be concluded that a meaningful strategy for meeting cardinality constraints in role mining would be to do an initial unconstrained decomposition of a given UPA matrix into UA and PA matrices, and then follow a post-processing approach to fix the violations for the given pair of MRC_{user} and MRC_{perm} constraints. If no valid solution can be generated while trying to fix the violations, the concurrent approach should be tried on the initial UPA matrix itself. The concurrent processing heuristics should be tried in the order NP, NU, XR and NR.

REFERENCES

[1] I. Molloy, N. Li, T. Li, Z. Mao, Q. Wang and J. Lobo: Evaluating Role Mining Algorithms. In: Proceedings of the 14th ACM Symposium on Access control Models and Technologies, pp. 95-104 (2009)

[2] E. Alina, H. William, M. Nikola, R. Prasad, R. Schreiber and R. E. Tarjan: Fast Exact and Heuristic Methods for Role Minimization Problems. In: Proceedings of the 13th ACM Symposium on Access Control Models and Technologies, pp. 1-10. (2008)

[3] D. Zhang, R. Kotagiri and E. Tim: Role Engineering using Graph Optimization. In: Proceedings of the 12th ACM Symposium on Access Control Models and Technologies, pp. 139-144. (2007)

[4] J. C. John, S. Sural, V. Atluri and J. Vaidya: Role Mining under Role-Usage Cardinality Constraint. In: Proceedings of the 27th International Conference on Information Security and Privacy (2012)

[5] H. Lu, J. Vaidya and V. Atluri: Optimal Boolean Matrix Decomposition: Application to Role Engineering. In: Proceedings of the IEEE 24th International Conference on Data Engineering, pp. 297-306. (2008)

[6] I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. B. Calo and J. Lobo: Mining Roles with Multiple Objectives. In: ACM Transactions on Information and System Security 13(4), 36 (2010)