Generalized Noise Role Mining

Jason Crampton
Royal Holloway, University of London
Egham, United Kingdom
jason.crampton@rhul.ac.uk

Eduard Eiben
Royal Holloway, University of London
Egham, United Kingdom
daniel.karapetyan@nottingham.ac.uk

Gregory Gutin
Royal Holloway, University of London
Egham, United Kingdom
diptapriyo.majumdar@rhul.ac.uk

Diptapriyo Majumdar
Royal Holloway, University of London
Egham, United Kingdom
diptapriyo.majumdar@rhul.ac.uk

Daniel Karapetyan
University of Nottingham
Nottingham, United Kingdom
daniel.karapetyan@nottingham.ac.uk

ABSTRACT
Role mining seeks to compute a set of roles $R$, a user-role authorization relation $UA$ and a permission-role authorization relation $PA$, given a user-permission authorization relation $UPA$, and is therefore a core problem in the specification of role-based authorization policies. Role mining is known to be hard in general and exact solutions are often impossible to obtain, so there exists an extensive literature on variants of the role mining problem that seek to find approximate solutions and algorithms that use heuristics to find reasonable solutions efficiently.

In this paper, we introduce the Generalized Noise Role Mining problem (GNRM) – a generalization of the MinNoise Role Mining problem – which we believe has considerable practical relevance. In particular, GNRM can produce “security-aware” or “availability-aware” solutions. Extending work of Fomin et al., we show that GNRM is fixed-parameter tractable, with parameter $r+k$, where $r$ is the number of roles in the solution and $k$ is the number of discrepancies between $UPA$ and the relation defined by the composition of $UA$ and $PA$. We also introduce a further variant of GNRM in which the accuracy of the solution is defined by the number of users and permissions that are affected, rather than the number of individual discrepancies $k$. We show that this variant of GNRM is also fixed-parameter tractable.

We then report the results of our experimental work using general-purpose solvers to solve instances of GNRM. Our key findings are that security-aware role mining seems to be easier than availability-aware role mining, based on reasonable assumptions about $UPA$, and security-aware role mining introduces a similar number of discrepancies to MinNoise role mining.

CCS CONCEPTS
• Security and privacy → Formal security models; • Theory of computation → Problems, reductions and completeness; Parameterized complexity and exact algorithms.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Woodstock ’18, June 03–05, 2018, Woodstock, NY © 2018 Association for Computing Machinery. ACM ISBN 978-1-4503-3888-8/18/06 . $15.00 https://doi.org/10.1145/3241898.324456

1 INTRODUCTION
Role-based access control (RBAC) [16] is a mature, standardized [2] and widely deployed means of enforcing authorization requirements in a multi-user computer system.

Authorization policies, in effect, specify which interactions are authorized between users of and resources provided by a computer system. Such policies are used by the system’s access control mechanism to control interactions between users and resources. RBAC policies authorize users for roles and roles for resources (usually referred to as “permissions” in the context of RBAC). RBAC can significantly reduce the administrative burden of specifying and maintaining authorization policies, provided the set of roles is small compared to the number of users and permissions.

The problem of identifying a suitable set of roles for an RBAC system has been studied extensively over the last 25 years. Role engineering is a top-down approach that seeks to identify roles by decomposing and analyzing business processes [15]. This approach does not generally scale well and requires substantial human effort [12]. Role mining, the bottom-up approach, attempts to discover a set of roles from a given authorization policy that associates users directly with permissions. More formally, the Role Mining Problem is defined as follows:

**Role Mining Problem (RMP)**

**Input:** A set of users $U$, a set of permissions $P$, a user-permission assignment relation $UPA \subseteq U \times P$, and a natural number $r$.

**Goal:** Find a set $R$ of at most $r$ roles, a user-role relation $UA \subseteq U \times R$, a role-permission assignment relation $PA \subseteq R \times P$ such that $(u,p) \in UPA$ if and only if there is $\rho \in R$ such that $(u,\rho) \in UA$ and $(\rho,p) \in PA$.

The value of $r$, for solutions that are of practical use, will be small compared to the sizes of $U$ and $P$. However, it may be impossible to find a solution to RMP in which $r$ is sufficiently small. Hence,
approximate solutions are often sought, in which \( r \) is small and the composition of \( UA \) and \( PA \) is close, in some suitable sense, to \( UPA \).

A substantial literature now exists on role mining. The problem is known to be hard in general and usually impossible to solve exactly (assuming the number of roles must be small relative to the number of users), so many approximate and heuristic techniques have been developed (see the survey paper of Mitra et al. [12]).

Recent work by Fomin et al. [3] has shown that a particular, well-known variant of the role mining problem is \textit{fixed-parameter tractable} (FPT). Informally, this variant is \( NP \)-hard, like many role mining problems, so any exact algorithm to solve the problem is unlikely to be polynomial in the size of the problem’s input, unless \( P = NP \). However, there exists an algorithm (an \textit{FPT algorithm}) whose running time is exponential in some of the input parameters, but polynomial in the others. Thus, this algorithm may well be effective if the relevant parameters are small in instances of the problem that arise in practice.

Informally, the problem considered by Fomin et al. takes a relation \( UPA \subseteq U \times P \) and natural numbers \( r \) and \( k \) as input. The goal is to find a set of roles \( R \) of cardinality less than or equal to \( r \), and relations \( UA \subseteq U \times R \) and \( PA \subseteq R \times P \) such that \(|UPA \Delta (UA \circ PA)| \leq k\) (where \( UA \circ PA \) denotes the composition of relations \( UA \) and \( PA \) and \( \Delta \) denotes symmetric set difference). In other words, the composition of \( UA \) and \( PA \) has to be similar (as defined by \( k \)) to \( UPA \). The assumption is that \( k \) and \( r \) will be small parameters. This problem has been studied by the RBAC community and is usually known as the \textit{MinNoise Role Mining Problem} (MNRP) [12].

One potential problem with MNRP is that it doesn’t distinguish between (a) an element that is in \( UPA \) but not in \( UA \circ PA \) (which means some user is no longer authorized for some permission), and (b) an element that is in \( UA \circ PA \) and not in \( UPA \) (which means that some user is now incorrectly authorized for some permission). We believe that in certain situations it will be important to insist that no additional authorizations are introduced by role mining (what we will refer to as \textit{security-aware} role mining), while in other situations we may require that no authorizations are lost by role mining (\textit{availability-aware} role mining).

In this paper, we introduce the \textit{GenNoise Role Mining} (GNRM), of which MNRP is a special case. Moreover, \textit{Security-aware Role Mining} and \textit{Availability-aware Role Mining} are also special cases. We extend the results of Fomin et al. by proving that GNRM is also FPT with parameter \( k + r \).

Our other theoretical contribution is to introduce a further variant of GNRM which may be useful in situations where no solution exists for small values of \( r \) and \( k \). Specifically, we replace \( k \) with parameters \( k' \) and \( k'' \) and require that \( UA \circ PA \) is identical to \( UPA \) except in at most \( k' \) rows and \( k'' \) columns.

The paper also includes results from our experimental work in which we use general-purpose solvers to solve instances of GNRM. These results show that it is often impossible to find solutions to GNRM (unless \( k \) and \( r \) are relatively large, in which case running times increase dramatically). The results also show that security-aware role mining is more efficient than availability-aware and MinNoise role mining, assuming \( UPA \) is relatively sparse. Moreover, security-aware role mining has a similar degree of accuracy to MinNoise role mining.

The remainder of this section contains essential background material and defines GNRM.

### 1.1 Parameterized complexity

An instance of a parameterized problem \( \Pi \) is a pair \((l, \kappa)\) where \( l \) is the \textit{main part} and \( \kappa \) is the \textit{parameter}; the latter is usually a non-negative integer. A parameterized problem is \textit{fixed-parameter tractable} (FPT) if there exists a computable function \( f \) such that any instance \((l, \kappa)\) can be solved in time \( O(f(\kappa)|I|^c) \), where \(|I|\) denotes the size of \( I \) and \( c \) is an absolute constant. An algorithm to solve the problem with this running time is called an FPT algorithm. The class of all fixed-parameter tractable decision problems is called FPT.

The function \( f(x) \) may grow exponentially as \( x \) increases, but the running time may be acceptable if \( \kappa \) is small for problem instances that are of practical interest. We adopt the usual convention of omitting the polynomial factor in \( O(f(\kappa)|I|^c) \) and write \( O^*(f(\kappa)) \) instead.

### 1.2 Matrix decomposition and role mining

A \textit{Boolean matrix} is a matrix in which all entries are either 0 or 1. Let \( \lor \) and \( \land \) denote the usual logical operators on the set \([0, 1]\). We extend these operators to Boolean matrices in the natural way [8]:

1. the sum \( A \lor B \) of Boolean matrices \( A \) and \( B \) is computed as usual with addition replaced by \( \lor \);
2. the product \( A \land B \) of Boolean matrices \( A \) and \( B \) is computed as usual with multiplication replaced by \( \land \) and addition by \( \lor \). Thus, if \( C = A \land B \) then
   \[
   c_{ij} = \lor_{p=1}^n a_{ip} \land b_{pj},
   \]
   where \( n \) is the number of columns in \( A \) and the number of rows in \( B \).

Henceforth all matrices are Boolean, unless specified otherwise.

Any binary relation \( X \subseteq Y \times Z \) may be represented by a matrix \( X \) with rows indexed by \( Y \) and columns indexed by \( Z \), where \( X_{ij} = 1 \iff (i, j) \in X \). Using matrices we can reformulate the \textit{Role Mining Problem} (RMP) as follows. Given a matrix \( UPA \) and an integer \( r \), find a matrix \( UA \) with \( r \) columns and a matrix \( PA \) with \( r \) rows such that \( UPA = UA \land PA \). Thus, role mining may be regarded as a matrix decomposition problem.

### 1.3 Generalized Noise Role Mining

As previously noted, there is often no solution to RMP if \( r \) is small. In such cases, it is helpful to consider an extension of RMP called \textit{Noise Role Mining} [1, 3, 4, 10, 17], where the input includes a natural number \( k \) and our aim is to find a matrix \( UA \) with \( r \) columns and a matrix \( UA \) with \( r \) rows such that \( d_{KL}(UPA, UA \land PA) \leq k \), where \( d_{KL}(UPA, UA \land PA) \) is the number of entries in which \( UPA \) and \( UA \land PA \) differ (i.e., the \textit{Hamming distance} between them).

\textit{Noise Role Mining} could be seen as a rather crude approach to the problem of decomposing \( UPA \), as it doesn’t distinguish between zeroes in \( UPA \) being replaced with ones in \( UA \land PA \) and ones being replaced with zeroes. In the first case, a user is assigned to a permission that they didn’t previously have – a potential security problem. In the second case, a user no longer has a permission...
that they had been assigned, meaning the user may not be able to perform some of their responsibilities—an availability problem.

Thus, it will be appropriate in many cases to find $UA$ and $PA$ such that either security or availability, as specified by $UPA$ is preserved. Informally, a refinement of Noise Role Mining, then, would be to define Availability-preserving Role Mining, where we require $UPA \leq UA \land PA$, in the sense that every entry in $UPA$ is less than or equal to the corresponding entry in $UA \land PA$. In other words, every permission authorized by $UPA$ is also authorized by $UA \land PA$. Similarly, we could define Security-preserving Role Mining, where we require $UPA \geq UA \land PA$.

An even more fine-grained problem—the topic of this paper—is Generalized Noise Role Mining (GNRM), of which Noise Role Mining, Availability-preserving Role Mining and Security-preserving Role Mining are all special cases. In GNRM we specify at the user level whether the decomposition into $UA$ and $PA$ is security-preserving, availability-preserving, neither, or both.

We now introduce some notation to enable us to express GNRM formally. For a positive integer $t$, let $\{t\} = \{1, 2, \ldots, t\}$. Let $A$ and $B$ be $m \times n$ Boolean matrices and $F$ be an $m \times n$ label matrix with entries $f_{ij} \in \{\top, \bot\}$. The matrix $F$ is used to define a generalized distance metric between $A$ and $B$. For any $(i, j) \in [m] \times [n]$ the $F$-distance from entry $a_{ij}$ to entry $b_{ij}$ of matrices $A$ and $B$ is

$$sd_F(a_{ij}, b_{ij}) = \begin{cases} \infty & f_{ij} = \perp \text{ and } a_{ij} \neq b_{ij}, \\ |a_{ij} - b_{ij}| & \text{otherwise} \end{cases}$$

In other words, if we are not allowed to change $a_{ij}$ in order to obtain $b_{ij}$ (symbol $\perp$) and $a_{ij} \neq b_{ij}$ then the $F$-distance from $a_{ij}$ to $b_{ij}$ is $\infty$. Otherwise, it is just $|a_{ij} - b_{ij}|$.

We define $sd_F(A, B) = \sum_{i=1}^{m} \sum_{j=1}^{n} sd_F(a_{ij}, b_{ij})$. Thus, the distance from $A$ to $B$ is finite if and only if for every $(i, j) \in [m] \times [n]$ such that $a_{ij} \neq b_{ij}$ we have $f(a_{ij}) = \perp$. We now define Generalized Noise Role Mining as a parameterized problem.

**Generalized Noise Role Mining (GNRM)**

**Input:** An $m \times n$ user-permission assignment matrix $UPA$, a label matrix $F$, and nonnegative integers $k$ and $r$.  
**Parameter:** $k + r$  
**Goal:** Is there an $m \times r$ user-role assignment matrix $UA$, an $n \times r$ role-permission assignment matrix $PA$ such that $sd_F(U_PA, UA \land PA) \leq k$? If the answer is yes, then return such matrices $UA$ and $PA$.

Note that GNRM is parameterized by the sum $k + r$. This is because Noise Role Mining parameterized separately by either $k$ or $r$ is intractable: in particular, for $k = 0$ we have Exact Role Mining which is NP-hard [5]; and for $r = 1$, Noise Role Mining is NP-hard [1, 4].

Note also that GNRM reduces to:

- **Noise Role Mining** if $f_{ij} = \top$ for all $(i, j) \in [m] \times [n]$;
- **Availability-preserving Role Mining** if $f_{ij} = \top$ if and only if $UPA_{ij} = 0$; and
- **Security-preserving Role Mining** if $f_{ij} = \top$ if and only if $UPA_{ij} = 1$.

The rest of the paper is organized in the following way. In Sections 2 and 3 we describe our FPT algorithms for solving GNRM and related problems. We describe and discuss our experimental results in Section 4, and conclude the paper with a summary of our contributions and ideas for future work.

## 2 AN FPT ALGORITHM FOR GNRM

Fomin et al. [3] proved that Noise Role Mining parameterized by $k + r$ is FPT. We will extend this result to the GNRM by reducing it to the Generalized P-matrix Approximation problem. The reduction is similar to the one used in [3]. However, Fomin et al. [3] solved Noise Role Mining as a decision problem (i.e. a problem where the aim is to decide whether the given instance is a yes- or no-instance), whereas we solve Generalized Noise Role Mining as a search problem (i.e. if the given instance is a yes-instance then a solution is also returned).

We first define Generalized P-matrix Approximation and prove that it is FPT. We then explain how this problem is used to establish that GNRM is FPT.

### 2.1 Generalized P-matrix approximation

Let $P$ be a $p \times q$ matrix (sometimes called a pattern matrix). We say that an $m \times n$ matrix $B$ is a $P$-matrix if there is a partition $\{l_1, \ldots, l_p\}$ of $[m]$ and a partition $\{j_1, \ldots, j_q\}$ of $[n]$ such that for every $i \in [p]$, $j \in [q]$, $s \in l_i, t \in j_j$, we have $b_{st} = p_{ij}$. Note that, by definition, every set in the partitions of $[m]$ and $[n]$ is non-empty. (Thus, $p \leq m$ and $q \leq n$.) In other words, $B$ is a $P$-matrix if $P$ can be obtained from $B$ by first permuting rows and columns, then partitioning the resulting matrix into blocks such that in each block $L$, all entries are of the same value $v(L)$ and finally replacing every block $L$ by one entry of value $v(L)$.

For example, let $P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$. Then $Q_1$ and $Q_2$ below are both $P$-matrices: permuting columns 2 and 3 in each matrix, then partitioning (into blocks of equal size for $Q_1$, and between rows 1 and 2 and columns 2 and 3 for $Q_2$) and “contracting” gives us $P$.

$$Q_1 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

$$Q_2 = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

**Generalized P-matrix Approximation**

**Input:** An $m \times n$ matrix $A$, a label matrix $F$, a $p \times q$ matrix $P$, and a nonnegative integer $k$.  
**Goal:** Is there an $m \times n$ P-matrix $B$ such that $sd_F(A, B) \leq k$? If the answer is yes, then return such a matrix $B$.

Very informally, this problem asks whether there exists a matrix $B$ that is (almost) the same as $A$ and contains the rows and columns of $P$, and, if so, returns $B$. Fomin et al. [3] used the special case of Generalized P-matrix Approximation, where $f_{ij} = \top$ for every $(i, j) \in [m] \times [n]$. It is called the P-matrix Approximation problem. We will use the following two results by Fomin et al. [3].
Observation 2.1. Let $P$ be a $p \times q$ matrix. Then, every $P$-matrix $B$ has at most $p$ pairwise distinct rows and at most $q$ pairwise distinct columns.

Proposition 2.2. Given an $m \times n$ matrix $A$ and a $p \times q$ matrix $P$, there is an algorithm that runs in time $2^{p \log p + q \log q}(nm)^{O(1)}$ and correctly outputs whether $A$ is a $P$-matrix.

If $A$ has at most $p - 1$ rows or at most $q - 1$ columns, then there is no $m \times n$ matrix $B$ that is a $P$-matrix and $sdf(A, B) \leq k$. In that case, the instance is a no-instance. Let us now assume that $A$ has at least $p$ rows and at least $q$ columns.

The next lemma was proved in [3] for P-MATRIX APPROXIMATION. Note that replacing $\top$ in $f_{ij} = \top$ by $\bot$ for some entries $f_{ij}$ will only reduce the set of yes-instances of Generalized P-Matrix Approximation. Thus, the next lemma follows from its special case in [3].

Lemma 2.3. If $A$ has at least $p + k + 1$ pairwise distinct rows or at least $q + k + 1$ pairwise distinct columns, then output that $(A, F, P, k)$ is a no-instance of Generalized P-Matrix Approximation.

This lemma implies the following reduction/preprocessing rule.

Reduction Rule 1. Let $A$ be a matrix. If $A$ has at least $p + k + 1$ pairwise distinct rows or at least $q + k + 1$ pairwise distinct columns, then output that $(A, F, P, k)$ is a no-instance of Generalized P-Matrix Approximation.

To simplify an instance $(A, F, P, k)$ of Generalized P-Matrix Approximation, we can apply the following reduction rule exhaustively. If a row $a_i$ is deleted by the reduction rule, the label matrix $F$ does not change for the other rows. This means that for the reduced instance with matrix $A'$, the label matrix $F$ restricted to the rows of $A'$, i.e. $\{a_1, \ldots, a_m\} \setminus \{a_i\}$. For simplicity of presentation, the label matrix for $A'$ will still be denoted by $F$.

We define a second reduction rule that is used to delete superfluous identical rows and columns.

Reduction Rule 2. If $A$ has at least $\max(p, k) + 2$ identical rows, then delete one of these identical rows. Similarly, if $A$ has at least $\max(q, k) + 2$ identical columns, then delete one of these identical columns.

We say two instances of Generalized P-Matrix Approximation are equivalent if they are both either yes-instances or no-instances. Fomin et al. proved that any application of Reduction Rule 2 to an instance of P-Matrix Approximation returns an equivalent instance of the problem [3, Claim 7]. It is easy to verify that the arguments in their proof of Claim 7 also apply to Generalized P-Matrix Approximation.

Applications of the two reductions rules described above either determine that the input instance is a no-instance or produce an equivalent instance with the following properties.

Lemma 2.4. Let $(A, F, P, k)$ be an instance of Generalized P-Matrix Approximation. Then, there exists a polynomial-time algorithm that either returns "no-instance" or transforms $(A, F, P, k)$ into an equivalent instance $(A', F, P, k)$ of Generalized P-Matrix Approximation. Moreover, the following properties are satisfied.

1. The matrix $A'$ has at least $p$ rows, at least $q$ columns, at most $(\max(p, k) + 1)(p + k)$ rows and at most $(\max(p, k) + 1)(p + k)$ columns.

2. Given a $P$-matrix $B'$ such that $sdf(A', B') \leq k$, in polynomial time we can compute a $P$-matrix $B$ such that $sdf(A, B) \leq k$.

Proof. Let $(A, F, P, k)$ be an input instance of Generalized P-Matrix Approximation. As $m \geq p$ and $n \geq q$, if $A$ has at most $p - 1$ rows or has at most $q - 1$ columns, then there is no $m \times n$ P-matrix $B$ such that $sdf(A, B) \leq k$. In such a case, we return "no-instance." Next, we apply Reduction Rule 1 to check the number of pairwise distinct rows as well as the number of pairwise distinct columns in $A$. If $A$ has $p + k + 1$ pairwise distinct rows or $q + k + 1$ pairwise distinct columns, then we return "no-instance." After that, we apply Reduction Rule 2 exhaustively and let $A'$ be the obtained matrix. We also obtain a stack $S$ which contains all deleted rows and columns.

We return $(A', F, P, k)$ as the output instance. Clearly, $A'$ has at most $p + k$ pairwise distinct rows and at most $q + k$ pairwise distinct columns. Moreover, $A'$ has at least $p$ rows and at least $q$ columns. Also, $A'$ can have at most $\max(p, k) + 1$ pairwise identical rows and at most $\max(q, k) + 1$ pairwise identical columns. This means that $A'$ has at most $(\max(p, k) + 1)(p + k)$ rows and at most $(\max(q, k) + 1)(q + k)$ columns. This completes the proof that property (1) holds.

Suppose that $B'$ is a $P$-matrix such that $sdf(A', B') \leq k$. Note that at any intermediate stage, when a row $r$ (a column $c$, respectively) was deleted from $A$, there were at least $(\max(p, k) + 1)$ additional rows identical to $r$ (at least $(\max(q, k) + 1)$ additional columns identical to $c$, respectively). Hence, in $A'$, if a row $r$ or a column $c$ was deleted by Reduction Rule 2, then there are exactly $\max(p, k) + 1$ rows identical to $r$ (exactly $\max(q, k) + 1$ columns identical to $c$, respectively). Since $sdf(A', B') \leq k$, at most $k$ entries with label $\top$ identical to $r$ (c, respectively) were modified in $B'$. Thus, $B'$ must have at least one row identical to $r$ (at least one column identical to $c$, respectively) if $r$ (c, respectively) was deleted by Reduction Rule 2. Therefore, the deleted rows (columns, respectively) are identical to some rows (columns, respectively) which are the same in $A'$ and $B'$. Thus, reinstalling the deleted rows and columns using stack $S$, we obtain matrices $A$ and $B$ such that $B$ is a $P$-matrix and $sdf(A, B) \leq k$.

Theorem 2.5. Generalized P-Matrix Approximation can be solved in time $2^{p \log p + q \log q}(nm)^{O(1)}((\max(p, k) + 1)(p + k)(\max(q, k) + 1)(q + k))^k$.

Proof. Let $(A, F, P, k)$ be an instance of Generalized P-Matrix Approximation. First, we invoke the polynomial-time algorithm of Lemma 2.4 to either determine that the input instance is a no-instance or generate an instance $(A', F, P, k)$ satisfying properties (1) and (2). Recall that the first property says that $A'$ has at most $(\max(p, k) + 1)(p + k)$ rows, and at most $(\max(q, k) + 1)(q + k)$ columns. This means that $A'$ has at most $(\max(p, k) + 1)(p + k)(\max(q, k) + 1)(q + k)$ entries. We then consider all possible sets of at most $k$ entries. For every entry of such a set, if the label of an entry is $\top$, we will modify it. This results in a modified matrix $B'$. We then invoke Proposition 2.2 to check whether $B'$ is a $P$-matrix or not. This checking takes $2^{p \log p + q \log q}(nm)^{O(1)}$-time. If $B'$ is a $P$-matrix,
it is a solution to Generalized P-Matrix Approximation for the instance \((A', F, B', k)\) as \(sdp(A', B') \leq k\) (since we changed at most \(k\) entries in \(A')\). Then, we make use of property (2) to construct \(B\) satisfying \(sdp(A, B) \leq k\) and return \(B\) as a solution of Generalized P-Matrix Approximation for the instance \((A, F, B, k)\). Recall that this step takes polynomial time. Hence, the overall algorithm takes \(2p\log p + 2q\log q(nm)O(1)\) \((\max(p, k) + 1)(p + k)(\max(q, k) + 1)(q + k)\)^3 time. \(\square\)

### 2.2 GNRM is FPT

We now explain how the algorithm for Generalized P-Matrix Approximation is used to solve GNRM and thus show it is FPT. The basic strategy is to consider all possible pairs of matrices whose product \(P\) would provide the basis for a solution to GNRM. The number of such pairs is bounded above by a function of \(r\). For each such \(P\), we determine whether the Generalized P-Matrix Approximation instance \((UPA, F, P, k)\) has a solution, in which case we can then compute a solution to the GNRM instance.

**Lemma 2.6.** Let \(P\) be an \(p \times q\) matrix such that \(P = X \land Y\) for a \(p \times r\) matrix \(X\) and an \(r \times q\) matrix \(Y\). Furthermore, consider an \(m \times n\) matrix \(B\) which is a \(P\)-matrix. Then, we can in polynomial time obtain an \(m \times r\) matrix \(X^r\), and \(r \times n\) matrix \(Y^r\) such that \(B = X^r \land Y^r\).

**Proof.** As \(B\) is a \(P\)-matrix, there are partitions \([l_1, \ldots, l_p]\) of \([m]\) and \([j_1, \ldots, j_q]\) of \([n]\) such that for every \(i \in [p], j \in [q]\), \(s \in [l_i], t \in [j_j]\), we have \(B_{st} = p_{ij}\).

We initialize \(X^r = X\) and \(Y^r = Y\). Consider an entry \(p_{ij}\). Let \(x_i\) be the \(i\)th row of \(X\) and \(y_j\) the \(j\)th column of \(Y\); then \(x_i \land y_j = p_{ij}\). Let \(c \in [p]\) and \(d \in [q]\) such that \(i \in l_c\) and \(j \in j_d\). Then, for any \(s \in [l_c]\) and \(t \in [j_d]\), set \(B_{st} = p_{ij}\). Then, for any \(s \in [l_c]\) and for any \(t \in [j_d]\), we insert \(x_i\) as the \(s\)th row of \(X^r\) and \(y_j\) as the \(t\)th column of \(Y^r\). \(\square\)

The Boolean rank of a matrix \(A\), denoted \(BRank(A)\), is the minimum natural number \(r\) such that \(A = X \land Y\) for some column-vectors \(X\) and \(Y\). In fact, \(BRank(A) = r\) if and only if \(A = x \land y^T\) for some column-vectors \(x\) and \(y\). Thus, a matrix \(A\) has Boolean rank 1 if and only if \(A = x \land y^T\) for some column-vectors \(x\) and \(y\).

**Theorem 2.7.** Generalized Noise Role Mining admits an \(O^*(2^{O(r^2 + rk)})\)-time algorithm.

**Proof.** Let \(B\) be an \(m \times n\) matrix and let \(r\) be the Boolean rank of \(B\). Thus, there are \(r\) matrices \(B^{(1)}, \ldots, B^{(r)}\), each of Boolean rank 1, such that \(B = B^{(1)} \land \cdots \land B^{(r)}\), where for each \(i \in [r]\), \(B^{(i)} = x_i \land y^T\) for some column-vectors \(x_i\) and \(y^T\). It can be shown by induction on \(r\) that \(B\) has at most \(2^r\) distinct rows and at most \(2^r\) distinct columns.\(^1\) Therefore, \(B\) is of Boolean rank at most \(r\) if and only if there is a \(p \times q\) matrix \(P\) of Boolean rank at most \(r\) for \(p = \min(2^r, m)\) and \(q = \min(2^r, n)\) such as \(B\) is a \(P\)-matrix.

Moreover, an \(m \times n\) matrix \(B\) is of rank \(r\) if \(r\) is the minimum natural number such that \(B = C \land D\), where \(C\) is an \(m \times r\)-matrix and \(D\) is an \(r \times n\)-matrix. Hence, Generalized Noise Role Mining can be reformulated as follows: Decide whether there is a \(p \times q\) pattern matrix \(P\) of Boolean rank \(r\) and an \(m \times n\) matrix \(B\) such that \(sdp(UPA, B) \leq k\) and if \(B\) does exist then find matrices \(U\) and \(A\) of sizes \(m \times r\) and \(r \times n\), respectively, such that \(B = U \land A\).

Thus, to solve Generalized Noise Role Mining with input \((UPA, F, P, k)\), we can use the following algorithm:

1. Generate all pairs \((X, Y)\) of matrices of sizes \(p \times r\) and \(r \times q\), respectively, and for each such pair compute \(P = X \land Y\).
2. For each \(P\), solve Generalized P-Matrix Approximation for the instance \((UPA, F, P, k)\). If \((UPA, F, P, k)\) is a yes-instance, then using the algorithm of Lemma 2.6 return matrices \(U\) and \(A\) of sizes \(m \times r\) and \(r \times n\) such that \(B = U \land A\), where \(B\) is the solution of the instance \((UPA, F, P, k)\).
3. If all instances above are no-instances of Generalized P-Matrix Approximation, return "no-instance."

It remains to evaluate the running time of the above algorithm. Since \(p \leq 2^r\) and \(q \leq 2^r\), there are at most \(2^{O(r^2)}\) pairs \((X, Y)\), and we can compute all matrices \(P\) in time \(2^{O(r^2)}\). Thus, the running time of the algorithm is dominated by that of Step 2. The running time of Step 2 is upper bounded by the number of matrices \(P\) (it is equal to \(2^{O(r^2)}\)) times the maximum running time of solving Generalized P-Matrix Approximation on an instance \((UPA, F, P, k)\) and computing \(U\) and \(A\), if the instance is a yes-instance. By Lemma 2.6, Theorem 2.5 and the bounds \(p \leq 2^r\), \(q \leq 2^r\), the maximum running time is upper bounded by \(O^*(2^{O(r^2 + rk)})\). It remains to observe that \(2^{O(r^2)} \cdot O^*(2^{O(r^2 + rk)}) = O^*(2^{O(r^2 + rk)})\). \(\square\)

### 3 USER AND PERMISSION COVERING NUMBERS

Unfortunately, instances of GNRM may not have a solution for acceptable values of \(k\) and \(r\) (see our experimental results, for example). We may, therefore, wish to limit the number of users and permissions that are affected by a solution to a role mining instance (but not the number of authorizations that are affected). This may enable us to find an acceptable solution to the role mining problem, except for a small number of users and/or permissions. It may then be possible to tweak this solution for the users and permissions whose authorizations do not satisfy the original user-permission relation. We provide a small illustrative example later in this section.

Accordingly we will introduce a new problem, related to GNRM, where instead of \(U\) and \(A\) (with \(r\) rows and columns, respectively) such that the Hamming distance between \(U\) and \(A\) is at most \(k\), our goal is to find \(U\) and \(A\) such that after making changes restricted to entries in at most \(cov_U\) rows and \(cov_P\) columns of \(U\), the modified \(U\) is equal to \(U \land A\). The parameters \(cov_U\) and \(cov_P\) are called the user covering number and permission covering number, respectively.
Let $i \in [m] \setminus I$. Note that $a_{ij} = b_{ij}$ whenever $j \in [n] \setminus J$. Therefore the $i$-th row of $A$, denoted $a_i$, can differ from the $i$-th row of $B$, denoted $b_i$, only in the coordinates in $J$. Therefore, there exists $J' \subseteq J$ such that $a_{ij} = 1 - b_{ij}$ for all $j \in J'$ and $a_{ij} = b_{ij}$ for all $j \in [n] \setminus J'$. It follows that $a_i$ is one of the $2^{|J|}$ many vectors we obtain from $b_i$ by flipping the value of the entries in $J' \subseteq J$. Since there are only $p$ many distinct rows in $B$, it follows that for all $i \in [m] \setminus I$, the row $a_i$ is one of the $2^{|J|}$ many vectors. Recall that $|J| \leq \text{cov}_U$ and $|J| \leq \text{cov}_P$. It follows that $A$ has at most $\text{cov}_U + p \cdot 2^{\text{cov}_P}$ pairwise distinct rows. Using an analogous argument, we obtain that $A$ has at most $\text{cov}_P + q \cdot 2^{\text{cov}_U}$ pairwise distinct columns. Since we started from an arbitrary solution, it follows that if $A$ has at least $\text{cov}_U + p \cdot 2^{\text{cov}_P} + 1$ pairwise distinct rows or at least $\text{cov}_P + q \cdot 2^{\text{cov}_U} + 1$ pairwise distinct columns, then $(A, F, P, \text{cov}_U, \text{cov}_P)$ is a no-instance. \hfill $\square$

The above lemma implies the following reduction/preprocessing rule.

**Reduction Rule 3.** Let $A$ be a matrix. If $A$ has at least $\text{cov}_U + p \cdot 2^{\text{cov}_P} + 1$ pairwise distinct rows or at least $\text{cov}_P + q \cdot 2^{\text{cov}_U} + 1$ pairwise distinct columns, then output that $(A, F, P, \text{cov}_U, \text{cov}_P)$ is a no-instance of Generalized Cover P-Matrix Approximation.

Unfortunately, even when there are at least $\max(p, \text{cov}_U) + 2$ identical rows (or $\max(q, \text{cov}_P) + 2$ identical columns) that are exactly the same in $A$, we may not be able to apply an equivalent of Reduction Rule 2 to $A$. The reason is that we may still need to change some entries of each of the rows (columns) to obtain the $P$-matrix $B$. Namely, we are allowed only to change the entries in at most $\text{cov}_U$ many rows in some set $I$ and at most $\text{cov}_P$ many columns in some set $J$, where $I$ and $J$ depend on the matrix $B$. Assume that we remove a row $r$ such that there are still at least $\max(p, \text{cov}_U) + 1$ rows identical to $r$ in the matrix $A'$ obtained from $A$ by removing $r$. Now assume that we have a $P$-matrix $B'$ such that there exist $I \subseteq [m]$ and $J \subseteq [n]$, such that $|I| \leq \text{cov}_U$, $|J| \leq \text{cov}_P$, and if $a_{ij} \neq b_{ij}$, then $f_{ij} = \top$ and either $i \in I$ or $j \in J$. Clearly there is $i \in [n] \setminus I$ such that $a_{ij}' \neq b_{ij}'$ for the $i$-th row of $A'$, is identical to $r$. But, $b_{ij}'$ (the $i$-th row of $B'$) can still differ from $a_{ij}'$ at the entries in $J$. Moreover $f_{ij} = \top$ whenever $a_{ij}' \neq b_{ij}'$ for some $j \in J$. This might not be true if we simply used $b_{ij}'$ as the replacement for $r$.

However, this problem disappears if the label matrix $F$ is less restrictive for the row $r$ than any of the remaining $\max(p, \text{cov}_U) + 1$ identical rows. Indeed, if we are allowed to modify the permission of a user in the $\max(p, \text{cov}_U) + 1$ identical rows to access some resource (according to $F$), then we are also allowed to modify the permission for the same resource for the user represented by the row $r$. This leads to the following two reduction rules.

**Reduction Rule 4.** If there exists $i \in [m]$ and $R \subseteq [m]$ such that $(1) i \notin R$, $(2) |R| = \max(p, \text{cov}_U) + 1$, and $(3)$ for all $i' \in R$ we have that the row $a_i'$ is identical to $a_i$ (and for all $i' \in R$ and all $j \in [n]$ we have that $f_{ij} = \top$ implies that $f_{ij} = \top$. Then delete the $i$-th row from the matrix $A$.

Observe that following the discussion above it is rather easy to see that we can obtain a solution for the instance $(A, F, P, \text{cov}_U, \text{cov}_P)$ from a solution $B'$ for $(A', F', P, \text{cov}_U, \text{cov}_P)$ obtained by applying Reduction Rule 4 once on the $i$-th row of $A$. Let $I' \subseteq [m - 1]$. Let $i \in [m] \setminus I$. Note that $a_{ij} = b_{ij}$ whenever $j \in [n] \setminus J$. Therefore
and \( J' \subseteq [n] \) be the sets witnessing that \( B' \) is a solution. As \( \max(p, \mathsf{cov}_U) + 1 > |J'| \), one of \( \max(p, \mathsf{cov}_U) + 1 \) many rows in \( R \) is not in \( J' \). Hence its replacement in \( B' \) differs from the original only in the entries in \( J' \). Since the \( i \)-th row in \( A \) is identical to this row and the label matrix \( F \) is less restrictive for the \( i \)-th row, we can use the same replacement for the \( i \)-th row as well.

Applying the same argument for columns, we get an analogous reduction rule for removing columns together with a way to reconstruct a solution from the reduced instance.

**Reduction Rule 5.** If there exists \( j \in [n] \) and \( C \subseteq [n] \) such that (1) \( j \notin C \), (2) \(|C| = \max(q, \mathsf{cov}_P) + 1 \), (3) for all \( j' \in C \) we have that the \( j' \)-th column of \( A \) is identical to the \( j \)-th column, and (4) for all \( j' \in C \) and all \( i \in [m] \) we have that \( f_{ij'} = \top \) implies that \( f_{ij} = \top \). Then delete the \( j \)-th column from the matrix \( A \).

Note that in particular, if there are at least \( \max(p, \mathsf{cov}_U) + 2 \) identical rows (or \( \max(q, \mathsf{cov}_P) + 2 \) identical columns) that are exactly the same in \( A \) and their corresponding rows (columns) in \( F \) are also identical, then Reduction Rule 4 (Reduction Rule 5) applies. In particular, in the cases when the label matrix \( F \) is availability-preserving, security-preserving, or contains only \( \top \) on all entries, we get that if the two rows/columns are identical in \( A \), then they are identical in \( F \). Using arguments similar to Lemma 2.4, one can prove that in all these cases, we can always reduce the original instance of the problem to an instance with at most \( \mathsf{cov}_U + p \cdot 2^{\mathsf{cov}_P} \cdot (\max(p, \mathsf{cov}_U) + 1) \) many rows and at most \( \mathsf{cov}_P + q \cdot 2^{\mathsf{cov}_U} \cdot (\max(q, \mathsf{cov}_P) + 1) \) many columns and obtain the following theorem by enumerating all possible solutions for such reduced instance.

In the rest of this section, we restrict our attention to the following three important subproblems of \( \mathsf{GNRM-cov} \): \( \mathsf{Noise Cover Role Mining}, \mathsf{Availability-preserving Cover Role Mining} \) and \( \mathsf{Security-preserving Cover Role Mining} \). Recall that for the first subproblem above \( f_{ij} = \top \) for all \( (i, j) \in [m] \times [n] \), for the second one (the third one, respectively), \( f_{ij} = \top \) if and only if \( a_{ij} = 1 \) (\( a_{ij} = 1 \), respectively).

**Theorem 3.3.** \( \mathsf{Noise/Availability-preserving/Security-preserving Cover P-Matrix Approximation} \) parameterized by \( p + q + \mathsf{cov}_U + \mathsf{cov}_P \) can be solved in \( \mathsf{FPT} \) time.

Finally, similarly to the proof of Theorem 2.7, we can prove that Theorem 3.3 in combination with Lemma 2.6 imply the following:

**Theorem 3.4.** \( \mathsf{Noise/Availability-preserving/Security-preserving Cover Role Mining} \) parameterized by \( p + q + \mathsf{cov}_U + \mathsf{cov}_P \) admits an \( \mathsf{FPT} \) algorithm.

Note that the \( \mathsf{FPT} \) algorithm of Theorem 3.4 implies the existence of \( \mathsf{FPT} \) algorithms for the three important subproblems of the following variation of \( \mathsf{GNRM-cov} \). The problem is similar to \( \mathsf{GNRM-cov} \), but where we have \( \mathsf{cov} \) as a parameter instead of \( \mathsf{cov}_U \) and \( \mathsf{cov}_P \). We can run the algorithm of Theorem 3.4 for every pair \( \mathsf{cov}_U \) and \( \mathsf{cov}_P \) such that \( \mathsf{cov} = \mathsf{cov}_U + \mathsf{cov}_P \) and stop when the algorithm returns yes or returns no for all pairs \( \mathsf{cov}_U \) and \( \mathsf{cov}_P \) such that \( \mathsf{cov} = \mathsf{cov}_U + \mathsf{cov}_P \).

### 4 EXPERIMENTAL EVALUATION

In this section, we describe experiments in which we solve \( \mathsf{GNRM} \) as a constraint satisfaction problem (CSP). Our experiments were designed to provide insights into the following four questions:

- How does the choice of \( F \) affect role mining, in terms of \( r, k \) and running times?
- What values of \( k \) and \( r \) are required to obtained solutions for a given \( \mathsf{UPA} \) and \( F \)?
- How does the time required to solve \( \mathsf{GNRM} \) vary with \( k \) and \( r \)?
- Does a general-purpose solver using the CSP formulation of \( \mathsf{GNRM} \) exhibit \( \mathsf{FPT} \)-like running times?

We first present our formulation of \( \mathsf{GNRM} \) as a CSP. We then describe the set-up for our experiments. In the remainder of the section, we report the results of our experiments and discuss their implications.

#### 4.1 CSP formulation of \( \mathsf{GNRM} \)

We present a CSP formulation of the \( \mathsf{GNRM} \) in order to solve the problem using a general-purpose solver. We could have implemented a bespoke \( \mathsf{FPT} \) algorithm to solve \( \mathsf{GNRM} \), based on our results in the preceding sections. However, we believe using a general-purpose solver is likely to be more useful in practice. First, the formulation of the \( \mathsf{GNRM} \) as a CSP is quite intuitive, and therefore easier to understand and maintain than a bespoke algorithm that relies on some relatively complex theory. Second, general-purpose solvers may perform well on instances of a hard problem that is known to be \( \mathsf{FPT} \).

Our formulation consists of two matrices of Boolean variables \( u_{aij}, i \in [m], \ell \in [r], p_{aij}, \ell \in [r], j \in [n] \). We also use a matrix of auxiliary Boolean variables \( d_{ij}, i \in [m], j \in [n] \), representing the discrepancies between \( \mathsf{UPA} \) and \( \mathsf{UPA} \). We use the values 0 and 1 to represent the values \( \bot \) and \( \top \), respectively, in the \( F \) matrix. Figure 1 describes the formulation in detail.

A naïve formulation of the problem would include a matrix of Boolean variables – to represent the product \( \mathsf{UPA} \) – and link these variables to the \( u_{aij} \) and \( p_{aij} \) variables. Then it would be easy to formulate the constraints and link the decision variables to variables \( d_{ij} \). However, knowing the values of \( u_{aij} \) and \( f_{ij} \) for a specific pair \( (i, j) \), we can formulate the constraints more compactly. Thus, the formulation described in Figure 1 defines the constraints separately for each combination of values of \( u_{aij} \) and \( f_{ij} \).

#### 4.2 Experimental Setup

We used synthetic data for our experiments because we wanted to explore particular questions in a structured manner, which required using instances that satisfied certain criteria, as explained.

---

\(^2\)The compact formulation performed significantly better than the naïve formulation in our experiments, so we only report results for the compact configuration. Let \( t_N \) and \( t_C \) denote, respectively, the running times of the solver when using the naïve and compact configurations. For instances that obviously have no solution (small \( r \) and small \( k \)), \( t_N \) and \( t_C \) are similar. As \( k \) increases, it becomes more difficult for the solver to determine whether the instance has a solution or not. Moreover, \( t_N \) increases much more rapidly than \( t_C \). For example, when \( r = 3 \) and \( k = 3 \), our instance has no solution and we have \( t_N \approx 2t_C \), whereas for \( r = 3 \) and \( k = 12 \), the instance still has no solution but \( t_N \approx 12t_C \).
For all \( i \in [m] \) and \( j \in [n] \) such that \( f_{i,j} = 0 \) and \( u_{a_{i,j}} = 0 \):
\[
ua_{i,\ell} + \pa_{a_{i,j}} \leq 1 \quad \forall \ell \in [r],
\]
(1)

For all \( i \in [m] \) and \( j \in [n] \) such that \( f_{i,j} = 0 \) and \( u_{a_{i,j}} = 1 \):
\[
x_{i,j,\ell} \leq ua_{i,\ell} \quad \forall \ell \in [r],
\]
(2)
\[
x_{i,j,\ell} \leq \pa_{a_{i,j}} \quad \forall \ell \in [r],
\]
(3)
\[
\sum_{\ell \in [r]} x_{i,j,\ell} \geq 1 \quad \forall \ell \in [r],
\]
(4)

For all \( i \in [m] \) and \( j \in [n] \) such that \( f_{i,j} = 1 \) and \( u_{a_{i,j}} = 0 \):
\[
ua_{i,\ell} + \pa_{a_{i,j}} \leq 1 + d_{i,j} \quad \forall \ell \in [r],
\]
(5)

For all \( i \in [m] \) and \( j \in [n] \) such that \( f_{i,j} = 1 \) and \( u_{a_{i,j}} = 1 \):
\[
ua_{i,\ell} + \pa_{a_{i,j}} \geq x_{i,j,\ell} \quad \forall \ell \in [r],
\]
(6)
\[
\pa_{a_{i,j}} \geq x_{i,j,\ell} \quad \forall \ell \in [r],
\]
(7)
\[
\sum_{\ell \in [r]} x_{i,j,\ell} \geq 1 - d_{i,j} \quad \forall \ell \in [r],
\]
(8)

\[
\sum_{i \in [m]} \sum_{j \in [n]} d_{i,j} \leq k,
\]
(9)

\[
x_{i,j,\ell} \in \{0, 1\} \quad \forall i \in [m], \forall j \in [n], \forall \ell \in [r],
\]
(10)

\[
d_{i,j} \in \{0, 1\} \quad \forall i \in [m], \forall j \in [n],
\]
(11)

\[
ua_{i,j} \in \{0, 1\} \quad \forall i \in [m], \forall j \in [n],
\]
(12)

\[
\pa_{a_{i,j}} \in \{0, 1\} \quad \forall i \in [m], \forall j \in [n].
\]
(13)

Figure 1: CSP formulation of GNRM.

Below. We used a pseudo-random instance generator used in earlier experimental work by Vaidya et al. [18]. The generator produces instances of UPA by creating random UA and PA matrices and multiplying them together. This means that we know an upper bound on the number of roles we need to mine. We used the following parameters in our generator:

- the number of roles per user was randomly chosen for each user from the interval \([0, 3]\);
- the number of permissions per role was randomly chosen for each role from \([0, \lceil 0.25n \rceil]\); and
- the total number of roles for generating the UA and PA matrices was set to 10.

The selection of the parameters was based on typical values used in the literature and to ensure that the authorization density (the proportion of non-zero entries in the UPA matrix) was around 20%. (All but one of the real-world datasets commonly used in role mining research have authorization densities less than 20% [13, Table 1].) We study three different problems for each of the instances we generate:

- **Security:** \( f_{i,j} = u_{a_{i,j}} \);
- **Availability:** \( f_{i,j} = 1 - u_{a_{i,j}} \);
- **Noise:** \( f_{i,j} = 1 \) for every \( i \) and \( j \).

To solve GNRM, we used the CP-SAT solver from Google's OR-Tools, version 9.2.9972. The experiments were conducted on a machine based on two Xeon E5-2630 v2 CPUs (2.60 GHz), with 32 GB of RAM. Hyper-threading was enabled, and the solver was allowed to use all the cores. The solver was controlled from a Python script.

### 4.3 Relationship between \( k \) and \( r \)

In our first experiment, we consider the relationship between \( k \) and \( r \). Specifically, we study how \( k_{\text{min}} \) — the minimum value of \( k \) needed for the instance to have a solution — depends on the number of roles \( r \) that we attempt to extract. For this experiment, we used an instance of size \( m = 20 \) and \( n = 20 \), with 77 non-zero entries in the UPA matrix.

Figure 2 shows how \( k_{\text{min}} \) varies with \( r \). As one would expect, \( k_{\text{min}} \) decreases as \( r \) grows, for all three instance types. Moreover, \( k_{\text{min}} \) for the Noise instance is no greater than \( k_{\text{min}} \) for the equivalent Availability and Security instances. Finally, note that the Availability instance requires the longest time to solve.

Observe that \( k_{\text{min}} \) is bounded by \( \sum_{i} \sum_{j} u_{a_{i,j}} \) in the Noise and Security instances; however, it is bounded by \( mn - \sum_{i} \sum_{j} u_{a_{i,j}} \) in the Availability instances. Since the UPA matrix is usually sparse, \( k_{\text{min}} \) in the Availability instances is typically higher than in the other instances. These are not profound insights, but they do suggest that it is usually desirable to perform security-aware role mining, rather than availability-aware role mining, at least in terms of remaining close to the original UPA matrix and the time taken to solve the instance.
4.4 Relationship between run-time, \( r \) and \( k \)

We now consider how the running times of the solver depend on the values of \( k \) and \( r \). The analysis is affected significantly by the so-called phase transition phenomenon. Informally, the hardest instances of a hard problem are at the boundary of satisfiable (there exists a solution) and unsatisfiable (there is no solution), and small changes in the inputs to such an instance may change whether it is satisfiable or not.

In the context of GNRM, an instance with small values of \( k \) and \( r \) is over-subscribed: i.e., the constraints are so tight that the instance is trivially unsatisfiable. Conversely, an instance with large values of \( k \) and \( r \) is under-subscribed: i.e., the constraints are so loose that the instance is trivially satisfiable. The hard instances are in between, where small changes to UPA, F, \( r \) or \( k \) may change an unsatisfiable instance into a satisfiable one (and vice versa).

Figures 3 and 4 show how the running time of the solver changes with \( k \) if the value of \( r \) is fixed at 4 and 6, respectively. Solid lines connect the satisfiable instances and dotted lines the unsatisfiable instances. As one would expect, the satisfiable instances occur for smaller values of \( k \) in Figure 4. Otherwise, the figures are very similar and clearly illustrate several points:

- Security instances are by far the easiest, with running times showing little variation and being less than one second for all \( k \);
- Noise and Availability instances are comparable in their complexity;
- more discrepancies are required to find a solution for Availability instances;
- there is a clear phase transition pattern, where instances change from easy to hard to easy again;
- unsatisfiable instances are generally harder, and the running times are more predictable; and
- the running times in the satisfiable region are much more variable, as the running time depends on the ‘luck’ of the solver in finding a feasible solution.

Note that the phase transition for Noise and Security instances occurs at the same value of \( k \). Moreover, Security instances are much quicker to solve. In other words, given an instance of GNRM, it seems reasonable to solve the Security version of the problem. Not only does this guarantee that no additional authorizations will be granted by the role mining solution, we have evidence to suggest that no better solution (in terms of the number of discrepancies \( k \) between the solution and UPA) will be obtained by solving the more complex Noise version.

We conclude this section by noting that we obtained plots similar to Figures 3 and 4 for other values of \( r \); space constraints prevent the inclusion of those plots. However, we found that the value of \( r \) significantly affects the difficulty of the instance. In particular, unsatisfiable Noise and Availability instances in the phase transition region take a significant amount of time to solve, as illustrated in Figure 5.

For a given UPA and a fixed value of \( r \), we have a collection of instances using different values of \( k \) as input. For all values of \( k \) less than \( k_{\text{min}} \), the instances in the collection are unsatisfiable (and the remaining instances are satisfiable), as can be seen in Figures 3 and 4. Figure 5 reports the running time for the instance \( k = k_{\text{min}} - 1 \), corresponding to the hardest unsatisfiable instance for a given \( r \). Note the log scale on the y-axis, which indicates that hard Noise and Availability instances take a considerable amount of time to solve, whereas the time taken to solve Security instances remains under a second in all cases.

4.5 Are run-times FPT-like?

In our last experiment, we test whether the solver based on our CSP formulation behaves in practice as an FPT algorithm. A positive outcome would be of practical importance, as it would mean off-the-shelf solvers could solve GNRM, a hard problem in principle, reasonably efficiently in practice.

Traditionally, such experiments involve fixing the parameter value and varying the instance size; if the running time scales polynomially with the instance size then the solver is considered to demonstrate FPT-like running time [7]. Furthermore, to ensure that the experiment is fair and the structures of test instances are comparable, it is common to use only phase transition instances.

With GNRM, the natural way of obtaining a phase transition instance of a given size would be to adjust \( r \) or \( k \). This approach was not available to us as \( r + k \) is the parameter used to obtain our
FPT result. Alternatively, one could consider changing the density of the UPA matrix. However, this would affect the properties of the instance, making the instances incomparable. Our workaround is to fix \( r + k \), the FPT parameter. While this might not be the ideal approach, it still provides some useful insights, and fineses some of the difficulties associated with finding comparable phase transition instances of GNRM.

In our experiments, we fixed \( r + k = 10 \) as we know that the instance with \( r = 0 \) and \( k = 10 \) is unsatisfiable and the instance with \( r = 10 \) and \( k = 0 \) is satisfiable (in fact, even \( r = 8 \) and \( k = 0 \) is guaranteed to produce a satisfiable instance as the number of users in this experiment is only 8). We considered values for \( n \) from 16 to 16,384. Then, among all the allowed combinations of \( k \) and \( r \) we pick the highest value of \( r \) that yields an unsatisfiable instance.

Figure 6 shows how the running times of the solver scale with the problem size \( n \). Both axes are logarithmic, hence any polynomial function would appear as a straight line, whereas an exponential function would look convex. The plots in Figure 6 are approximately straight lines, indicating polynomial running times. Based on our computational results, we estimate that the running times are approximately quadratic in \( n \). Hence, we conclude that our solution approach has FPT-like running times, even though we did
We believe that GNRM and its special cases of security-aware and availability-aware role mining are interesting additions to the family of role mining problems. In particular, our experiments have shown that there are significant practical benefits to considering security-aware role mining, rather than MinNoise or availability-aware role mining. The time taken to solve security-aware instances is typically much smaller and the number of discrepancies it introduces is not much different from MinNoise. Indeed, the time taken to solve a security-aware instance for a fixed value of \( r \) and matrix \( UPA \) varies very little with \( k \) (as shown in Figure 2). And our experiments reported in Section 4.4 found that \( K_{\text{min}} \) was the same for a given instance irrespective of whether we used MinNoise or security-aware role mining, for all \( r \geq 4 \). Moreover, security-aware role mining has the significant advantage, from a security perspective, of not introducing any additional authorizations.

Nevertheless, role mining remains a challenging problem for all but the smallest instances. Our experiments on FPT-like running times were able to find solutions for instances containing thousands of permissions (see Figure 6), but there were very few users in those instances. Further research is required to establish whether off-the-shelf solvers and a CSP formulation of GNRM (perhaps, specifically, the security-aware variant) can be used to solve real-world role mining problems.

It should also be noted that the running times of methods for solving role mining problems do not necessarily have to be particularly fast. Role mining does not generally have to be performed in an on-line environment and does not have to be done repeatedly. (This is in contrast to algorithms to solve WSP, for example, which must respond to user requests to execute workflow steps in something approaching real time.)

Of course, we have to bear in mind that these experiments were performed on synthetic data, although this approach is often taken in research on role mining. It was important in the context of our work to be able to control the structure of the problem instances, and generating our own instances was the only viable approach. However, in future work we intend to explore whether the conclusions obtained from our work with synthetic instances, especially on the value of security-aware role mining, also hold for real-world instances.

5 RELATED WORK

Several papers have explored how matrix decomposition can be used to solve role mining problems [3, 6, 10, 11]. Fomin et al. established that MinNoise role mining (as a decision problem) is FPT by reducing the problem to one of matrix decomposition [3]. Lu et al. use matrix decomposition to develop greedy and heuristic approaches for solving MinNoise and related role mining problems [10, 11]. Exact role mining (where \( UPA = UA \land PA \)) with the optimization goal of minimizing the number of roles in the solution, has been expressed as a constraint satisfaction problem via matrix decomposition, and then solved using an SMT solver [6]. While our work uses a similar approach and tools, the problem we solve (that of minimizing the discrepancies in the solution) is different. The survey paper by Mitra et al. [12] provides an excellent overview of the many other variants of the role mining problem and techniques that have been used to solve them.

Our work extends existing work in two different ways. First, we define a more general form of noisy role mining, in which the solution may contain discrepancies of various types, defined as part of the input to the problem. In particular, we are able to specify security-aware and availability-aware role mining, which we believe are novel and useful contributions. We also introduce a new variant of noisy role mining, in which we allow arbitrary discrepancies in the GNRM solution for a small number of users and permissions. Second, we extend the work of Fomin et al. by...
showing that a search version of the matrix decomposition decision problem they used is also FPT. This, in turn, allows us to establish that our generalized noise role mining problem is FPT and will return a solution (not simply a yes or no answer).

Lu et al. [9] studied the role mining problem in the context of weighted rank-one binary matrix factorization. Particular choices of weights lead to problem instances that are analogous to availability-preserving role mining and security-preserving role mining.

6 CONCLUDING REMARKS

This paper introduces the Generalized Noise Role Mining problem (GNRM). We believe this is a useful contribution to the literature on role mining, not least because it allows us to define security- and availability-aware role mining problems. We have also defined GNRM-cov, a variant of GNRM that allows for arbitrary discrepancies between UPA and UA \( \alpha \) PA, provided those discrepancies are restricted to a small number of users and permissions. We believe GNRM-cov may be useful in practice, given that many instances do not have a solution, unless \( r \) or \( k \) (or both) is (unacceptably) large. Restricting changes to a small number of users and permissions may mean that solutions exist and may even help to identify misallocations of users and permissions in the input data. (It is well known that access control configurations may contain errors and UPA may have been derived from legacy access control systems with the intention of migrating to a role-based system [14].)

We have shown that GNRM and GNRM-cov are fixed-parameter tractable, which means that they can be solved in a reasonable amount of time, provided problem instances only require solutions in which the sum of the number of roles and the number of discrepancies from the input user-permission relation is small. Algorithms for role mining do not need to be particularly fast, but they cannot be exponential in the size of the input, given the size of typical instances. Knowing that algorithms exist that do solve role mining problems, subject to certain constraints on the solution, provides grounds for cautious optimism about the feasibility of solving real-world role mining problems.

Our experimental work provides further cause for optimism. In particular, the results on security-aware role mining suggest that it is possible to find solutions relatively quickly if we require those solutions to preserve the security of the original configuration, something that is generally desirable.

The work in this paper provides plenty of scope for future work. In particular, we would like to explore whether our work on matrix decomposition and FFT results can be extended to other variants of role mining (see [12, Section 2]). We also intend to perform further experimental work using real-world data sets (see [12, Section 5.2]).

REFERENCES


