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
eduard.eiben@rhul.ac.uk

Gregory Gutin
Royal Holloway, University of London
Egham, Surrey, United Kingdom
g.gutin@rhul.ac.uk

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

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

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

KEYWORDS

role mining, generalized noise role mining, fixed-parameter tractability

ACM Reference Format:
Jason Crampton, Eduard Eiben, Gregory Gutin, Daniel Karapetyan, and Diptapriyo Majumdar. 2018. Generalized Noise Role Mining. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/1122445.1122456

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 particlar, well-known variant of the role mining problem is *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 *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 Δ 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 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 *security-aware* role mining), while in other situations we may require that no authorizations are lost by role mining (*availability-aware* role mining).

In this paper, we introduce the GenNoise Role Mining (GNRM), of which MNRP is a special case. Moreover, Security-aware Role Mining and 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 Π is a pair (I, κ) where I is the *main part* and κ is the *parameter*; the latter is usually a non-negative integer. A parameterized problem is *fixed-parameter tractable* (FPT) if there exists a computable function f such that any instance (I, κ) 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 κ 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 *Boolean matrix* is a matrix in which all entries are either 0 or 1. Let $\vee$ and $\wedge$ 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 $\mathbf{A} \vee \mathbf{B}$ of Boolean matrices $\mathbf{A}$ and $\mathbf{B}$ is computed as usual with addition replaced by $\vee$;
(2) the product $\mathbf{A} \wedge \mathbf{B}$ of Boolean matrices $\mathbf{A}$ and $\mathbf{B}$ is computed as usual with multiplication replaced by $\wedge$ and addition by $\vee$. Thus, if $\mathbf{C} = \mathbf{A} \wedge \mathbf{B}$ then

$$c_{ij} = \bigvee_{p=1}^{n} a_{ip} \wedge b_{pj},$$

where n is the number of columns in $\mathbf{A}$ and the number of rows in $\mathbf{B}$.

Henceforth all matrices are Boolean, unless specified otherwise.

Any binary relation $X \subseteq Y \times Z$ may be represented by a matrix $\mathbf{X}$ with rows indexed by Y and columns indexed by Z, where $\mathbf{X}_{ij} = 1$ iff $(i, j) \in X$. Using matrices we can reformulate the 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 $\mathbf{UPA} = \mathbf{UA} \wedge \mathbf{PA}$. 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 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 $d_{\mathrm{H}}(\mathbf{UPA}, \mathbf{UA} \wedge \mathbf{PA}) \leq k$, where $d_{\mathrm{H}}(\mathbf{UPA}, \mathbf{UA} \wedge \mathbf{PA})$ is the number of entries in which **UPA** and $\mathbf{UA} \wedge \mathbf{PA}$ differ (i.e., the *Hamming distance* between them).

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 $\mathbf{UA} \wedge \mathbf{PA}$ 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 $\mathbf{UPA} \leq \mathbf{UA} \wedge \mathbf{PA}$, in the sense that every entry in **UPA** is less than or equal to the corresponding entry in $\mathbf{UA} \wedge \mathbf{PA}$. In other words, every permission authorized by **UPA** is also authorized by $\mathbf{UA} \wedge \mathbf{PA}$. Similarly, we could define Security-preserving Role Mining, where we require $\mathbf{UPA} \geq \mathbf{UA} \wedge \mathbf{PA}$.

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 $\mathbf{f}_{ij} \in \{\top, \bot\}$. 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 $\mathbf{a}_{ij}$ *to* entry $\mathbf{b}_{ij}$ of matrices **A** and **B** is

$$\mathrm{sd}_{\mathbf{F}}(\mathbf{a}_{ij}, \mathbf{b}_{ij}) = \begin{cases} \infty & \mathbf{f}_{ij} = \bot \text{ and } \mathbf{a}_{ij} \neq \mathbf{b}_{ij}, \\ |\mathbf{a}_{ij} - \mathbf{b}_{ij}| & \text{otherwise} \end{cases}$$

In other words, if we are *not* allowed to change $\mathbf{a}_{ij}$ in order to obtain $\mathbf{b}_{ij}$ (symbol $\bot$) and $\mathbf{a}_{ij} \neq \mathbf{b}_{ij}$ then the *F*-distance from $\mathbf{a}_{ij}$ to $\mathbf{b}_{ij}$ is ∞. Otherwise, it is just $|\mathbf{a}_{ij} - \mathbf{b}_{ij}|$.

We define $\mathrm{sd}_{\mathbf{F}}(\mathbf{A}, \mathbf{B}) = \sum_{i=1}^{m} \sum_{j=1}^{n} \mathrm{sd}_{\mathbf{F}}(\mathbf{a}_{ij}, \mathbf{b}_{ij})$. Thus, the *distance* from **A** to **B** is finite if and only if for every $(i, j) \in [m] \times [n]$ such that $\mathbf{a}_{ij} \neq \mathbf{b}_{ij}$ we have $\mathbf{f}(a_{ij}) = \top$.

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 $r \times n$ role-permission assignment matrix **PA** such that $\mathrm{sd}_{\mathbf{F}}(\mathbf{UPA}, \mathbf{UA} \wedge \mathbf{PA}) \leq k$? 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 $\mathbf{f}_{ij} = \top$ for all $(i, j) \in [m] \times [n]$;
- Availability-preserving Role Mining if $\mathbf{f}_{ij} = \top$ if and only if $\mathbf{UPA}_{ij} = 0$; and
- Security-preserving Role Mining if $\mathbf{f}_{ij} = \top$ if and only if $\mathbf{UPA}_{ij} = 1$.

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 $\{I_1, \ldots, I_p\}$ of $[m]$ and a partition $\{J_1, \ldots, J_q\}$ of $[n]$ such that for every $i \in [p]$, $j \in [q]$, $s \in I_i, t \in J_j$, 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 $v(\mathbf{L})$ and finally replacing every block **L** by one entry of value $v(\mathbf{L})$.

For a example, let $\mathbf{P} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$. Then $\mathbf{Q}_1$ and $\mathbf{Q}_2$ below are both **P**-matrices: permuting columns 2 and 3 in each matrix, then partitioning (into blocks of equal size for $\mathbf{Q}_1$, and between rows 1 and 2 and columns 2 and 3 for $\mathbf{Q}_2$) and "contracting" gives us **P**.

$$\mathbf{Q}_1 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \qquad \mathbf{Q}_2 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

> 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 $\mathrm{sd}_F(\mathbf{A}, \mathbf{B}) \leq k$? 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 $\mathbf{f}_{ij} = \top$ 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* $\mathbf{P}$ *be a* $p \times q$ *matrix. Then, every* $\mathbf{P}$*-matrix* $\mathbf{B}$ *has at most* p *pairwise distinct rows and at most* q *pairwise distinct columns.*

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

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

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

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

This lemma implies the following reduction/preprocessing rule.

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

To simplify an instance $(\mathbf{A}, \mathbf{F}, \mathbf{P}, k)$ of Generalized $\mathbf{P}$-Matrix Approximation, we can apply the following reduction rule exhaustively. If a row $\mathbf{a}_i$ is deleted by the reduction rule, the label matrix $\mathbf{F}$ does not change for the other rows. This means that for the reduced instance with matrix $\mathbf{A}'$, the label matrix is $\mathbf{F}$ restricted to the rows of $\mathbf{A}'$ i.e. $\{\mathbf{a}_1, \ldots, \mathbf{a}_m\} \setminus \{\mathbf{a}_i\}$. For simplicity of presentation, the label matrix for $\mathbf{A}'$ will still be denoted by $\mathbf{F}$.

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

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

We say two instances of Generalized $\mathbf{P}$-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 $\mathbf{P}$-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 $\mathbf{P}$-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* $(\mathbf{A}, \mathbf{F}, \mathbf{P}, k)$ *be an instance of* Generalized $\mathbf{P}$-Matrix Approximation*. Then, there exists a polynomial-time algorithm that either returns "no-instance" or transforms* $(\mathbf{A}, \mathbf{F}, \mathbf{P}, k)$ *into an equivalent instance* $(\mathbf{A}', \mathbf{F}, \mathbf{P}, k)$ *of* Generalized $\mathbf{P}$-Matrix Approximation*. Moreover the following properties are satisfied.*

1. *The matrix* $\mathbf{A}'$ *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* $\mathbf{P}$*-matrix* $\mathbf{B}'$ *such that* $\mathrm{sd}_{\mathbf{F}}(\mathbf{A}', \mathbf{B}') \leq k$*, in polynomial time we can compute a* $\mathbf{P}$*-matrix* $\mathbf{B}$ *such that* $\mathrm{sd}_{\mathbf{F}}(\mathbf{A}, \mathbf{B}) \leq k$.

Proof. Let $(\mathbf{A}, \mathbf{P}, k)$ be an input instance of Generalized $\mathbf{P}$-Matrix Approximation. As $m \geq p$ and $n \geq q$, if $\mathbf{A}$ has at most $p-1$ rows or has at most $q-1$ columns, then there is no $m \times n$ $\mathbf{P}$-matrix $\mathbf{B}$ such that $\mathrm{sd}_{\mathbf{F}}(\mathbf{A}, \mathbf{B}) \leq k$. 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 $\mathbf{A}$. If $\mathbf{A}$ 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 $\mathbf{A}'$ be the obtained matrix. We also obtain a stack S which contains all deleted rows and columns.

We return $(\mathbf{A}', \mathbf{F}, \mathbf{P}, k)$ as the output instance. Clearly, $\mathbf{A}'$ has at most $p + k$ pairwise distinct rows and at most $q + k$ pairwise distinct columns. Moreover, $\mathbf{A}'$ has at least p rows and at least q columns. Also, $\mathbf{A}'$ can have at most $\max\{p, k\} + 1$ pairwise identical rows and at most $\max\{q, k\} + 1$ pairwise identical columns. This means that $\mathbf{A}'$ 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 $\mathbf{B}'$ is a $\mathbf{P}$-matrix such that $\mathrm{sd}_{\mathbf{F}}(\mathbf{A}', \mathbf{B}') \leq k$. Note that at any intermediate stage, when a row $\mathbf{r}$ (a column $\mathbf{c}$, respectively) was deleted from $\mathbf{A}$, there were at least $(\max\{p, k\} + 1)$ additional rows identical to $\mathbf{r}$ (at least $(\max\{q, k\} + 1)$ additional columns identical to $\mathbf{c}$, respectively). Hence, in $\mathbf{A}'$, if a row $\mathbf{r}$ or a column $\mathbf{c}$ was deleted by Reduction Rule 2, then there are exactly $\max\{p, k\} + 1$ rows identical to $\mathbf{r}$ (exactly $\max\{q, k\} + 1$ columns identical to $\mathbf{c}$, respectively). Since $\mathrm{sd}_{\mathbf{F}}(\mathbf{A}', \mathbf{B}') \leq k$, at most k entries with label $\top$ identical to $\mathbf{r}$ ($\mathbf{c}$, respectively) were modified in $\mathbf{B}'$. Thus, $\mathbf{B}'$ must have at least one row identical to $\mathbf{r}$ (at least one column identical to $\mathbf{c}$, respectively) if $\mathbf{r}$ ($\mathbf{c}$, 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 in $\mathbf{A}'$ and $\mathbf{B}'$. Thus, reinstating the deleted rows and columns using stack S, we obtain matrices $\mathbf{A}$ and $\mathbf{B}$ such that $\mathbf{B}$ is a $\mathbf{P}$-matrix and $\mathrm{sd}_{\mathbf{F}}(\mathbf{A}, \mathbf{B}) \leq k$. □

Theorem 2.5. Generalized $\mathbf{P}$-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 $(\mathbf{A}, \mathbf{F}, \mathbf{P}, k)$ be an instance of Generalized $\mathbf{P}$-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 $(\mathbf{A}', \mathbf{F}, \mathbf{P}, k)$ satisfying properties (1) and (2). Recall that the first property says that $\mathbf{A}'$ has at most $(\max\{p, k\}+1)(p+k)$ rows, and at most $(\max\{q, k\}+1)(q+k)$ columns. This means that $\mathbf{A}'$ 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 $\mathbf{B}'$. We then invoke Proposition 2.2 to check whether $\mathbf{B}'$ is a $\mathbf{P}$-matrix or not. This checking takes $2^{p \log p + q \log q}(nm)^{O(1)}$-time. If $\mathbf{B}'$ is a $\mathbf{P}$-matrix,

it is a solution to Generalized **P**-Matrix Approximation for the instance $(\mathbf{A}', \mathbf{F}, \mathbf{B}', k)$ as $\mathrm{sd}_{\mathbf{F}}(\mathbf{A}', \mathbf{B}') \leq k$ (since we changed at most k entries in $\mathbf{A}'$). Then, we make use of property (2) to construct $\mathbf{B}$ satisfying $\mathrm{sd}_{\mathbf{F}}(\mathbf{A}, \mathbf{B}) \leq k$ and return $\mathbf{B}$ as a solution of Generalized **P**-Matrix Approximation for the instance $(\mathbf{A}, \mathbf{F}, \mathbf{B}, k)$. Recall that this step takes polynomial time. Hence, the overall algorithm takes $2^{p \log p + q \log q}(nm)^{O(1)}((\max\{p,k\}+1)(p+k)(\max\{q,k\}+1)(q+k))^k$ time. □

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 $\mathbf{P}$ could provide the basis for a solution to GNRM. The number of such pairs is bounded above by a function of r. For each such $\mathbf{P}$, we determine whether the Generalized **P**-Matrix Approximation instance $(\mathbf{UPA}, \mathbf{F}, \mathbf{P}, k)$ has a solution, in which case we can then compute a solution to the GNRM instance.

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

Proof. As $\mathbf{B}$ is a $\mathbf{P}$-matrix, there are partitions $\{I_1, \ldots, I_p\}$ of $[m]$ and $\{J_1, \ldots, J_q\}$ of $[n]$ such that for every $i \in [p], j \in [q]$, $s \in I_i, t \in I_j$, $\mathbf{b}_{st} = \mathbf{p}_{ij}$.

We initialize $\mathbf{X}^* = \mathbf{X}$ and $\mathbf{Y}^* = \mathbf{Y}$. Consider an entry $\mathbf{p}_{ij}$. Let $\mathbf{x}_i$ be the i'th row of $\mathbf{X}$ and $\mathbf{y}^j$ the j'th column of $\mathbf{Y}$; then $\mathbf{x}_i \wedge \mathbf{y}^j = \mathbf{p}_{ij}$. Let $c \in [p]$ and $d \in [q]$ such that $i \in I_c$ and $j \in J_d$. Then, for any $s \in I_c$ and $t \in J_d$, set $b_{st} = \mathbf{p}_{ij}$. Then, for any $s \in I_c$ and for any $t \in I_d$, we insert $\mathbf{x}_i$ as the s'th row of $\mathbf{X}^*$ and $\mathbf{y}^j$ as the t'th column of $\mathbf{Y}^*$. □

The *Boolean rank* of a matrix $\mathbf{A}$, denoted $\mathrm{BRank}(\mathbf{A})$, is the minimum natural number r such that $\mathbf{A} = \mathbf{B} \wedge \mathbf{C}$, where $\mathbf{B}$ and $\mathbf{C}$ are matrices such that the number of columns in $\mathbf{B}$ and the number of rows in $\mathbf{C}$ is r. Thus, a matrix $\mathbf{A}$ has Boolean rank 1 if and only if $\mathbf{A} = \mathbf{x} \wedge \mathbf{y}^T$ for some column-vectors $\mathbf{x}$ and $\mathbf{y}$. In fact, $\mathrm{BRank}(\mathbf{A}) = r$ if and only if r is the minimum natural number such that $\mathbf{A} = \mathbf{X}^{(1)} \vee \cdots \vee \mathbf{X}^{(r)}$, where matrices $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(r)}$ are of Boolean rank 1 [8].

Theorem 2.7. Generalized Noise Role Mining *admits an $O^*(2^{O(r2^r+rk)})$-time algorithm.*

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

Moreover, an $m \times n$-matrix $\mathbf{B}$ is of rank r if r is the minimum natural number such that $\mathbf{B} = \mathbf{C} \wedge \mathbf{D}$, where $\mathbf{C}$ is an $m \times r$-matrix and $\mathbf{C}$ 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 $\mathbf{P}$ of Boolean rank r and an $m \times n$ $\mathbf{P}$-matrix $\mathbf{B}$ such that $\mathrm{sd}_{\mathbf{F}}(\mathbf{UPA}, \mathbf{B}) \leq k$ and if $\mathbf{B}$ does exist then find matrices $\mathbf{UA}$ and $\mathbf{PA}$ of sizes $m \times r$ and $r \times n$, respectively, such that $\mathbf{B} = \mathbf{UA} \wedge \mathbf{PA}$.

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

1. Generate all pairs $(\mathbf{X}, \mathbf{Y})$ of matrices of sizes $p \times r$ and $r \times q$, respectively, and for each such pair compute $\mathbf{P} = \mathbf{X} \wedge \mathbf{Y}$;
2. For each $\mathbf{P}$, solve Generalized **P**-Matrix Approximation for the instance $(\mathbf{UPA}, \mathbf{F}, \mathbf{P}, k)$. If $(\mathbf{UPA}, \mathbf{F}, \mathbf{P}, k)$ is a yes-instance, then using the algorithm of Lemma 2.6 return matrices $\mathbf{UA}$ and $\mathbf{PA}$ of sizes $m \times r$ and $r \times n$ such that $\mathbf{B} = \mathbf{UA} \wedge \mathbf{PA}$, where $\mathbf{B}$ is the solution of the instance $(\mathbf{UPA}, \mathbf{F}, \mathbf{P}, k)$;
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 $2^{O(r2^r)}$ pairs $(\mathbf{X}, \mathbf{Y})$, and we can compute all matrices $\mathbf{P}$ in time $2^{O(r2^r)}$. 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 $\mathbf{P}$ (it is equal to $2^{O(r2^r)}$) times the maximum running time of solving Generalized **P**-Matrix Approximation on an instance $(\mathbf{UPA}, \mathbf{F}, \mathbf{P}, k)$ and computing $\mathbf{UA}$ and $\mathbf{PA}$, 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 $O^*(2^{O(r2^r+rk)})$. It remains to observe that $2^{O(r2^r)} \cdot O^*(2^{O(r2^r+rk)}) = O^*(2^{O(r2^r+rk)})$. □

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 limit 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 $\mathbf{UA}$ and $\mathbf{PA}$ (with r rows and columns, respectively) such that the Hamming distance between $\mathbf{UPA}$ and $\mathbf{UA} \wedge \mathbf{PA}$ is at most k, our goal is to find $\mathbf{UA}$ and $\mathbf{PA}$ such that after making changes restricted to entries in at most cov_U rows and cov_P columns of $\mathbf{UPA}$, the modified $\mathbf{UPA}$ is equal to $\mathbf{UA} \wedge \mathbf{PA}$. The parameters cov_U and cov_P are called the *user covering number* and *permission covering number*, respectively.

[1] For $r = 1$, since $\mathbf{B}^{(1)} = \mathbf{x}^1 \wedge (\mathbf{y}^1)^T$, $\mathbf{B}^{(1)}$ has at most two distinct rows, $\mathbf{y}^1$ and the all-zero row, and has at most two distinct columns, $\mathbf{x}^1$ and the all-zero column. Now let $r \geq 2$. By induction hypothesis, $\mathbf{B} = \mathbf{B}^{(\leq r-1)} \vee \mathbf{B}^{(r)}$, where $\mathbf{B}^{(\leq r-1)}$ has at most 2^{r-1} rows and columns and $\mathbf{B}^{(r)}$ at most two rows and columns. Since every row (column, respectively) of $\mathbf{B}$ is the disjunction of the corresponding rows (columns, respectively) of $\mathbf{B}^{(\leq r-1)}$ and $\mathbf{B}^{(r)}$, the number of distinct rows (columns, respectively) in $\mathbf{B}$ is at most $2^{r-1} \cdot 2 = 2^r$.

Generalized Noise Cover Role Mining (GNRM-cov)
Input: An $m \times n$ user-permission assignment matrix **UPA**, a label matrix **F**, and nonnegative integers r, cov_U and cov_P.
Parameter: $r + \mathsf{cov}_U + \mathsf{cov}_P$.
Goal: Is there an $m \times r$ user-role assignment matrix **UA** and an $r \times n$ role-permission assignment matrix **PA** such that there exists $I \subseteq [m]$ and $J \subseteq [n]$ where $|I| \leq \mathsf{cov}_U$, $|J| \leq \mathsf{cov}_P$, and if $\mathbf{UPA}_{ij} \neq (\mathbf{UA} \wedge \mathbf{PA})_{ij}$, then (1) $\mathbf{f}_{ij} = \top$, and (2) either $i \in I$ or $j \in J$? If the answer is yes, then return such matrices **UA** and **PA**.

Example 3.1. Consider an instance of Security-preserving Role Mining with

$$\mathbf{UPA} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

It is not hard to see that if $k = 0$ the minimum number of roles is 4. If $k = 1$ then by flipping the entry $(1, 4)$ to 0, we reduce the minimum number of roles to 3. For $k = 2$ we cannot reduce the minimum number of roles to 2, but we can for $k = 3$, by flipping the entries $(1, 4), (2, 1), (2, 3)$.

Now consider it as an instance of Security-preserving Cover Role Mining. If $\mathsf{cov} = 2$, we reduce the minimum number of roles to 2 by flipping some entries in rows 1 and 2, namely the entries $(1, 4), (2, 1), (2, 3)$, in contrast to Security-preserving Role Mining, where the minimum number of roles is 3 for $k = 2$.

After Theorem 3.4, which shows that three important subproblems of GNRM-cov are FPT, we will discuss how the FPT algorithm of Theorem 3.4 can be used for variations of the subproblems of GNRM-cov.

We show that the approach in Section 2 extends to an FPT algorithm for the important subproblems of GNRM-cov as well. The number of roles r is again a part of the parameter and Lemma 2.6 still applies. Hence it is natural to introduce the following problem.

Generalized Cover **P**-Matrix Approximation
Input: An $m \times n$ matrix **A**, a label matrix **F**, a $p \times q$ matrix **P** and nonnegative integers cov_U and cov_P.
Goal: Is there an $m \times n$ **P**-matrix **B** such that there exists $I \subseteq [m]$ and $J \subseteq [n]$ where $|I| \leq \mathsf{cov}_U$, $|J| \leq \mathsf{cov}_P$, and if $\mathbf{a}_{ij} \neq \mathbf{b}_{ij}$, then (1) $\mathbf{f}_{ij} = \top$, and (2) either $i \in I$ or $j \in J$. If the answer is yes, then return such a matrix **B**.

Lemma 3.2. *If* **A** *has at least* $\mathsf{cov}_U + p \cdot 2^{\mathsf{cov}_P} + 1$ *pairwise distinct rows or at least* $\mathsf{cov}_P + q \cdot 2^{\mathsf{cov}_U} + 1$ *pairwise distinct columns, then* $(\mathbf{A}, \mathbf{F}, \mathbf{P}, \mathsf{cov}_U, \mathsf{cov}_P)$ *is a no-instance of* Generalized Cover **P**-Matrix Approximation.

Proof. Let **B** be a **P**-matrix and $I \subseteq [m], J \subseteq [n]$ such that $|I| \leq \mathsf{cov}_U$, $|J| \leq \mathsf{cov}_P$, and if $\mathbf{a}_{i,j} \neq \mathbf{b}_{i,j}$, then $i \in I$ or $j \in J$. Suppose that $(\mathbf{A}, \mathbf{F}, \mathbf{P}, \mathsf{cov}_P, \mathsf{cov}_U)$ is a yes-instance. Then such a **P**-matrix **B** and sets $I \subseteq [m]$ and $J \subseteq [n]$ exist. Observe that **B** has at most p pairwise distinct rows and at most q pairwise distinct columns. Using this fact, we show that **A** has at most $|I| + 2^{|J|}p$ pairwise distinct rows and at most $|J| + 2^{|I|}q$ pairwise distinct columns.

Let $i \in [m] \setminus I$. Note that $\mathbf{a}_{ij} = \mathbf{b}_{ij}$ whenever $j \in [n] \setminus J$. Therefore the i-th row of **A**, denoted $\mathbf{a}_i$, can differ from the i-th row of **B**, denoted $\mathbf{b}_i$, only in the coordinates in J. Therefore, there exists $J' \subseteq J$ such that $\mathbf{a}_{ij} = 1 - \mathbf{b}_{ij}$ for all $j \in J'$ and $\mathbf{a}_{ij} = \mathbf{b}_{ij}$ for all $j \in [n] \setminus J'$. It follows that $\mathbf{a}_i$ is one of the $2^{|J|}$ many vectors we obtain from $\mathbf{b}_i$ by flipping the value of the entries in $J' \subset J$. Since there are only p many distinct rows in **B**, it follows that for all $i \in [m] \setminus I$, the row $\mathbf{a}_i$ is one of the $2^{|J|}p$ many vectors. Recall that $|I| \leq \mathsf{cov}_U$ and $|J| \leq \mathsf{cov}_P$. It follows that **A** has at most $\mathsf{cov}_U + p \cdot 2^{\mathsf{cov}_P}$ pairwise distinct rows. Using an analogous argument, we obtain that **A** has at most $\mathsf{cov}_P + q \cdot 2^{\mathsf{cov}_U}$ pairwise distinct columns. Since we started from an arbitrary solution, it follows that if **A** has at least $\mathsf{cov}_U + p \cdot 2^{\mathsf{cov}_P} + 1$ pairwise distinct rows or at least $\mathsf{cov}_P + q \cdot 2^{\mathsf{cov}_U} + 1$ pairwise distinct columns, then $(\mathbf{A}, \mathbf{F}, \mathbf{P}, \mathsf{cov}_U, \mathsf{cov}_P)$ is a no-instance. □

The above lemma implies the following reduction/preprocessing rule.

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

Unfortunately, even when 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**, 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 cov_U many rows in some set I and at most 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 $\max\{p, \mathsf{cov}_U\} + 1$ rows identical to **r** in the matrix $\mathbf{A}'$ obtained from **A** by removing **r**. Now assume that we have a **P**-matrix $\mathbf{B}'$ such that there exist $I \subseteq [m]$ and $J \subseteq [n]$, such that $|I| \leq \mathsf{cov}_U$, $|J| \leq \mathsf{cov}_P$, and if $\mathbf{a}'_{ij} \neq \mathbf{b}'_{ij}$, then $\mathbf{f}_{ij} = \top$ and either $i \in I$ or $j \in J$. Clearly there is $i \in [n] \setminus I$ such that $\mathbf{a}'_i$, the i-th row of $\mathbf{A}'$, is identical to **r**. But, $\mathbf{b}'_i$ (the i-th row of $\mathbf{B}'$) can still differ from $\mathbf{a}'_i$ at the entries in J. Moreover $\mathbf{f}_{ij} = \top$ whenever $\mathbf{a}'_{ij} \neq \mathbf{b}'_{ij}$ for some $j \in J$. This might not be true if we simply used $\mathbf{b}'_i$ 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 $\max\{p, \mathsf{cov}_U\} + 1$ identical rows. Indeed, if we are allowed to modify the permission of a user in the $\max\{p, \mathsf{cov}_U\} + 1$ 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)* $|R| = \max\{p, \mathsf{cov}_U\} + 1$, *(3) for all* $i' \in R$ *we have that the row* $\mathbf{a}_{i'}$ *is identical to* $\mathbf{a}_i$, *and (4) for all* $i' \in R$ *and all* $j \in [n]$ *we have that* $\mathbf{f}_{i'j} = \top$ *implies that* $\mathbf{f}_{ij} = \top$. *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 $(\mathbf{A}, \mathbf{F}, \mathbf{P}, \mathsf{cov}_U, \mathsf{cov}_P)$ from a solution $\mathbf{B}'$ for $(\mathbf{A}', \mathbf{F}', \mathbf{P}, \mathsf{cov}_U, \mathsf{cov}_P)$ obtained by applying Reduction Rule 4 once on the i-th row of **A**. Let $I' \subseteq [m - 1]$

and $J' \subseteq [n]$ be the sets witnessing that $\mathbf{B}'$ is a solution. As $\max\{p, \mathsf{cov}_U\} + 1 > |I'|$, one of $\max\{p, \mathsf{cov}_U\} + 1$ many rows in R is not in I'. Hence its replacement in $\mathbf{B}'$ differs from the original only in the entries in J'. Since the i-th row in $\mathbf{A}$ is identical to this row and the label matrix $\mathbf{F}$ 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 a solution for 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 $\mathbf{A}$ is identical to the j-th column, and (4) for all $j' \in C$ and all $i \in [m]$ we have that $\mathbf{f}_{ij'} = \top$ implies that $\mathbf{f}_{ij} = \top$. Then delete the j-th column from the matrix $\mathbf{A}$.*

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 $\mathbf{A}$ and their corresponding rows (columns) in $\mathbf{F}$ are also identical, then Reduction Rule 4 (Reduction Rule 5) applies. In particular, in the cases when the label matrix $\mathbf{F}$ is availability-preserving, security-preserving, or contains only $\top$ on all entries, we get that if the two rows/columns are identical in $\mathbf{A}$, then they are identical in $\mathbf{F}$. 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 GNRM-cov: Noise Cover Role Mining , Availability-preserving Cover Role Mining and Security-preserving Cover Role Mining. Recall that for the first subproblem above $\mathbf{f}_{ij} = \top$ for all $(i, j) \in [m] \times [n]$, for the second one (the third one, respectively), $\mathbf{f}_{ij} = \top$ if and only if $\mathbf{a}_{ij} = 0$ ($\mathbf{a}_{ij} = 1$, respectively).

Theorem 3.3. Noise/Availability-preserving/Security-preserving Cover **P**-Matrix Approximation *parameterized by $p + q + \mathsf{cov}_U + \mathsf{cov}_P$ can be solved in 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. Noise/Availability-preserving/Security-preserving Cover Role Mining *parameterized by $p + q + \mathsf{cov}_U + \mathsf{cov}_P$ admits an FPT algorithm.*

Note that the FPT algorithm of Theorem 3.4 implies the existence of FPT algorithms for the three important subproblems of the following variation of GNRM-cov. The problem is similar to GNRM-cov, but where we have cov as a parameter instead of cov_U and cov_P. We can run the algorithm of Theorem 3.4 for every pair cov_U and 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 cov_U and 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 GNRM expressed as a constraint satisfaction problem (CSP). Our experiments were designed to provide insights into the following four questions:

- How does the choice of $\mathbf{F}$ 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 **UPA** and $\mathbf{F}$?
- How does the time required to solve GNRM vary with k and r?
- Does a general-purpose solver using the CSP formulation of GNRM exhibit FPT-like running times?

We first present our formulation of 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 GNRM

We present a CSP formulation of the GNRM in order to solve the problem using a general-purpose solver. We could have implemented a bespoke FPT algorithm to solve 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 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 FPT [7].

Our formulation consists of two matrices of Boolean variables $ua_{i,\ell}$, $i \in [m]$, $\ell \in [r]$, and $pa_{\ell,j}$, $\ell \in [r]$, $j \in [n]$. We also use a matrix of auxiliary Boolean variables $d_{i,j}$, $i \in [m]$, $j \in [n]$, representing the discrepancies between **UA** $\wedge$ **PA** and **UPA**. We use the values 0 and 1 to represent the values $\bot$ and $\top$, respectively, in the $\mathbf{F}$ 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 **UA** $\wedge$ **PA** – and link these variables to the $ua_{i,\ell}$ and $pa_{\ell,j}$ variables. Then it would be easy to formulate the constraints and link the decision variables to variables $d_{i,j}$. However, knowing the values of $upa_{i,j}$ and $f_{i,j}$ 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 $upa_{i,j}$ and $f_{i,j}$.[2]

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 $upa_{i,j} = 0$:			
$ua_{i,\ell} + pa_{\ell,j} \leq 1$	$\forall \ell \in [r]$,	(1)	Since the discrepancy is not allowed, either $ua_{i,\ell}$ or $pa_{\ell,j}$ has to be zero for every ℓ.
For all $i \in [m]$ and $j \in [n]$ such that $f_{i,j} = 0$ and $upa_{i,j} = 1$:			
$x_{i,j,\ell} \leq ua_{i,\ell}$	$\forall \ell \in [r]$,	(2)	We need $ua_{i,\ell} = pa_{\ell,j} = 1$ for some ℓ. We enforce that at least one of $x_{i,j,1}..x_{i,j,r}$ is 1 and also if $x_{i,j,\ell} = 1$ then $ua_{i,\ell} = pa_{\ell,j} = 1$.
$x_{i,j,\ell} \leq pa_{\ell,j}$	$\forall \ell \in [r]$,	(3)	
$\sum_{\ell \in [r]} x_{i,j,\ell} \geq 1$	$\forall \ell \in [r]$,	(4)	
For all $i \in [m]$ and $j \in [n]$ such that $f_{i,j} = 1$ and $upa_{i,j} = 0$:			
$ua_{i,\ell} + pa_{\ell,j} \leq 1 + d_{i,j}$	$\forall \ell \in [r]$,	(5)	If both $ua_{i,\ell}$ and $pa_{\ell,j}$ are ones for some ℓ then this is a discrepancy.
For all $i \in [m]$ and $j \in [n]$ such that $f_{i,j} = 1$ and $upa_{i,j} = 1$:			
$ua_{i,\ell} \geq x_{i,j,\ell}$	$\forall \ell \in [r]$,	(6)	If either $ua_{i,\ell} = 0$ or $pa_{\ell,j} = 0$ for every ℓ, this is a discrepancy. Auxiliary variable $x_{i,j,\ell}$ is forced to 0 if either $ua_{i,\ell} = 0$ or $pa_{\ell,j} = 0$. If $x_{i,j,\ell} = 0$ for every ℓ then we force $d_{i,j} = 1$.
$pa_{\ell,j} \geq x_{i,j,\ell}$	$\forall \ell \in [r]$,	(7)	
$\sum_{\ell \in [r]} x_{i,j,\ell} \geq 1 - d_{i,j}$	$\forall \ell \in [r]$,	(8)	
$\sum_{i \in [m]} \sum_{j \in [n]} d_{i,j} \leq k$,		(9)	The number of discrepancies is restricted by k.
$x_{i,j,\ell} \in \{0,1\}$	$\forall i \in [m],\ \forall j \in [n],\ \forall \ell \in [r]$,	(10)	Auxiliary variables, see the cases where $upa_{i,j} = 1$.
$d_{i,j} \in \{0,1\}$	$\forall i \in [m],\ \forall j \in [n]$,	(11)	Indicates whether there is a discrepancy in the corresponding element.
$ua_{i,j} \in \{0,1\}$	$\forall i \in [m],\ \forall j \in [r]$,	(12)	Defines the **UA** matrix.
$pa_{i,j} \in \{0,1\}$	$\forall i \in [r],\ \forall j \in [n]$.	(13)	Defines the **PA** matrix.

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 $[0, \lfloor 0.25n \rfloor]$; 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: $f_{i,j} = upa_{i,j}$;
Availability: $f_{i,j} = 1 - upa_{i,j}$;
Noise: $f_{i,j} = 1$ for every i and j.

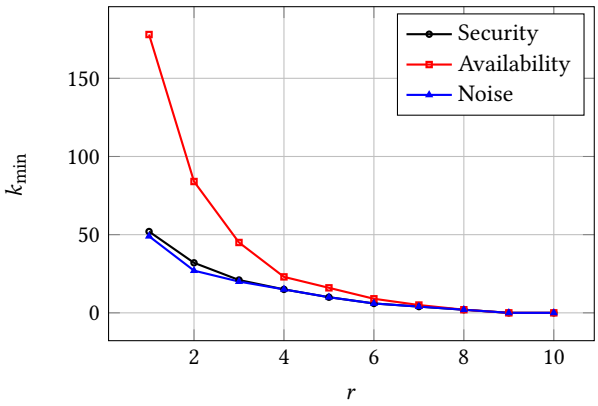
To solve GNRM, we used the CP-SAT solver from Google's OR-Tools, version 9.2.9972. The experiments were conducted on a machine based on two Xeon E5-2630 v2 CPUs (2.60 GHz), with 32 GB of RAM. Hyper-threading was enabled, and the solver was allowed to use all the cores. The solver was controlled from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

4.3 Relationship between k and r

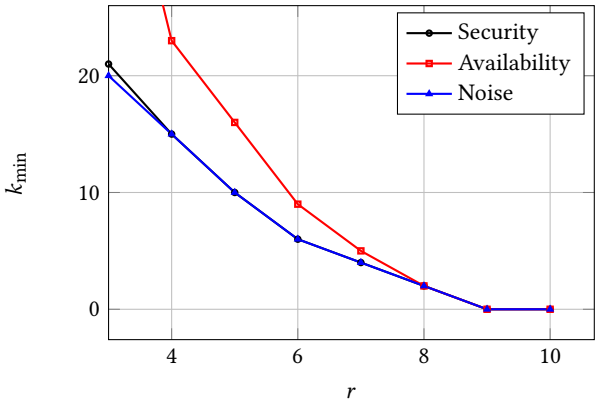
In our first experiment, we consider the relationship between k and r. Specifically, we study how $k_{\min}$ – the minimum value of k needed for the instance to have a solution – depends on the number of roles r that we attempt to extract. For this experiment, we used an instance of size $m = 20$ and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

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

Observe that $k_{\min}$ is bounded by $\sum_i \sum_j upa_{i,j}$ in the Noise and Security instances; however, it is bounded by $nm - \sum_i \sum_j upa_{i,j}$ in the Availability instances. Since the UPA matrix is usually sparse, $k_{\min}$ 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.



(a) Full plot



(b) Zoomed in version of Figure 2a

Figure 2: Relationship between $k_{\min}$ and r ($n = m = 20$)

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 $k_{\min}$ 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 $k = k_{\min} - 1$, 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

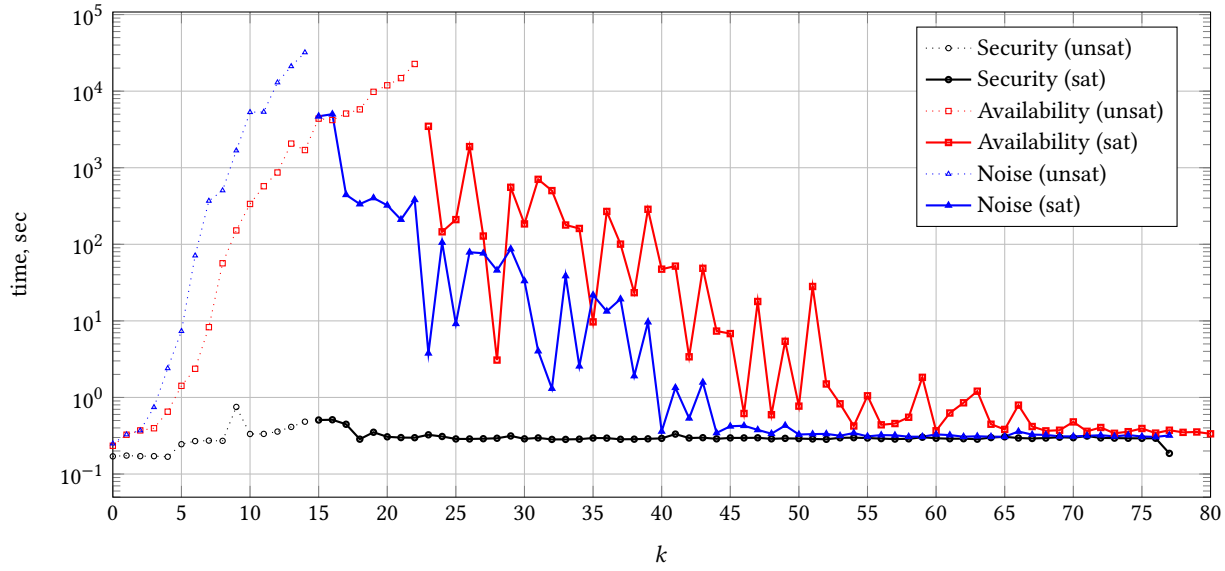


Figure 3: Relationship between running time and k ($r = 4$, $n = 20$)

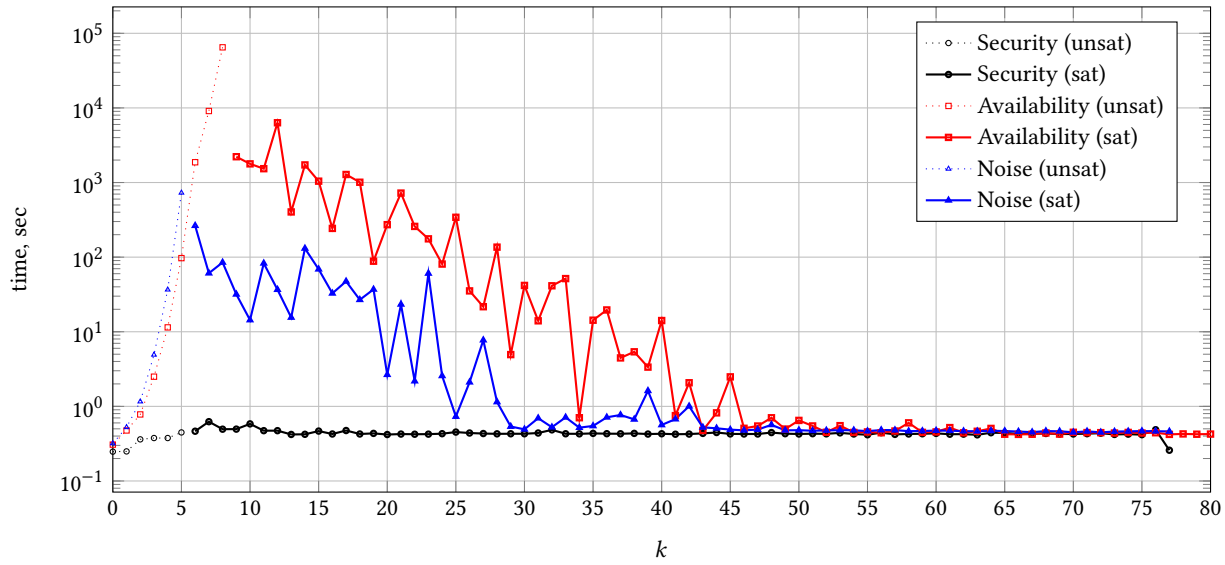


Figure 4: Relationship between running time and k ($r = 6$, $n = 20$)

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 finesses 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

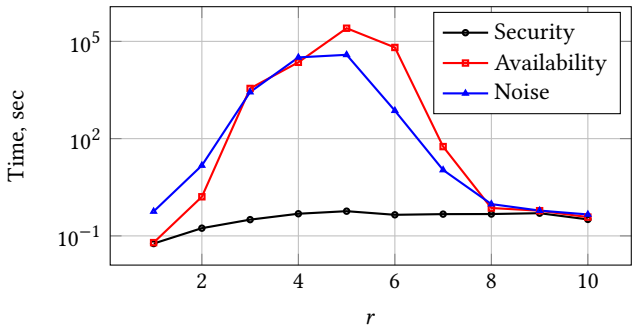


Figure 5: Relationship between running time and r in the phase transition region

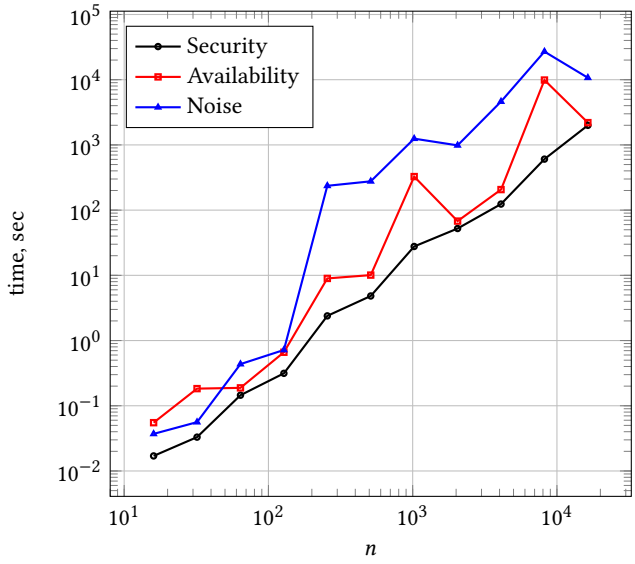


Figure 6: Test for FPT-like behaviour of the solver ($m = 8$, $r + k = 10$)

not explicitly exploit the FPT structure of the problem in our CSP formulation. Similar results have been obtained in the context of the workflow satisfiability problem (WSP) [7], where the CSP-based solver had FPT-like running time even though the CSP formulation of WSP was conventional (and did not make use of the theoretical FPT algorithm for WSP).

4.6 Discussion

We belive 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 $k_{\min}$ 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** ∧ **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** $\wedge$ **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 exponentional 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 FPT 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

[1] Chen Dan, Kristoffer Arnsfelt Hansen, He Jiang, Liwei Wang, and Yuchen Zhou. 2018. Low Rank Approximation of Binary Matrices: Column Subset Selection and Generalizations. In *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018 (LIPIcs, Vol. 117)*, Igor Potapov, Paul G. Spirakis, and James Worrell (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 41:1–41:16.

[2] David F. Ferraiolo, Ravi S. Sandhu, Serban I. Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. 2001. Proposed NIST standard for role-based access control. *ACM Trans. Inf. Syst. Secur.* 4, 3 (2001), 224–274.

[3] Fedor V. Fomin, Petr A. Golovach, and Fahad Panolan. 2020. Parameterized low-rank binary matrix approximation. *Data Min. Knowl. Discov.* 34, 2 (2020), 478–532.

[4] Nicolas Gillis and Stephen A. Vavasis. 2018. On the Complexity of Robust PCA and ℓ_1-norm Low-Rank Matrix Approximation. *Mathematics of Operations Research* 43, 4 (2018), 1072–1084.

[5] David A. Gregory, Norman J. Pullman, Kathryn F. Jones, and J. Richard Lundgren. 1991. Biclique coverings of regular bigraphs and minimum semiring ranks of regular matrices. *J. Comb. Theory, Ser. B* 51, 1 (1991), 73–89.

[6] Jafar Haadi Jafarian, Hassan Takabi, Hakim Touati, Ehsan Hesamifard, and Mohamed Shehab. 2015. Towards a General Framework for Optimal Role Mining: A Constraint Satisfaction Approach. In *SACMAT*. ACM, 211–220.

[7] Daniel Karapetyan, Andrew J. Parkes, Gregory Gutin, and Andrei Gagarin. 2019. Pattern-Based Approach to the Workflow Satisfiability Problem with User-Independent Constraints. *Journal of Artificial Intelligence Research* 66 (2019), 85–122. https://doi.org/10.1613/jair.1.11339

[8] Ki Hang Kim. 1982. *Boolean Matrix Theory and Applications*. Monographs and Textbooks in Pure and Applied Mathematics, Vol. 70. Marcel Dekker, New York.

[9] Haibing Lu, Xi Chen, Junmin Shi, Jaideep Vaidya, Vijayalakshmi Atluri, Yuan Hong, and Wei Huang. 2020. Algorithms and Applications to Weighted Rank-one Binary Matrix Factorization. *ACM Trans. Manag. Inf. Syst.* 11, 2 (2020), 7:1–7:33.

[10] Haibing Lu, Jaideep Vaidya, and Vijayalakshmi Atluri. 2008. Optimal Boolean Matrix Decomposition: Application to Role Engineering. In *ICDE*. IEEE Computer Society, 297–306.

[11] Haibing Lu, Jaideep Vaidya, and Vijayalakshmi Atluri. 2014. An optimization framework for role mining. *J. Comput. Secur.* 22, 1 (2014), 1–31. https://doi.org/10.3233/JCS-130484

[12] Barsha Mitra, Shamik Sural, Jaideep Vaidya, and Vijayalakshmi Atluri. 2016. A Survey of Role Mining. *ACM Comput. Surv.* 48, 4 (2016), 50:1–50:37.

[13] Ian M. Molloy, Ninghui Li, Tiancheng Li, Ziqing Mao, Qihua Wang, and Jorge Lobo. 2009. Evaluating role mining algorithms. In *SACMAT*. ACM, 95–104.

[14] Ian M. Molloy, Ninghui Li, Yuan (Alan) Qi, Jorge Lobo, and Luke Dickens. 2010. Mining roles with noisy data. In *SACMAT*. ACM, 45–54.

[15] Gustaf Neumann and Mark Strembeck. 2002. A scenario-driven role engineering process for functional RBAC roles. In *SACMAT*. ACM, 33–42.

[16] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. 1996. Role-Based Access Control Models. *Computer* 29, 2 (1996), 38–47.

[17] Emre Uzun, Vijayalakshmi Atluri, Haibing Lu, and Jaideep Vaidya. 2011. An Optimization Model for the Extended Role Mining Problem. In *DBSec (Lecture Notes in Computer Science, Vol. 6818)*. Springer, 76–89.

[18] Jaideep Vaidya, Vijayalakshmi Atluri, Janice Warner, and Qi Guo. 2010. Role Engineering via Prioritized Subset Enumeration. *IEEE Transactions on Dependable and Secure Computing* 7, 3 (2010), 300–314. https://doi.org/10.1109/TDSC.2008.61