BINARY MATRIX-BASED PREPROCESSING FOR SCALABLE ROLE MINING: A COMPARATIVE EVALUATION ON UPA DATASETS

Nurul Syazwina Mohd Yusran, Nazirah Abd Hamid and Norhidayah Muhammad

Faculty of Informatics Computing, Universiti Sultan Zainal Abidin (UniSZA), Tembila, Terengganu

ABSTRACT

RBAC is widely used to manage access permissions in organisational systems. However, role mining becomes difficult when the UPA matrix contains repeated and redundant structures. This study compares three data-mining-based preprocessing algorithms:UPA Aggregation, GreConD Boolean Matrix Decomposition (BMD), and K-Medoids with Jaccard distanceto reduce binary UPA matrices before downstream role detection and selection. The algorithms were evaluated using nine prepared binary UPA datasets based on representation reduction, information preservation, and runtime. K-Medoids Jaccard achieved the highest average matrix-only reduction (90.22%) but lower preservation (0.8738). GreConD BMD achieved 72.15% reduction and 0.7831 preservation. UPA Aggregation achieved 89.26% reduction and full preservation (1.0000). After three warm-up runs, each algorithm–dataset experiment was repeated 30 times; the mean core runtimes across datasets were 0.0091 s for UPA Aggregation, 1.8500 s for GreConD BMD, and 5.5552 s for K-Medoids Jaccard. The findings indicate that UPA Aggregation offers the most reliable trade-off for reducing the input matrix when exact access-control information must be maintained. The findings support UPA Aggregation as a preprocessing method for subsequent role-mining stages, while its effect on final role quality remains to be evaluated..

KEYWORDS

Role Mining, RBAC, UPA, Preprocessing, Scalability

1. INTRODUCTION

Role-Based Access Control (RBAC) is a well-established access-control model for managing authorization in large information systems. It assigns permissions through organisational roles rather than directly to individual users [1], [2]. In large-scale organisations, users often require different combinations of permissions based on their job functions, responsibilities, and departments. As digital services expand, the numbers of users, permissions, applications, and access transactions also increase. Consequently, manual access-control administration becomes difficult to maintain and more prone to error. Without a clear mechanism for managing these permissions, organisations may face inconsistent access policies, excessive privileges, and higher administrative overhead. Scalable and efficient access-control management is therefore required to support operational efficiency and system security.

Role mining is a practical approach for supporting RBAC implementation by identifying candidate roles from existing user-permission data [3], [4]. Access-control information is commonly represented as a User-Permission Assignment (UPA) matrix, where rows represent users, columns represent permissions, and binary values indicate whether a user has a particular permission [4], [5]. This representation supports analysis of access patterns and identification of

groups of users or permissions that may form meaningful roles. When the numbers of users and permissions grow, however, the UPA matrix may become high-dimensional, sparse, and redundant. This condition increases the search space and computational cost of role discovery. Scalability therefore remains an important issue when role mining is applied to large access-control datasets [5], [6], [8].

As a possible solution, preprocessing can be applied before role detection to reduce the size and redundancy of the UPA matrix. In this study, preprocessing is not limited to conventional data cleaning. It refers to binary matrix-based reduction that produces a smaller representation for later role mining while retaining the user-permission relationships required for reconstruction. Several related approaches are suitable for this purpose because they operate on binary structures [7], [10]–[13].

Previous studies have examined role mining and related reduction methods, but direct comparative evidence remains limited. Many studies apply different algorithms to different datasets or assess different outputs, which makes direct comparison difficult [6], [9], [14]. A method that performs well on one dataset may be less suitable for another dataset with different dimensions, sparsity, redundancy, or permission distribution. In addition, much of the literature evaluates final role-mining outputs, including role minimization, optimisation, hierarchy construction, and restructuring [15]–[18]. Preprocessing algorithms that reduce the binary UPA input before role detection are less frequently evaluated as a comparable stage in their own right.

Accordingly, this study contributes a controlled comparison of three existing reduction paradigms using the same prepared matrices, a common reconstruction-based preservation measure, repeated runtime measurements, and both matrix-only and mapping-aware representation costs.

The present study evaluates UPA Aggregation, GreConD BMD, and K-Medoids Jaccard using nine binary UPA datasets. The evaluation considers representation reduction, information preservation, and runtime. Representation reduction is examined using both matrix-only reduction and mapping-aware logical reduction, thereby considering the compactness of the reduced matrix together with the additional mappings required for reconstruction. Information preservation measures the consistency between the original and reconstructed assignments, while runtime reflects the computational cost of preprocessing under the same experimental environment.

By reducing redundant structures before role detection, these methods are intended to provide a more compact and manageable UPA representation for subsequent role-mining stages. The present evaluation therefore focuses on the effectiveness of the preprocessing representation, while its influence on the quality and interpretability of the final mined roles remains to be examined in downstream evaluation.

This study addresses three research questions: (i) to what extent are the selected preprocessing algorithms suitable for binary UPA datasets; (ii) how do the algorithms differ in representation reduction, information preservation, and runtime; and (iii) which method offers the most suitable trade-off before role detection? The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 describes the methodology and experimental environment. Section 4 presents the results and discussion. Section 5 concludes the paper and outlines future work.

2. RELATED WORK

RBAC provides a systematic mechanism for assigning permissions through roles instead of direct user-permission assignments [1], [2]. Roles normally represent job functions, responsibilities, or access requirements. This arrangement becomes increasingly useful as the number of users, systems, and privileges grows. In practice, however, organisations may accumulate access permissions over time through operational changes, staff movement, system expansion, and policy updates. As a result, a large and difficult-to-manage UPA structure may exist before a formal RBAC policy is established.

Role mining addresses this problem by identifying candidate roles from existing user-permission assignments. Early approaches such as ORCA and RoleMiner demonstrated that bottom-up analysis can extract role structures from access-control data [3], [4]. Vaidya et al. subsequently formalized the Role Mining Problem as finding a minimal descriptive role set for existing assignments [5]. Later work extended role mining toward role minimization, subset enumeration, semantic role discovery, hierarchy construction, and optimisation [6], [19].

The input to role mining is commonly represented as a binary UPA matrix. This representation permits role mining to be studied through data mining, matrix decomposition, graph methods, and clustering. Lu et al. modelled role engineering using BMD, where the UPA matrix is decomposed into user-role and role-permission factor matrices [7]. This factorisation directly connects binary matrix analysis with the RBAC assignment structure.

Scalability remains a major concern because large matrices may be sparse, redundant, and expensive to search. Abolfathi et al. proposed a scalable graph-embedding and unsupervised-learning approach [8], while RMPlib was introduced to support more controlled benchmarking of role-mining problem instances [9]. Recent studies have also considered bi-objective optimisation, overlapping clustering, role clustering, edge minimization, and maximal biclique enumeration [15]–[18], [21]. Most of these methods focus on the generated role structure rather than a reduced UPA representation used as input to a later stage.

BMD is relevant to binary UPA preprocessing because it represents a binary matrix using Boolean factors. GreConD is a greedy concept-based method for Boolean matrix factorisation [10]. In a role-mining context, the factors can be interpreted in a form analogous to user-role and role-permission assignments [7]. Its reconstruction quality, however, depends on the number and capability of the extracted factors.

Clustering is also relevant because users with similar permission patterns may indicate related access requirements. Distance-based reduction has previously been considered for role-mining scalability [11], while recent methods use overlapping clustering and role clustering to generate or reorganize roles [16], [17]. Clustering methods are normally approximate: users assigned to one cluster may not have identical permission vectors, which can introduce false-positive or false-negative assignments during reconstruction.

Structural aggregation forms a third direction. Anderer described preprocessing that aggregates users and permissions into equivalence classes to reduce the UPA dimensions without losing information [12]. Related work also notes that users with identical permission sets can be represented by a class row while identifiers are retained through mappings [13]. This approach reduces exact redundancy rather than merging merely similar patterns.

Preprocessing is treated as a distinct evaluation problem in this paper because it transforms the security-sensitive matrix that will be supplied to role discovery. Work on probabilistic role

mining distinguishes the observed assignment matrix from a reconstructed representation and evaluates false-positive and false-negative assignments [23]. This distinction is relevant here: an input reduction that changes a zero to one may introduce an unauthorized assignment, while a change from one to zero may remove an operationally required assignment. Therefore, preprocessing must be evaluated not only by compactness but also by reconstruction behaviour.

The three selected methods consequently represent lossless structural aggregation, Boolean factorisation, and similarity-based clustering. Their common binary input and reconstructable outputs make it possible to compare reduction, preservation, and runtime under one experimental framework.

3. METHODOLOGY AND EXPERIMENTAL SETUP

This section describes the experimental framework used to evaluate binary matrix-based preprocessing for UPA data. The objective is to identify a method that reduces the matrix representation while retaining the original user-permission relationships required for subsequent role detection and selection.

3.1. Experimental Framework

The experiment begins with UPA datasets provided in spreadsheet format. Each dataset is loaded and standardised as a binary matrix. The same prepared matrix is then processed independently by UPA Aggregation, GreConD BMD, and K-Medoids Jaccard. Each method produces a reduced representation and a reconstructed UPA matrix. The reconstruction is compared with the original matrix, after which reduction, preservation, and runtime are reported.

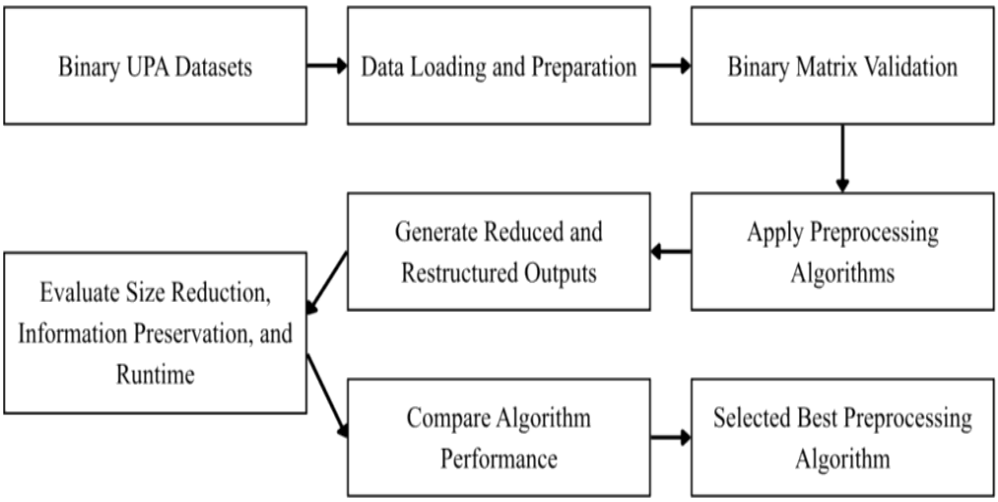

Figure 1. Overall preprocessing framework.

This framework ensures that all methods use the same prepared data and evaluation definitions. Performance differences can therefore be interpreted primarily in relation to the reduction strategy rather than differences in data preparation.

3.2. Dataset Preparation

Nine UPA matrices derived from real-world access-control datasets commonly used in role-mining research were evaluated in this study [11], [19], [22], [23]. Each matrix was standardised

into binary form, where a value of 1 indicates that a user is assigned a permission, while a value of 0 indicates that the permission is not assigned. Missing, empty, or invalid entries encountered during data preparation were treated as 0. Empty user rows and permission columns were retained to ensure that all three algorithms were evaluated using the same input dimensions and structure.

Formally, each UPA dataset is represented as a binary matrix:

$$M \in \{0,1\}^{mxn}$$

where m represents the number of users, n represents the number of permissions, and each entry M_{ij} is defined as:

$$M_{ij} = \begin{cases} 1, & \text{if user i is assigned permission j} \\ 0, & otherwise \end{cases}$$

The APJ and Americas matrices represent network access-control configurations derived from Cisco firewall profiles, while Firewall1 and Firewall2 were generated from access policies associated with Check Point firewalls. The Domino dataset contains user and access profiles from a Lotus Domino server, the Customer dataset represents an access-control matrix from an HP customer, and the Healthcare dataset originates from the U.S. Veterans Administration [22], [23], [26]. Americas2 and DominoV1 were included as prepared variants of the corresponding Americas and Domino matrices used in the present experiment.

Table 1. Structural characteristics of the prepared UPA matrices.

Dataset	**User**	**Permissions**	**Matrix Size**	**Positive assignments**	**Density (%)**	**Unique user patterns**	**Unique permission patterns**
americas.xlsx	3,478	1,587	5,519,586	105,206	1.9060	260	350
americas2.xlsx	1,588	3,477	5,521,476	105,206	1.9054	349	259
apj.xlsx	2,045	1,164	2,380,380	6,842	0.2874	565	578
customer.xlsx	10,962	284	3,113,208	45,428	1.4592	5,657	277
Domino.xlsx	79	231	18,249	728	3.9394	24	39
DominoV1.xlsx	79	231	18,249	728	3.9394	24	39
fire1.xlsx	709	365	258,785	31,951	12.3465	86	90
fire2.xlsx	591	325	192,075	36,429	18.9660	12	12
hc.xlsx	46	46	2,116	1,486	70.2268	18	19

The matrices vary substantially in structure. Density ranges from 0.2874% for APJ to 70.2268% for HC. Fire2 has only 12 unique user patterns among 591 users, whereas Customer has 5,657 unique patterns among 10,962 users. The prepared Customer matrix also contains 940 empty user rows and 7 empty permission columns; Domino and DominoV1 each contain 2 empty user rows and 1 empty permission column. These characteristics were retained because they can influence exact aggregation, cluster formation, and factor coverage.

3.3. Preprocessing Algorithms and Selection

The study was designed as a focused comparison rather than an exhaustive benchmark of all available preprocessing techniques. The three algorithms were selected based on four criteria: compatibility with binary UPA matrices, representation of distinct reduction paradigms, the ability to produce a compact representation that can be reconstructed to the original user-permission dimensions, and uitability for evaluation under a common framework based on representation reduction, information preservation, and runtime. These criteria are important because differences in algorithm outputs and optimisation objectives can complicate direct comparison under a single experimental setting [6], [19]. Accordingly, UPA Aggregation, GreConD BMD, and K-Medoids Jaccard were selected to represent exact equivalence-class aggregation, Boolean matrix decomposition, and distance-based prototype clustering, respectively.

UPA Aggregation was selected as the lossless baseline because it groups only identical user and permission vectors while retaining the mappings required to recover the original identifiers. This strategy measures the amount of exact structural redundancy in the UPA matrix without changing the original permission assignments [12], [13].

GreConD BMD was selected as the factorisation baseline because Boolean matrix decomposition is structurally compatible with binary UPA data and produces factors analogous to user-role and role-permission matrices [7], [10]. Its inclusion also enables the study to examine the trade-off between compact factor representation and reconstruction coverage.

K-Medoids Jaccard was selected as the clustering baseline because k-medoids represents each cluster using an observed data object rather than a synthetic mean and can operate with arbitrary dissimilarities, including Jaccard distance [24], [25]. For binary permission data, this allows each medoid to remain an actual user-permission profile, although users assigned to the same medoid may have non-identical permission sets.

Other preprocessing approaches may also be applicable to binary UPA data. However, including methods with different representation forms, input requirements, parameter settings, or optimisation objectives would require a broader and method-specific experimental design. The present study therefore focuses on three representative reduction paradigms that can be evaluated consistently using the same input matrices, reconstruction dimensions, and performance measures.

Table 2. Summary of the selected preprocessing algorithms.

Algorithm	**Main strategy**	**Reduced representation**	**Reconstruction**
UPA Aggregation	Groups identical user rows and permission columns	User-class × permission-class matrix	Class mappings
GreConD BMD	Decomposes the UPA matrix into Boolean factors	User-factor and factor-permission matrices	Boolean matrix product
K-Medoids Jaccard	Groups similar users using Jaccard distance	Medoid prototype matrix	User-to-cluster mapping

3.3.1. UPA Aggregation

UPA Aggregation is a lossless structural reduction method. It groups users with identical permission vectors and permissions with identical user-assignment vectors. Each user class and permission class is represented by one row or column. The original matrix can be reconstructed exactly using the class mappings.

Algorithm 1. UPA Aggregation

Input: Binary UPA matrix M

Output: Reduced matrix RM and user/permission class mappings

1. Read the binary UPA matrix M.
2. Identify users with identical permission vectors and form user classes.
3. Select one representative user for each user class.
4. Identify permissions with identical user-assignment vectors and form permission classes.
5. Select one representative permission for each permission class.
6. Construct RM using the representative users and permissions.
7. Return RM and the class mappings.

3.3.2. GreConD BMD

GreConD BMD decomposes the binary UPA matrix into a user-factor matrix UA and a factor-permission matrix PA. The reconstructed matrix is obtained using Boolean matrix multiplication. GreConD is used here as a decomposition-based preprocessing baseline. Its preservation depends on whether the permitted factors cover the positive assignments in the original matrix.

Algorithm 2. GreConD BMD

Input: Binary UPA matrix M, maximum factors $K_{max} = 100$, minimum gain = 1
Output: UA, PA, and reconstructed matrix $\hat{M}$

1. Initialize the residual matrix R = M and empty factor matrices.
2. Search the residual matrix for the Boolean concept with the largest gain.
3. Append the selected user factor and permission factor to UA and PA.
4. Remove newly covered positive entries from R.
5. Repeat until R is empty, gain <1, or K_{max} is reached.
6. Reconstruct $\hat{M}$ using the Boolean product UA $\odot$ PA.
7. Return UA, PA, and $\hat{M}$.

3.3.3. K-Medoids Jaccard

K-Medoids Jaccard groups users according to the Jaccard distance between their binary permission vectors. Each cluster is represented by an observed medoid profile. Users assigned to one medoid may nevertheless have non-identical permission vectors, so reconstruction is approximate.

$$k = \min(N_u, K_{max}, \max(2, \lceil r \times N_u \rceil)), \text{ where } r = 0.5 \text{ and } K_{max} = 300$$

Algorithm 3. K-Medoids Jaccard

Input: Binary UPA matrix M, reduction factor r = 0.5, maximum clusters = 300
Output: Medoid matrix MD, user-cluster mapping, and reconstructed matrix $\hat{M}$

1. Represent each user as a binary permission vector.
2. Extract unique user-permission patterns.
3. Determine k from the number of unique patterns.
4. Initialize k medoids using observed user vectors.

5. Compute Jaccard distances and assign each pattern to its nearest medoid.
6. Update medoids until convergence or 20 iterations are reached.
7. Reconstruct $\hat{M}$ by replacing each user with its assigned medoid.
8. Return MD, the user-cluster mapping, and $\hat{M}$.

3.4. Evaluation Metrics

The algorithms were evaluated in terms of representation reduction, information preservation, and runtime. Representation reduction was assessed using two complementary measures: matrix-only reduction and mapping-aware logical reduction. Matrix-only reduction measures the compactness of the core reduced representation produced by each algorithm, whereas mapping-aware logical reduction also includes the auxiliary mappings required to restore the original user-permission dimensions.

Matrix-only reduction is calculated as:

$$\text{Matrix-only reduction}(\%) = \left(1 - \frac{C_{red}}{C_{orig}}\right) \times 100$$

where $C_{orig}$is the number of cells in the original UPA matrix and $C_{red}$is the number of cells in the reduced representation.
Mapping-aware logical reduction is calculated as:

$$\text{Mapping-aware logical reduction}(\%) = \left(1 - \frac{C_{red} + C_{map}}{C_{orig}}\right) \times 100$$

where C_{map} represents the number of mapping entries required for reconstruction.

For UPA Aggregation, C_{red} is the size of the user-class × permission-class matrix, while C_{map}includes the user-class and permission-class mappings. For GreConD BMD, Cred is the total number of cells in the user-factor and factor-permission matrices, and no separate external mapping is required because the factor matrices remain aligned with the original users and permissions. For K-Medoids Jaccard, C_{red} is the size of the medoid matrix, while C_{map} contains one cluster assignment for each original user. Since matrix values and mapping indices may use different data types, a byte-based sensitivity analysis was also performed using `uint8` matrix entries and `int32` mapping entries.

Information preservation is measured using matrix-level Jaccard similarity over positive assignments:

$$\text{Jaccard}(M, \hat{M}) = \frac{|M \cap \hat{M}|}{|M \cup \hat{M}|} = \frac{TP}{(TP + FP + FN)}$$

where M is the original UPA matrix and$\hat{M}$is the reconstructed matrix. A value of 1.0000 indicates exact reconstruction. A false positive occurs when the reconstructed matrix contains an assignment that was absent from the original matrix, whereas a false negative occurs when an original assignment is absent from the reconstruction. In an access-control context, false positives may introduce excessive access, while false negatives may remove required access.

Runtime was measured as the duration of the core preprocessing operation in seconds. File loading, file saving, diagnostics, and visualisation were excluded from the reported core runtime. Reconstruction time was measured separately and was not included in the main runtime comparison.

3.5. Experimental Environment and Runtime Protocol

The experiments were implemented in Python on 64-bit Windows 11 using an AMD64 processor with 12 logical CPU cores. NumPy 2.4.2, pandas 3.0.1, and openpyxl 3.1.5 were used. The binary matrices were stored as uint8, and mapping arrays were stored as int32. Although the machine contained an NVIDIA GeForce RTX 4050 Laptop GPU, all three algorithms were executed on the CPU and did not use GPU computation.

For each algorithm–dataset pair, three warm-up executions were completed, followed by 30 measured repetitions. The random state was fixed at 42 where applicable. The reported dataset-wise runtime is the arithmetic mean ± sample standard deviation of the 30 core executions, and 95% confidence intervals were retained in the experimental records. This protocol was used to obtain stable estimates of the runtime performance.

UPA Aggregation required no value of k. GreConD used a maximum of 100 factors and a minimum gain of 1. K-Medoids used k = ceil(0.5 × unique user patterns), with a minimum of 2 and a cap of 300; the maximum number of iterations was 20.

3.6. Algorithm Selection Criteria

The final preprocessing recommendation was based on the combined interpretation of representation reduction, preservation, and runtime. Information preservation was treated as the primary criterion because a reduced UPA representation should not silently change security-sensitive assignments. Representation reduction indicates the compactness achieved through preprocessing, while runtime reflects the associated computational overhead. A method was therefore considered suitable when it substantially reduced the representation, preserved the original assignments to the greatest possible extent, and completed the preprocessing within a reasonable runtime.

4. RESULTS AND DISCUSSION

This section presents the overall and dataset-wise results. The discussion focuses on representation fairness, structural dataset characteristics, preservation errors, and repeated runtime measurements.

4.1. Overall Performance

Table 3. Average performance across the nine prepared matrices (mean ± SD across datasets).

Algorithm	Matrix-only reduction (%)	Mapping-aware logical reduction (%)	Preservation	Core runtime (s)	Interpretation
UPA Aggregation	89.26 ± 15.85	88.22 ± 15.98	1.0000 ± 0.0000	0.0091 ± 0.0117	High reduction, lossless, fastest
K-Medoids Jaccard	90.22 ± 6.57	89.76 ± 6.96	0.8738 ± 0.1424	5.5552 ± 9.2362	Highest reduction, approximate
GreConD BMD	72.15 ± 21.55	72.15 ± 21.55	0.7831 ± 0.3286	1.8500 ± 2.2096	Moderate reduction, variable

					preservation

K-Medoids Jaccard achieved the highest average matrix-only reduction (90.22%), followed closely by UPA Aggregation (89.26%). GreConD BMD achieved a lower mean reduction of 72.15%. The ranking changed little after mappings were included: K-Medoids remained highest at 89.76%, UPA Aggregation reached 88.22%, and GreConD remained at 72.15%.

The mapping-aware byte sensitivity analysis produced average reductions of 88.38% for K-Medoids Jaccard, 85.13% for UPA Aggregation, and 72.15% for GreConD BMD. Thus, excluding mappings slightly overstated the compactness of UPA Aggregation and K-Medoids, but it did not reverse their overall reduction ranking. This result supports retaining matrix-only reduction as the structural measure while reporting mapping-aware cost as a fairness check.

Preservation differentiates the methods more clearly. UPA Aggregation reconstructed every assignment exactly. K-Medoids retained most assignments on average but introduced both false positives and false negatives. GreConD showed the greatest variability, with exact reconstruction on five matrices but poor coverage on several larger or structurally complex matrices under the 100-factor limit.

4.2. Representation Reduction Analysis

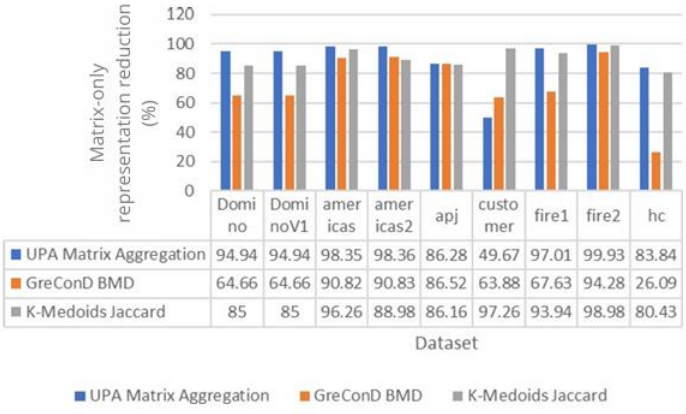

Figure 2. Dataset-wise matrix-only representation reduction.

All three algorithms reduced every matrix, but the reduction mechanism produced different patterns. UPA Aggregation achieved 99.93% reduction on Fire2, 98.36% on Americas2, 98.35% on Americas, and 97.01% on Fire1. These matrices contain substantial exact redundancy. Fire2, for example, contains only 12 unique user patterns among 591 users and only 12 unique permission patterns among 325 permissions.

Customer produced the lowest UPA Aggregation reduction (49.67%). This result is consistent with its 5,657 unique user patterns and 277 unique permission patterns. Exact aggregation cannot merge profiles that differ by even one permission, so the lower reduction reflects limited exact redundancy rather than instability of the method.

K-Medoids achieved 97.26% reduction on Customer because the 5,657 unique user profiles were represented using the capped number of medoids. This aggressive reduction is possible because

clustering groups similar rather than identical profiles. The preservation results show the cost of that approximation.

GreConD reduction ranged from 26.09% on HC to 94.28% on Fire2. Its compactness depends on the number of factors needed relative to the matrix dimensions. A factor representation is beneficial when many positive assignments can be covered by a small set of Boolean concepts; the benefit decreases when the representation requires many factors.

4.3. Information Preservation and Reconstruction Errors

Table 4. Dataset-wise matrix Jaccard preservation.

Dataset	UPA Aggregation	GreConD BMD	K-Medoids Jaccard
Domino.xlsx	1.0000	1.0000	0.7704
DominoV1.xlsx	1.0000	1.0000	0.7704
americas.xlsx	1.0000	0.7378	0.9862
americas2.xlsx	1.0000	0.7966	0.9807
apj.xlsx	1.0000	0.4469	0.7827
customer.xlsx	1.0000	0.0665	0.6105
fire1.xlsx	1.0000	1.0000	0.9797
fire2.xlsx	1.0000	1.0000	0.9986
hc.xlsx	1.0000	1.0000	0.9853

UPA Aggregation achieved 1.0000 preservation on all nine matrices. This follows directly from its equivalence-class construction: only identical rows and columns are merged, and the mappings restore the original dimensions without changing assignments.

K-Medoids obtained an average preservation of 0.8738. It performed strongly on Fire2 (0.9986), Americas (0.9862), HC (0.9853), Americas2 (0.9807), and Fire1 (0.9797). Its lowest value occurred on Customer (0.6105), where reconstruction introduced 5,414 false positives and 14,390 false negatives. The cluster cap produced a very small prototype representation, but many distinct profiles were forced to share medoids.

GreConD achieved exact reconstruction on Domino, DominoV1, Fire1, Fire2, and HC. It was less successful on Americas (0.7378), Americas2 (0.7966), APJ (0.4469), and particularly Customer (0.0665). On Customer, 42,409 of the 45,428 original positive assignments remained uncovered after the factor limit was reached. GreConD introduced no false positives in these experiments; its preservation loss was caused by false negatives resulting from incomplete positive coverage.

These error directions have different practical meanings. K-Medoids may both add and remove assignments because a user is replaced by a medoid profile. GreConD, under the implemented coverage procedure, produced a conservative subset of assignments but may omit many required permissions when factors are insufficient. UPA Aggregation avoids both types of reconstruction error.

4.4. Runtime Efficiency

Table 5. Core runtime by dataset (mean ± sample SD over 30 repetitions, seconds).

Dataset	UPA Aggregation	GreConD BMD	K-Medoids Jaccard
americas.xlsx	0.015216 ± 0.001300	5.073332 ± 0.036643	2.188302 ± 0.573566
americas2.xlsx	0.013501 ± 0.000959	5.524706 ± 0.112015	10.412017 ± 2.942815
apj.xlsx	0.010031 ± 0.000682	2.639595 ± 0.035093	10.015261 ± 0.658811
customer.xlsx	0.036286 ± 0.002276	2.441115 ± 0.024985	27.325784 ± 0.264078
Domino.xlsx	0.001063 ± 0.000253	0.033777 ± 0.003177	0.002383 ± 0.000319
DominoV1.xlsx	0.001013 ± 0.000096	0.034900 ± 0.004483	0.002395 ± 0.000155
fire1.xlsx	0.002084 ± 0.000259	0.815503 ± 0.017775	0.032743 ± 0.001180
fire2.xlsx	0.001883 ± 0.000466	0.074601 ± 0.005115	0.016363 ± 0.000387
hc.xlsx	0.000835 ± 0.000094	0.012534 ± 0.000417	0.001170 ± 0.000314

The repeated measurements confirm that UPA Aggregation has the lowest computational overhead on every dataset. Its highest mean runtime was 0.036286 s on Customer. GreConD was slowest on Americas2 (5.524706 s) and Americas (5.073332 s), where it extracted the maximum 100 factors. K-Medoids was slowest on Customer (27.325784 s), Americas2 (10.412017 s), and APJ (10.015261 s). These matrices contain many unique profiles, and the clustering procedure repeatedly computes distances and updates medoids.

The standard deviations are generally small relative to the means, although K-Medoids shows greater variability on Americas and Americas2. This is consistent with its more intensive iterative assignment and update operations. The runtime comparison therefore supports UPA Aggregation as the lowest-overhead preprocessing option under the current CPU implementation.

4.5. Trade-off Analysis and Algorithm Selection

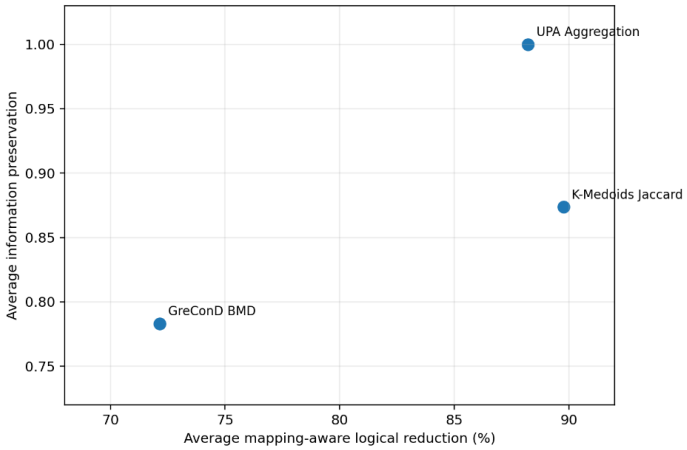

Figure 3. Trade-off between mapping-aware reduction and information preservation.

If reduction alone is considered, K-Medoids appears slightly better than UPA Aggregation. However, its average preservation is lower and its reconstruction can add or remove assignments. UPA Aggregation provides nearly the same reduction while preserving every assignment. The mapping-aware analysis confirms that this comparison remains similar after the required mappings are counted.

GreConD remains a relevant binary factorisation baseline, but the current 100-factor configuration did not provide consistent preservation across the nine matrices. Increasing the factor limit could improve coverage, but it would also enlarge the representation and change the reduction-runtime trade-off.

Considering the three criteria together, UPA Aggregation is selected as the most suitable preprocessing method for the next research stage. The decision is based on exact reconstruction, high mapping-aware reduction, and the lowest repeated runtime. Downstream evaluation is therefore required to determine how the selected preprocessing representation affects the quality of the final mined roles.

5. CONCLUSION

This study compared UPA Aggregation, GreConD BMD, and K-Medoids Jaccard as preprocessing methods for binary UPA matrices. All three algorithms reduced the matrices, but they differed in reduction strength, reconstruction behaviour, and runtime. K-Medoids produced the highest average reduction but introduced approximate assignments. GreConD provided a Boolean factor representation but showed variable preservation under the fixed factor limit. UPA Aggregation achieved high reduction, exact preservation, and the lowest runtime.

The mapping-aware analysis showed that counting reconstruction mappings slightly reduced the reported compactness of UPA Aggregation and K-Medoids but did not change the main interpretation. Dataset structure also explained much of the observed behaviour: exact redundancy favoured aggregation, many distinct profiles favoured aggressive clustering but increased reconstruction error, and factor adequacy determined GreConD coverage.

Based on the combined evaluation of representation reduction, information preservation, and runtime, UPA Aggregation is recommended as the preprocessing method for the next stage of the study. Future work will evaluate its downstream effects on role count, role quality, interpretability, administrative usefulness, and mining time to determine whether the reduced representation improves the complete role-mining pipeline without compromising access-control information.

REFERENCES

[1] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, “Role-based access control models,” Computer, vol. 29, no. 2, pp. 38–47, Jan. 1996, doi: 10.1109/2.485845.

[2] ANSI/INCITS,RoleBasedAccessControl,ANSIINCITS359-2004/INCITS359-2012.

[3] J. Schlegelmilch and U. Steffens, “Role mining with ORCA,” *Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies*, pp. 168–176, Jun. 2005, doi: 10.1145/1063979.1064008.

[4] J. Vaidya, V. Atluri, and J. Warner, “RoleMiner: Mining roles using subset enumeration,” Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 144–153, Oct. 2006, doi: 10.1145/1180405.1180424.

[5] J.Vaidya, V. Atluri,and Q.Guo,“Therolemining problem:finding aminimaldescriptive setof roles,” Proceedings of the 12th ACM Symposium on Access Control Models and Technologies,pp.175–184,Jun.2007,doi:10.1145/1266840.1266870.

[6] I. Molloy, N. Li, T. Li, Z. Mao, Q. Wang, and J. Lobo, “Evaluating role mining algorithms,” Proceedingsof the 14th ACM Symposium on AccessControl Modelsand Technologies, pp. 95–104, Jun. 2009, doi: 10.1145/1542207.1542224.

[7] H. Lu, J. Vaidya, and V. Atluri, “Optimal Boolean Matrix Decomposition: Application to Role Engineering,” 2008 IEEE 24th InternationalConferenceonData Engineering,pp.297–306,Apr. 2008, doi: 10.1109/icde.2008.4497438.

[8] M. Abolfathi, Z. Raghebi, J. H. Jafarian, and F. Banaei-Kashani, "An Accurate and ScalableRole Mining Algorithm based on Graph Embedding and Unsupervised Feature Learning," Proceedings of the Annual Hawaii International Conference on System Sciences/Proceedings of the Annual Hawaii International Conference on System Sciences, Jan. 2020, doi: 10.24251/hicss.2020.242.

[9] S. Anderer, B. Scheuermann, S. Mostaghim, P. Bauerle, and M. Beil, "RMPlib: A Library of Benchmarks for the Role Mining Problem," Proceedings of the 26th ACM Symposium onAccess Control Models and Technologies, pp. 3–13, Jun. 2021, doi: 10.1145/3450569.3463566.

[10] M. Trnecka and R. Vyjidacek, "Revisiting the GreCon algorithm for Boolean matrix factorisation," *Knowledge-Based Systems*, vol. 249, p. 108895, May 2022, doi: 10.1016/j.knosys.2022.108895.

[11] N. A. Hamid, S. R. Selamat, R. Ahmad, and M. Mohamad, "Hamming distance approach to reduce role mining scalability," International Journal of Advanced Computer Science and Applications, vol. 14, no. 6, Jan. 2023, doi: 10.14569/ijacsa.2023.0140654.

[12] S. Anderer, "Role Mining for Industrial-strength ERP Systems Using Evolutionary Algorithms," Ph.D. dissertation, Otto von Guericke University Magdeburg, Germany, 2023.

[13] S. Anderer, T. Kempter, B. Scheuermann, and S. Mostaghim, "Dynamic optimisation of role concepts for Role-Based access control using evolutionary algorithms," *SN Computer Science*, vol. 4, no. 4, May 2023, doi: 10.1007/s42979-023-01805-1.

[14] N. A. Hamid, A. Alarood, A.A. Manaf, and R. Ahmad,"Assessing role mining algorithmsusing RMiner tool in identifying the best model for access control," *Journal of Theoretical andApplied Information Technology*, vol. 96, no. 20, pp. 6948–6954, Oct. 2018.

[15] J.Crampton,E.Eiben,G.Gutin,D.Karapetyan,andD.Majumdar,"Bi-objectiveoptimisationin role mining," ACM Transactions on Privacy and Security, vol. 28, no. 1, pp. 1–22, Oct. 2024, doi: 10.1145/3697833.

[16] Y.Yang,J.Li,T.Zhang,L.Chen,G.Huang,andZ.Lv,"IRMAOC:aninterpretablerolemining algorithm based on overlapping clustering," Cybersecurity, vol. 8, no. 1, Jan. 2025, doi: 10.1186/s42400-024-00348-z.

[17] O. Durdag and A. Coskuncay, "Reconfiguring Role-Based access control via role clustering," IEEE Access, vol. 13, pp. 72984–72993, Jan. 2025, doi: 10.1109/access.2025.3563057.

[18] P.GillandM. Tripunitara, "NewAlgorithmsforMinimizingthe NumberofEdgesin Bottom-Up Role Mining," Proceedings of the 30th ACM Symposium on Access Control Models and Technologies, pp. 22–32, Jul. 2025, doi: 10.1145/3734436.3734443.

[19] B. Mitra, S. Sural, J. Vaidya, and V. Atluri, "A survey of role mining," ACM Computing Surveys, vol. 48, no. 4, pp. 1–37, 2016, doi: 10.1145/2871148.

[20] N. A. Hamid, R. Ahmad, and S. R. Selamat, "A conceptual model of role based access control using role mining algorithm," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 12, no. 3, p. 1394, Dec. 2018, doi: 10.11591/ijeecs.v12.i3.pp1394-1400.

[21] M. Tripunitara, "Minimizing the Number of Roles in Bottom-Up Role-Mining using Maximal BicliqueEnumeration,"*Proceedingsofthe30thACMSymposiumonAccessControlModelsand Technologies*, pp. 4–15, Jul. 2025, doi: 10.1145/3734436.3734442.

[22] Z. Xu and S. D. Stoller, "Algorithms for mining meaningful roles," *Proceedings 17th ACM Symposium on Access Control Models and Technologies*, pp. 57–66, 2012, doi: 10.1145/2295136.2295146.

[23] M. Frank, J. M. Buhman, and D. Basin, "Role Mining with Probabilistic Models," ACM Transactions on Information and System Security, vol. 15, no. 4, pp. 1–28, Apr. 2013, doi: 10.1145/2445566.2445567.

[24] L. Kaufman and P. J. Rousseeuw, "Partitioning Around Medoids (Program PAM)," in Finding Groups in Data, John Wiley & Sons, pp. 68–125, 1990, doi: 10.1002/9780470316801.ch2.

[25] E. Schubert and P. J. Rousseeuw, "Fast and eager k -medoids clustering: $O(k)$ runtime improvement of the PAM, CLARA, and CLARANS algorithms," Information Systems, vol. 101, p. 101804, May 2021, doi: 10.1016/j.is.2021.101804.

[26] A. Ene, W. Horne, N. Milosavljevic, P. Rao, R. Schreiber, and R. E. Tarjan, "Fast exact and heuristic methods for role minimization problems," *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*, pp. 1–10, 2008.

AUTHORS

Nurul Syazwina Mohd Yusran is a postgraduate student at Universiti Sultan Zainal Abidin, Terengganu, Malaysia, pursuing a Master of Science in Computer Science. Her research interests include cybersecurity, access control, machine learning, data mining, and Role-Based Access Control (RBAC). Her current research focuses on role mining in RBAC using machine learning algorithms.

Nazirah Abd Hamid is a lecturer at Universiti Sultan Zainal Abidin, Terengganu, Malaysia. She received her BIT from Universiti Utara Malaysia (UUM), her M.Sc. from Universiti Teknologi Malaysia (UTM), and her Ph.D. from Universiti Teknikal Malaysia Melaka (UTeM). She specializes in biometric security systems and security services, including digital forensics, steganography, network security, public key infrastructure, and biometrics.

Norhidayah Muhammad received her Ph.D. from Universiti Teknologi Mara (UiTM),her M.Sc.from Universiti Malaysia Pahang (UMP), and her BIT from Universiti Malaysia Terengganu (UMT). She is a lecturer at Universiti Sultan Zainal Abidin, specializing in software project management, cryptography and cryptanalysis(includingkeymanagementandrandomnumbergeneration),data hiding and steganography, data security and digital security and privacy, and data encryption.