

Keywords: clustering, interpretability, scalability, categorical data, embeddings

Ashuza KUDERHA ^{1,2,*}, Olamma IHEANETU ¹

¹ Covenant University, Nigeria, kuderha.ashuzapgs@stu.cu.edu.ng, olamma.iheanetu@covenantuniversity.edu.ng

² Université Catholique de Bukavu, DR Congo

* Corresponding author: kuderha.ashuzapgs@stu.cu.edu.ng

Parallelogram-mode: A novel clustering method for categorical data

Abstract

The need for clustering methods that treat the interpretability of clustering results as valuable as cluster correctness is particularly important in high-risk domains. The heterogeneous nature of categorical data makes its clustering results less interpretable than those for numerical data. This challenge is amplified by the scarcity of works investigating the clustering of categorical data. Existing interpretable categorical data clustering algorithms do not optimise for scalability. In this paper, we address the scalability limitation by employing a vectorial approach to interpretable categorical data clustering that takes full advantage of the computational acceleration offered by modern computing hardware, such as GPUs. We formulate categorical data clustering as a rule-set search task that manipulates high-dimensional vector representations of entities in the input dataset. We propose an efficient and effective algorithm called “Coupling-Sorting-Branching” to solve the search task. This approach provides a scalable and fully interpretable method for clustering categorical data. Extensive experimental results across 14 real-world datasets demonstrate that our algorithm outperforms existing methods in cluster quality and explainability by producing IF-THEN rules.

1. INTRODUCTION

The increasing availability of large quantities of data accentuates the need to develop methods that can efficiently process it (Khoei & Singh, 2025). The processing tasks carried out on that data are generally categorised into two groups. In the first group, there are exploratory processing tasks that involve discerning the general structures of data without pre-specified models (Downey, 2025). In the second group, confirmatory processing tasks aim to validate existing models using the available data (Lange et al., 2025).

Clustering is a processing task in the exploratory data analysis category that segments input data into groups based on chosen similarity or dissimilarity metrics (T. Dinh et al., 2025).

There are four types of clustering techniques: partitional (Kutbay, 2018), hierarchical (Ran et al., 2023), model-based (Gormley et al., 2023), and density-based (Bhattacharjee & Mitra, 2021) techniques.

The methods to cluster numerical data often fail to be efficiently applied to categorical data as the latter does not have an inherent order between values. That said, specialised methods have been developed to solve clustering of categorical data (T. Dinh et al., 2025).

Researchers and practitioners across diverse domains employ clustering of categorical data for exploratory data analysis. Some of the domains in which categorical data clustering is used are healthcare (Kita et al., 2025), in precision agriculture (Kumar et al., 2023), and finance (Rao & Mandhala, 2025).

Beyond the quality of the clustering outputs, the explainability of the clustering process and its results is very important in high-stakes fields such as healthcare, precision agriculture, and finance. In these fields, trust in the technology is a function of users' ability to interpret the process and its outputs (Oliveira et al., 2025).

The current state-of-the-art method for categorical data clustering, developed to improve the interpretability of categorical data outputs, is the Significance-based Decision Tree (SigDT) method (Hu et al., 2025), which constructs a decision tree guided by a splitting process formalised as a multivariate two-sample testing problem. This method focuses on improving clusterability prediction, interpretable splitting, and cluster number prediction. Another method focusing on the interpretability of categorical data is Clustering using Unsupervised Binary Trees (CUBT) (Fraiman et al., 2013). This method has three steps: splitting the input data using binary trees to reduce its heterogeneity, pruning the binary trees, and joining similar clusters. The

CUBT method demonstrated the ability to cluster effectively across a wide range of cluster structures while maintaining high interpretability and efficiency (Fraiman et al., 2013).

Existing methods for interpretable categorical data clustering place less emphasis on computational scalability, as they were not designed with the primary goal of being highly scalable (Fraiman et al., 2013; Hu et al., 2025). This low level of scalability is their major limitation for usage in large-scale applications. The identified gap is the lack of interpretable categorical data clustering methods whose primary design goal is scalability. There is a need to develop methods that not only optimise for accuracy and interpretability but also offer an acceptable interpretability-scalability trade-off.

The solution we are proposing: (i) eliminates the reliance on distance metrics for categorical data by processing the categorical data in a high-dimensional continuous vector space using a stable and computationally efficient similarity measurement; (ii) efficiently selects optimal subsets of features for every cluster by employing an innovative feature subsets search approach; (iii) efficiently computes clustering of large scale data by employing a feature-set trimming technique; (iv) eliminates the need to initialize the clusters or to specify the number of clusters in advance by using a hands-off data-driven selection of the clusters; and (v) shows a high level of interpretability of categorical data clustering algorithm output by providing IF-THEN type of interpretation of the output as rules describing the cluster generators.

The contributions of this work can be summarised as an innovative method called “Parallelogram-Mode” that offers high interpretability of the data clustering outcome and scalable computation. This method builds cluster generators that are resultant vectors formed by summing attribute-pairs embeddings using the parallelogram technique. The proposed method leverages modern computational hardware to perform efficient, interpretable clustering of categorical data, as its processing steps can be performed in parallel on a GPU or a multi-core CPU.

The remaining parts of the paper are organised as follows: Section 2 reviews methods most relevant to our study. Section 3 presents a detailed description of our proposed method. Section 4 presents the benchmark results. At the end, Section 5 concludes the paper.

2. RELATED WORKS

Conventional clustering techniques frequently operate as "black boxes," maximising objective functions while concealing the variables that determine the groupings (Oliveira et al., 2025). On the other hand, interpretable clustering seeks clear, intelligible explanations for every cluster. Rule-based, prototype-based, and tree-structured algorithms that can explain clusters using logical rules or representative exemplars have been inspired by this for categorical data (Fraiman et al., 2013; Hu et al., 2025). In this section, important techniques are examined, emphasising their ability to handle categorical features and scale to large volumes of data.

1. Rule-based and tree-structured clustering methods

– Decision-tree clustering

Using the data, one branch of interpretable clustering creates a decision tree, with each leaf (cluster) specified by a combination of attribute-value tests along the way. Naturally, the rules on the branches explain these tree-based groups. For instance, SigDT is a decision-tree technique for categorical data (Hu et al., 2025). It statistically tests splits on categorical variables in order to generate a binary tree. When no split is statistically significant, SigDT stops. At each node, it chooses the split whose attribute-value distributions differ the most (as determined by χ^2 tests).

Tree-structured clustering methods emphasise transparency: split choices can be examined, and clusters are defined by explicit (or fuzzy) rules. An earlier example of these methods is Clustering Using Binary Trees (CUBT) (Fraiman et al., 2013). Tree-based clustering is generally more interpretable (users can follow paths to understand assignments), but it is typically less scalable due to the high cost of creating and analysing multiple candidate splits on large datasets.

– Rule learning with optimisation

Rule-based clustering methods optimise for explanations of concise rules. A popular method in this category formalises clustering as a mixed-integer optimisation that simultaneously allocates points to clusters and determines a Boolean rule for each cluster that maximises both rule accuracy/distinctiveness and cluster homogeneity (Carrizosa et al., 2023). In this method, clusters are specified by conjunctions of attribute-value criteria, thereby producing clusters with provably optimal rule-based descriptions that offer very high

interpretability. These methods do not scale well beyond intermediate datasets due to a computational trade-off: solving an MIP is costly (global optimisation is interleaved with clustering and rule-finding).

– Conceptual clustering

Conceptual clustering is a category of methods that gradually construct a probabilistic tree over categorical attributes. COBWEB (Fisher, 1987) is a conceptual clustering method in which the tree's nodes are “concepts” that summarise the attribute-value distribution of their objects. The COBWEB classification tree, despite being outdated, may be readily interpreted because each internal node represents a cluster that is characterised by its attribute-value probabilities. Although it has limited scalability (category utility calculations and hierarchical approaches develop quadratically), it is nevertheless a traditional interpretable method.

2. Scalable and parallel clustering methods

Recent research has focused on parallel and streaming frameworks for truly large-scale categorical data. In addition to CLOPE (Yang et al., 2002) and Squeezer (He et al., 2002), Limbo (Andritsos et al., 2004) summarises data using a height-balanced tree of distributional cluster features (DCF) and then applies an agglomerative information bottleneck to the summaries. Although it scales hierarchically, this two-stage method (summarise, then cluster summaries) is rather intricate. In the class of distributed clustering methods, the PUMA method (Pang et al., 2019) uses Hadoop to perform parallel subspace clustering by first identifying, then merging, sub-clusters on each node. Similarly, by breaking down the data into attribute subsets, MiCS-P (Li et al., 2022) uses Spark to compute mutual-information-based clusters in parallel. By dispersing computation, PUMA and MiCS-P both offer excellent scalability on large categorical datasets, but at the expense of interpretability (the algorithm's goal, not explicit rules, defines the resulting clusters).

3. METHODOLOGY

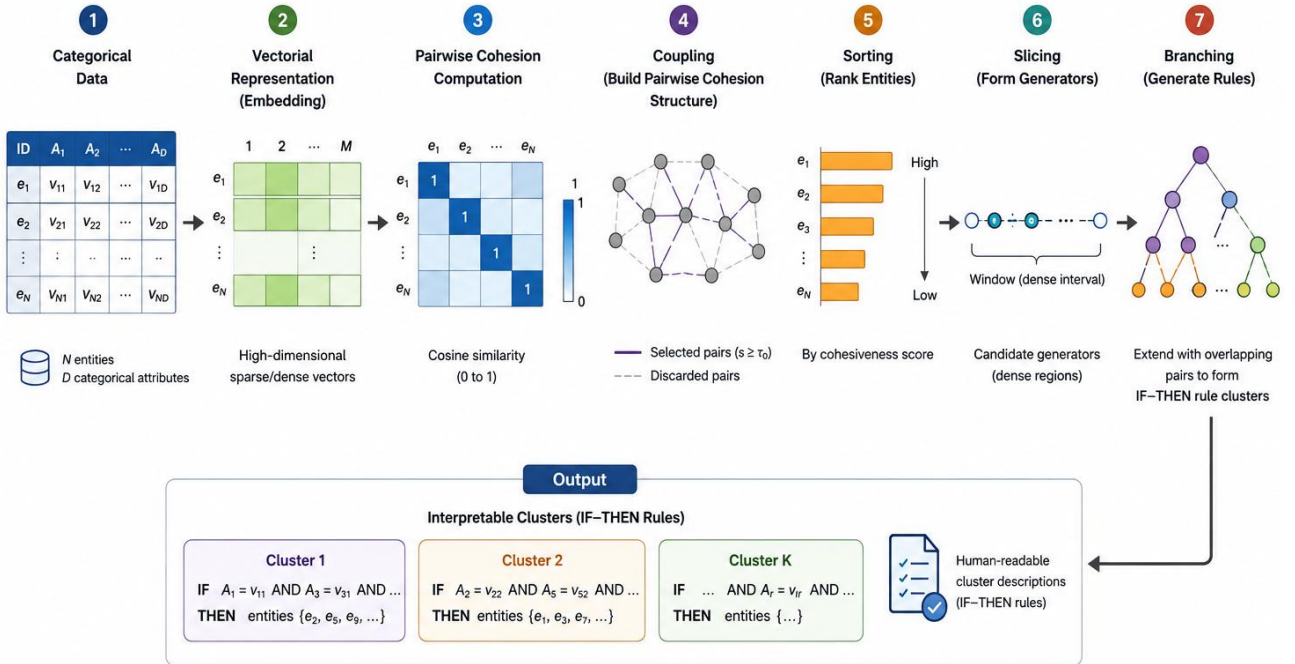


Fig. 1. Parallelogram-mode framework

3.1. Mathematical notations

Let the input dataset be

$$X = \{x^{(1)}, x^{(2)}, x^{(3)}, \dots, x^{(N)}\} \quad (1)$$

Where N is the number of observations. Each observation is represented by p categorical attributes:

$$x^{(i)} = (x_1^{(i)}, x_2^{(i)}, x_3^{(i)}, \dots, x_4^{(i)}), \quad i = 1, \dots, N \quad (2)$$

For each categorical attribute A_j , let

$$V_j = \{v_{j_1}, v_{j_2}, v_{j_3}, \dots, v_{j_{m_j}}\} \quad (3)$$

denote the set of observed modalities of A_j , where

$$m_j = |V_j| \quad (4)$$

The complete set of distinct attribute-value pairs is defined as

$$Q = \{(A_j, v): j = 1, \dots, p, v \in V_j\} \quad (5)$$

Let

$$M = |Q| = \sum_{j=1}^p m_j \quad (6)$$

denotes total number of different attribute-value pairs; an attribute-value pair $q = (A_j, v)$ denotes an interpretable categorical condition, such as “temperature=cold”. The proposed method describes a cluster by a generator that is a subset of attribute-value pairs that are mutually cohesive:

$$G = \{q_1, q_2, \dots, q_r\} \subseteq Q \quad (7)$$

The generator corresponds to the conjunction

$$G \equiv q_1 \wedge q_2 \wedge \dots \wedge q_r \quad (8)$$

Then, the generator can be written as an interpretable *IF – THEN* rule:

$$IF q_1 \wedge q_2 \wedge \dots \wedge q_r THEN cluster = C \quad (9)$$

In this work, generators are built in an unsupervised fashion. Reference class labels, if available, are not used for generator construction, but are kept for the subsequent external evaluation of the resulting clustering.

– Binary representation and co-occurrence

For each observation $x^{(i)}$, we define a binary indicator vector $z^{(i)} \in \{0,1\}^M$, where

$$z_q^{(i)} = \begin{cases} 1, & \text{if the attribute – value pair } q \text{ occurs in } x^{(i)} \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

The binary representation allows to directly compute the empirical co-occurrence (Leydesdorff & Vaughan, 2006; Sedighi, 2016) of attribute-value pairs. The empirical co-occurrence of two pairs $q_a, q_b \in Q$ is computed as:

$$C(q_a, q_b) = \frac{1}{N} \sum_{i=1}^N z_{q_a}^{(i)} z_{q_b}^{(i)} \quad (11)$$

The full co-occurrence matrix is

$$\mathbf{C} = [C(q_a, q_b)]_{a,b=1}^M \in \mathbb{R}^{M \times M} \quad (12)$$

The vectorial representation of an attribute-value pair q_a is defined as the corresponding row of the co-occurrence matrix:

$$\mathbf{e}_{q_a} = [C(q_a, q_1), C(q_a, q_2), C(q_a, q_3), \dots, C(q_a, q_M)]^T \in \mathbb{R}^M \quad (13)$$

Hence, the full embedding matrix is

$$\mathbf{E} = \begin{bmatrix} \mathbf{e}_{q_1}^\top \\ \mathbf{e}_{q_2}^\top \\ \dots \\ \mathbf{e}_{q_M}^\top \end{bmatrix} \in \mathbb{R}^{M \times M} \quad (14)$$

– Cohesion measurement

The cohesion of two attribute-value pairs is the cosine similarity (Lahitani et al., 2016) between their corresponding embeddings:

$$s(q_a, q_b) = \frac{\mathbf{e}_{q_a}^\top \mathbf{e}_{q_b}}{\|\mathbf{e}_{q_a}\|_2 \|\mathbf{e}_{q_b}\|_2} \quad (15)$$

Because the components of the co-occurrence embeddings are non-negative, the similarity satisfies $0 \leq s(q_a, q_b) \leq 1$.

CoSoBra algorithm uses the cohesion score to rank the candidate relations and to identify strongly associated attribute-value pairs.

3.2. Coupling-Sorting-Branching algorithm

The proposed Coupling-Sorting-Branching (CoSoBra) algorithm is designed to construct cluster generators from highly cohesive attribute-value pairs.

Let the set of candidate unordered pairs be:

$$P = \{(q_a, q_b) : q_a, q_b \in Q, a < b\} \quad (16)$$

The number of candidate pairs is:

$$K = |P| = \binom{M}{2} = \frac{M(M-1)}{2} \quad (17)$$

Step 1: Coupling

For each pair of candidates $\pi = (q_a, q_b) \in P$, the algorithm computes its cohesion score $s(\pi) = s(q_a, q_b)$. The coupling stage thus produces as output the scored candidate set:

$$L = \{(\pi_j, s(\pi_j)) : j = 1, 2, \dots, K\} \quad (18)$$

Step 2: Sorting

The candidate relations are ranked in descending order of cohesion:

$$s(\pi_1) \geq s(\pi_2) \geq \dots \geq s(\pi_K) \quad (19)$$

This study uses Quicksort to sort candidate pairs. This ordering favours the strongest categorical associations to emerge at the beginning of the candidate sequence, so that the branching operation preferentially builds generators based on highly cohesive relations.

Step 3: Slicing

The branching process may require several passes through the candidate list. To reduce this computation, CoSoBra algorithm uses an additional step, a slicing operation. Here, candidate relations with relatively low cohesion are discarded. This slicing step can be thought of as a fundamental sparsifying operation. It transforms the full graph of pairwise relations into a sparse subgraph. It helps curb the combinatorial explosion of paths that would be considered at the branching stage by pruning away weak associations early on.

Let $\bar{s} = \frac{1}{K} \sum_{j=1}^K s(\pi_j)$ be the mean cohesion.

Let $\alpha > 0$ be the slicing hyperparameter. The threshold is defined as:

$$T_\alpha = \alpha \bar{s} \quad (20)$$

The set of candidates kept is then:

$$P_\alpha = \{\pi_j \in P: s(\pi_j) \geq T_\alpha\} \quad (21)$$

Let $K_\alpha = |P_\alpha|$ and by construction $K_\alpha \leq K$. Therefore, the hyperparameter α determines the selectivity of the slicing operation. Increasing α decreases the number of relations that reach the branching phase, avoiding memory overload and greatly boosting computational efficiency, but possibly at the expense of ignoring useful high-order relations. Thus, the slicing mechanism provides an explicit, mathematically sound way to control the trade-off between computational scalability and clustering quality.

Step 4: Branching

The branching stage combines compatible candidate relations to produce cluster generators. Suppose that two retained relations are $\pi_a = (q_u, q_v)$ and $\pi_b = (q_v, q_w)$. Since the terminal attribute-value pair π_a is identical to the initial attribute-value pair of π_b , the two relations can be joined:

$$q_u \rightarrow q_v \rightarrow q_w \quad (22)$$

This yields the generator with the associated conditions $G = \{q_u, q_v, q_w\}$. More generally, a sequence of relations $(q_1, q_2), (q_2, q_3), \dots, (q_{r-1}, q_r) \in P_\alpha$ defines the generator

$$G = \{q_1, q_2, \dots, q_r\} \quad (23)$$

The generator may then be expressed as:

$$IF\ q_1 \wedge q_2 \wedge \dots \wedge q_r THEN\ cluster = C \quad (24)$$

Here, the cluster C is the observed partitioning of the data. It is not a label and it is not known a priori before applying the condition part of the generator that is $q_1 \wedge q_2 \wedge \dots \wedge q_r$.

So the branching operation is the step that turns the ranked, sparsified pairwise cohesion structure into explainable and human-readable cluster descriptions.

The Algorithm 1 presents the operational stages of the CoSoBra algorithm.

Algorithm 1 Coupling–Sorting–Branching (CoSoBra)

Require: Set of candidate unordered pairs $\mathcal{P}$, Embeddings $\mathbf{E}$, Slicing hyperparameter $\alpha > 0$

Ensure: Cluster generators formulated from categorical conditions

Stage 1: Coupling

- 1: **for** each candidate pair $\pi = (q_a, q_b) \in \mathcal{P}$ **do**
- 2: Compute cohesion score $s(\pi) \leftarrow s(q_a, q_b)$ via Eq. (15)
- 3: **end for**

Stage 2: Sorting

- 4: Sort $\mathcal{P}$ in descending order of $s(\pi)$ to yield sequence $\mathcal{L}$

Stage 3: Slicing

- 5: Compute mean cohesion $\bar{s} \leftarrow \frac{1}{K} \sum_{j=1}^K s(\pi_j)$
- 6: Set threshold $\tau_\alpha \leftarrow \alpha \bar{s}$
- 7: Retain sparse candidate set $\mathcal{P}_\alpha \leftarrow \{\pi \in \mathcal{L} \mid s(\pi) \geq \tau_\alpha\}$

Stage 4: Branching

- 8: **for** each relation chain in $\mathcal{P}_\alpha$ with overlapping terminal/initial pairs **do**
 - 9: Join relations (e.g., $q_u \rightarrow q_v$ and $q_v \rightarrow q_w \implies q_u \rightarrow q_v \rightarrow q_w$)
 - 10: Extract generator $G \leftarrow \{q_1, \dots, q_r\}$
 - 11: Formulate IF–THEN rule: IF $q_1 \wedge \dots \wedge q_r$ THEN cluster = C
 - 12: **end for**
-

3.3. Computational complexity analysis

The computational complexity of CoSoBra depends on the number of observations N , the number of categorical attributes p , and the number M of distinct attribute-value pairs.

It is important to note that the transformation of the raw categorical dataset into a binary (one-hot) design matrix $Z \in \{0,1\}^{N \times M}$ is considered a standard data preprocessing step. Therefore, the time and spatial footprints associated with it are removed from the core algorithm analysis.

1. Hardware-accelerated co-occurrence construction

Accumulating pairwise co-occurrences across active attributes formally requires $T_{cooccurrence} = O(Np^2)$. More important, as the co-occurrence matrix C can naturally be represented as a matrix multiplication of the preprocessed binary design matrix Z :

$$C = \frac{1}{N} Z^T Z \quad (25)$$

The iterative accumulation $O(Np^2)$ is rewritten as a General Matrix Multiply (GEMM) primitive that calls Basic Linear Algebra Subprograms (BLAS) Level-3 operations. These operations are explicitly designed to maximise spatial and temporal cache locality. For large datasets, frameworks such as PyTorch employ cuBLAS to distribute the block matrix multiplications across thousands of parallel GPU tensor cores. As BLAS operations consistently reach the theoretical peak Floating Point Operations Per Second (FLOPS) of the host hardware, the constant hidden inside the $O(Np^2)$ bound is very small, making the co-occurrence compilation computationally cheap.

2. Tensor-driven cohesion computation

In theory, for all candidate pairs, we need to compute cosine similarity between the dense M -dimensional embeddings with $T_{cohesion} = O(M^3)$.

Let $\tilde{E}$ be the row-wise *max-normalized* version of the embedding matrix E . The complete pairwise cohesion matrix S containing all scores $s(\pi)$ is computed algebraically via:

$$S = \tilde{E} \tilde{E}^T \quad (26)$$

While $O(M^3)$ is a prohibitive bound for scalar execution, matrix multiplication can be considered an embarrassingly parallel operation that scales very well on modern hardware architectures. For instance, consider a complex categorical dataset yielding $M = 10,000$ distinct modalities. The resulting $\tilde{E} \tilde{E}^T$ matrix multiplication takes approximately $2 * 10^{12}$ floating-point operations. A typical modern deep learning GPU can achieve tens to hundreds of TeraFLOPS ($> 10^{13}$ operations per second). Thus, the total spatial cohesion tensor for 10,000 different modalities can be computed in a fraction of a second. In practice the theoretical $O(M^3)$ bottleneck disappears completely.

3. Sorting operation

Sorting the K candidate relations requires $T_{sort} = O(K \log K)$ time on average (Iliopoulos, 2015). Since $K = O(M^2)$, the average-case complexity is $O(M^2 \log M)$.

4. Slicing and complexity reduction

The average cohesion score can be computed in a single pass over the candidate scores. Therefore, $T_{slice} = O(K) = O(M^2)$.

While the slicing operation itself does not reduce the upfront cost of cohesion or sorting computations, its fundamental complexity benefit lies in the downstream branching stage. The slicing operation reduces the amount of working memory needed for the relational subgraph by aggressively filtering the candidate set and enforces a strict upper bound on the number of edges to traverse, K_α .

5. Branching operation

Let K_α be the number of relations surviving slicing. The branching process repeatedly scans the retained relation list. For a direct implementation, the worst case number of comparisons is quadratic in K_α :

$$T_{branch} = O(K_\alpha^2) \quad (27)$$

This is where the computational argument for slicing becomes most clear. Without slicing ($\alpha = 0$), the algorithm operates on a fully dense graph where $K_\alpha = K = O(M^2)$, yielding an intractable $T_{branch, worst} = O(M^4)$ worst-case time.

However, a well-tuned slicing threshold T_α enforces deep sparsity so that $K_\alpha \ll K$. In practical regimes, slicing limits the active degree of each attribute-value node to a small constant. In this case K_α tends to $O(M)$ and collapses the effective branching time down to $O(M^2)$.

6. Memory complexity

The memory requirements of the CoSoBra algorithm are determined by the co-occurrence matrix, the set of all candidate relations, and the set of candidate relations that are kept. Since $K = O(M^2)$ and $K_\alpha \leq K$, the absolute worst case static memory complexity is limited by:

$$S(N, M) = O(M^2) \quad (28)$$

Slicing plays a critical role in limiting the amount of dynamic memory used at runtime. While the static matrices fundamentally require $O(M^2)$ storage, the active branching search space is limited entirely by $O(K_\alpha)$. Only keeping the sparse edges K_α in active memory during the recursive branching lookups avoids dynamic memory exhaustion and CPU cache trashing when M is massive.

The Table 1 summarizes the asymptotic computational and memory requirements of the core operational stages.

Tab. 1. Computational and memory requirements of core operational stages of the CoSoBra algorithm

Operation	Time complexity	Memory complexity
Co-occurrence (GEMM)	$O(Np^2)$	$O(M^2)$
Cohesion (GEMM)	$O(M^3)$	$O(M^2)$
Sorting	$O(K \log K) \text{ avg}$	$O(K)$
Slicing	$O(K)$	$O(K_\alpha)$
Branching	$O(K_\alpha^2)$	$O(K_\alpha)$
Overall	$O(Np^2 + M^3 + K_\alpha^2)$	$O(M^2)$

The complexity analysis shows that the CoSoBra algorithm requires an upfront cubic time investment in $O(M^3)$ to compute spatial cohesion. The combinatorial problem of the following $O(M^4)$ rule-generation phase is alleviated by the slicing parameter α that makes the search space sparse to $O(K_\alpha^2)$ (see Table 1). Moreover, the algorithmic complexity of the $O(Np^2)$ and $O(M^3)$ steps is independent of execution time, and their purely linear-algebraic structure allows modern computing ecosystems to execute them at near-peak hardware TFLOPS, making them effectively cheap in practical applications.

3.4. Benchmarking 1

The datasets used in this first benchmarking are presented in Table 2. The reference benchmark used here is from the work that introduced the SigDT method (Hu et al., 2025). They are from different domains, including health and medicine, social science, biology, business, agriculture, and games.

For this first benchmarking, the proposed Parallelogram-Mode method is compared against two state-of-the-art methods from SigDT’s benchmarking (Hu et al., 2025) for interpretable categorical data clustering: CUBT (Fraiman et al., 2013) and SigDT (Hu et al., 2025).

Two preprocessing steps are done prior to the clustering. Firstly, missing values are imputed using the mean for numerical features and the mode for categorical features. Secondly, the numerical features are transformed into categorical features using a binning-based discretisation technique (Hu et al., 2025).

Tab. 2. Datasets used in the first benchmarking

Dataset	Domain	#Features	#Samples	#Labels
Balance scale (Siegler, 1976)	Social Science	4	625	3
Breast cancer (Wolberg, 1990)	Health and Medicine	9	699	2
Car evaluation (Bohanec, 1988)	Business	6	1728	4
Hayes Roth (Hayes-Roth & Hayes-Roth, 1977)	Social Science	4	160	3
Heart disease (Janosi et al., 1989)	Health and Medicine	13	303	2
Congressional voting records (<i>Congressional Voting Records</i> , 1987)	Social Science	16	435	2
Lenses (<i>Lenses</i> , 1990)	Health	3	24	3
Lymphography (Zwitter & Soklic, 1988)	Health and Medicine	19	148	4
Primary tumor (Zwitter & Soklic, 1987)	Health and Medicine	17	339	21
Promoter sequences (Towell et al., 1993)	Biology	57	106	2
Soybean small (Michalski, 1980)	Agriculture	35	47	4
Tic-Tac-Toe (Aha, 1991)	Games	9	958	2
Zoo (Forsyth, 1990)	Biology	16	101	7

For this first benchmarking, we used two evaluation metrics: weighted F1-score and Purity. The weighted F1-score is useful when evaluating the performance of a classifier trained with an imbalanced dataset. It is the harmonic mean of Precision and Recall. It is computed as follows:

$$F1_{weighted} = \sum_{i=1}^K w_i * \left(2 * \frac{Precision_i * Recall_i}{Precision_i + Recall_i} \right) \quad (29)$$

Where the class weight is its support proportion:

$$w_i = \frac{n_i}{N}$$

Where:

- n_i : number of true samples belonging to class i .
- N : total number of samples.
- K : number of classes.

Purity is a performance evaluation metric for clustering that measures the extent to which a given cluster contains data points from a single class. It is computed as follows:

$$Purity = \frac{1}{N} \sum_{k=1}^K \max_{1 \leq j \leq J} |C_k \cap L_j| \quad (30)$$

Where:

- N : Total number of data points.
- K : Total number of clusters.
- J : Total number of unique classes.
- C_k : The set of points in cluster k .
- L_j : The set of points with the true class label j .
- $\max_j |C_k \cap L_j|$: The count of the dominant class in cluster k .

In this study, the one-sided Wilcoxon signed-rank test (Katakam, 2025) will be used as the benchmark to assess whether the proposed Parallelogram-Mode method significantly outperforms the state-of-the-art methods for Categorical data clustering, namely SigDT and CUBT.

The cluster identifiers obtained with the Parallelogram-Mode method, an unsupervised clustering algorithm, are arbitrary and do not correspond to the identifiers of the reference classes (unseen during clustering), so a post hoc label-assignment procedure was applied before calculating the performance evaluation metrics.

Let $\hat{c}_i \in \{1, 2, \dots, K\}$ be the cluster to which observation i belongs, and let $y_i \in \{1, 2, \dots, C\}$ be its reference class label. For each discovered cluster k , its associated set of observations is defined as

$$C_k = \{i: \hat{c}_i = k\} \quad (31)$$

The class label that is assigned to the cluster k is determined using a majority-voting scheme. In particular, the class selected was the most frequent class label among the observations belonging to that cluster:

$$l_k = c \in \{1, 2, \dots, C\} \operatorname{argmax} \sum_{i \in C_k} I(y_i = c) \quad (32)$$

Where $I(\cdot)$ is the indicator function and l_k is the class label of cluster k . The predicted class of observation i was then taken as the majority label of its discovered cluster:

$$\tilde{y}_i = l_{\hat{c}_i} \quad (33)$$

Therefore, all observations that are part of a specific discovered cluster are given the class label that is the most common among the reference labels in that cluster. This procedure enables a direct mapping between the clusters obtained by the unsupervised algorithm and the reference classes for external (or downstream) evaluation.

The Parallelogram-Mode method has only one hyperparameter α to be tuned. The extensive experiment informs us about the right way to search for the optimal value of α . This search can be guided by 2 principles:

- First: setting the initial value of α to 2. The extensive experiment we did in this work shows that setting the value of α to 2 gives an empty set S of attribute-value pairs' couples for the CoSoBra algorithm to work with. Starting with this empty set is the best option of a starting point when running the CoSoBra algorithm.
- Second: Decreasing the value of α while checking the running time and the number of discovered clusters.

Table 3 presents the different values of α for the datasets used in the first benchmarking task. These values of α were found via fine-tuning with respect to execution time, not Purity or f1-score. The user should select the value of α that gives the preferred execution time.

Tab.3. Optimal values of the hyperparameter

Dataset	α
Balance scale	0.99
Breast cancer	1
Car evaluation	0.99
Hayes Roth	0.99
Heart disease	1.2
Congressional voting records	1.2
Lenses	0.99
Lymphography	1.2
Primary tumor	1.1
Promoter sequences	1.1
Soybean small	1.11
Tic-Tac-Toe	1
Zoo	1

3.5. Benchmarking 2

Four datasets were used in this second benchmarking task. The benchmarking considered here is based on the recent review of categorical data clustering (T. Dinh et al., 2025). These data come from diverse domains, such as agriculture, politics, and biodiversity. Table 4 presents the datasets used.

Tab. 4. Datasets used in the second benchmarking

Dataset	Domain	#Features	#Samples	#Labels
Soybean (Michalski, 1980)	Agriculture	35	47	4
Zoo (Forsyth, 1990)	Biodiversity	16	101	7
Congressional voting records (<i>Congressional Voting Records</i> , 1987)	Politics	16	435	2
Mushroom (<i>Mushroom</i> , 1981)	Agriculture	22	8124	2

All the datasets presented in Table 4 have 100% categorical features. No preprocessing operation was done on those datasets (T. Dinh et al., 2025). This means that the missing values were not imputed. Leaving the missing values in the datasets will allow us to assess the Parallelogram-Mode's ability to handle them natively. During benchmarking, we will examine how missing values in the datasets affect the proposed method's performance.

The second benchmarking compares the proposed method against 14 existing methods for clustering categorical data (T. Dinh et al., 2025). The table 5 presents the 14 methods used in the comparative assessment.

For the second benchmarking, we used 4 performance evaluation metrics to assess the methods selected for comparison. These metrics are often used to evaluate performance on the task of clustering categorical data and were included in the recent review of categorical data clustering (T. Dinh et al., 2025). The table 6 presents the 4 performance evaluation metrics. We used the majority-voting method for cluster-to-class label assignment (see Section 3.4.D).

Tab. 5. Methods used in the comparative assessment

Method	Source
K-modes Huang	(Huang, 1997)
Fuzzy K-modes	(Huang & Ng, 1999)
ROCK	(Guha et al., 2000)
K-representatives	(Mar San et al., 2004)
K-modes Cao	(Cao et al., 2013)
Genetic Fuzzy K-modes	(Mukhopadhyay et al., 2009)
MGR	(Mukhopadhyay et al., 2009)
EGA FMC	(Narasimhan et al., 2018)
MFk-means	(Ben Salem et al., 2018)
K-means like	(Nguyen et al., 2023)
K-PbC	(D.-T. Dinh & Huynh, 2020)
LSH K-representatives	(Mau & Huynh, 2021)
GT-Kmodes	(Zhang & Chen, 2021)
MIS	(Sankhe et al., 2022)

The comparison between the proposed Parallelogram-Mode method and the existing methods in Table 5 will be based on the mean values of the 4 performance evaluation metrics in Table 6. This choice is motivated by the use of a few datasets in the selected benchmark, which makes computing the one-sided Wilcoxon signed-rank test unsuitable.

Tab. 6. Performance evaluation metrics used in the second benchmarking

Metric	Formulation	Meaning
Accuracy	$\frac{\sum_{k=1}^p a_k}{n}$	Measures how well the clustering matches the ground truth.
Purity	$\frac{1}{N} \sum_{k=1}^K \max_{1 \leq j \leq J} C_k \cap L_j $	Measures the extent to which a given cluster contains a single class.
Adjusted Rand Index (ARI)	$\frac{RI - E[RI]}{\max(RI) - E[RI]}$	Measures clustering similarity while accounting for random agreement.
Normalised Mutual Information (NMI)	$\frac{I(U; V)}{\sqrt{H(U)H(V)}}$	Measures the amount of information shared between two clusterings, normalised to account for the size of the clusters.

Table 7 presents the values of the hyperparameter α used in the second benchmarking. These values of α were found via finetuning.

Tab. 7. Optimal values of the hyperparameter for the second benchmarking were found through finetuning with respect to the execution time

Dataset	α
Soybean	1.11
Zoo	1
Congressional voting records	1.2
Mushroom	1.3

4. RESULTS AND DISCUSSION

4.1. Benchmarking 1

Table 8 presents the results of the first benchmarking. The results for CUBT and SigDT are taken from the paper introducing SigDT (Hu et al., 2025). CUBT (MI) (using Mutual Information) and CUBT (Ham) (using Hamming distance) are variants of the CUBT algorithm. The average values of the two selected performance evaluation metrics are also computed. The average values in bold indicate cases where the Parallelogram-Mode method outperforms SigDT and CUBT.

Tab. 8. Results of the first benchmarking

Dataset	CUBT (MI)			CUBT (Ham)			SigDT			Parallelogram-Mode (Ours)		
	f1-score	Purity	Avg.	f1-score	Purity	Avg.	f1-score	Purity	Avg.	f1-score	Purity	Avg.
Balance scale (Bs)	0.267	0.547	0.41	0.245	0.56	0.4	0.557	0.59	0.57	0.75	0.76	0.76
Breast cancer (Bc)	0.786	0.924	0.86	0.799	0.924	0.86	0.72	0.911	0.82	0.96	0.96	0.96
Car evaluation (Ce)	0.22	0.7	0.46	0.475	0.7	0.59	0.553	0.7	0.63	0.76	0.77	0.77
Hayes Roth (Hr)	0.35	0.5	0.43	0.33	0.5	0.42	0.436	0.485	0.46	0.61	0.61	0.61
Heart disease (Hd)	0.302	0.541	0.42	0.508	0.571	0.54	0.473	0.551	0.51	0.78	0.78	0.78
Congressional voting records (Hv)	0.658	0.894	0.78	0.749	0.926	0.84	0.569	0.95	0.76	0.92	0.92	0.92
Lenses (Ls)	0.443	0.625	0.53	-	-	-	0.498	0.625	0.56	0.88	0.88	0.88
Lymphography (Ly)	0.537	0.756	0.65	0.38	0.662	0.52	0.476	0.722	0.6	0.83	0.84	0.84
Primary tumor (Pt)	0.173	0.295	0.23	0.167	0.327	0.25	0.228	0.3	0.26	0.52	0.55	0.54
Promoter sequences (Ps)	0.593	0.802	0.7	-	-	-	0.584	0.802	0.69	0.84	0.84	0.84
Soybean small (So)	0.846	0.786	0.82	0.667	0.573	0.62	1	0.998	1	1	1	1
Tic-Tac-Toe (Tt)	0.384	0.678	0.53	0.298	0.678	0.49	0.42	0.721	0.57	0.66	0.7	0.68
Zoo (Zo)	0.916	0.831	0.87	0.795	0.732	0.76	0.686	0.802	0.74	0.96	0.96	0.96

The results of the one-sided Wilcoxon signed-rank test are presented in the table 9. We report the significance of the comparison between the proposed Parallelogram-Mode method and the CUBT and SigDT methods.

Tab. 9. Significance evaluation of the performance of the Parallelogram-Mode method against the existing methods for categorical data clustering

Methods vs Parallelogram-Mode	Metrics with p-value		
	f1-score	Purity	Average
CUBT (MI)	0.0001***	0.0001***	0.0001***
CUBT (Ham)	0.0005***	0.0010***	0.0005***
SigDT	0.0002***	0.0012**	0.0002***

From the results presented in the table 8 it can be seen that the Parallelogram-Mode method outperformed the state-of-the-art methods SigDT (Hu et al., 2025) and CUBT (Fraiman et al., 2013). This good performance of the proposed method is due to the fact that it produces generators that can generate denser clusters. This means that the proposed method tends to generate clusters for which members (or samples) satisfy the maximum number of antecedent conditions in the underlying rule. This finding supports the proposed method's capability to optimise for intra-cluster similarity while relaxing the constraint on the maximum number of clusters it can produce.

It can be seen in the table 9 that the superiority, with respect to performance, of the proposed Parallelogram-Mode method against the SigDT and CUBT methods is significant. We found that the Parallelogram-Mode method is significantly superior to the CUBT method (with the highest $p - value \leq 0.001$) and is significantly superior to the SigDT method (with the highest $p - value \leq 0.01$).

4.2. Benchmarking 2

The results of the second benchmarking are presented in tables 10-13. The values in bold indicate cases in which the proposed Parallelogram-Mode method outperformed other methods for categorical data clustering.

The performance metric values for existing methods are taken from the benchmarking reference (T. Dinh et al., 2025). The standard deviation values were not reported in the reference benchmarking.

Tab. 10. Results of the Soybean dataset for the second benchmarking.

Methods	Accuracy	Purity	ARI	NMI
K-modes Huang (1997)	98.21	98.21	95.52	96.42
Fuzzy K-modes (1999)	75.22	82.09	64.48	77.01
ROCK (2000)	82.69	82.69	67.16	82.09
K-representatives (2004)	86.87	90.45	83.28	90.45
K-modes Cao (2013)	99.7	99.7	99.7	99.7
Genetic Fuzzy K-modes (2009)	56.42	61.49	34.93	49.25
MGR (2009)	82.69	82.69	67.16	82.09
EGA FMC (2018)	84.78	88.36	74.03	82.99
MFK-means (2018)	85.37	88.06	77.61	86.57
K-means like (2023)	99.7	99.7	99.7	99.7
K-PbC (2020)	99.7	99.7	99.7	99.7
LSH K-representatives (2021)	99.7	99.7	99.7	99.7
GT-Kmodes (2021)	99.7	99.7	99.7	99.7
MIS (2022)	68.06	78.51	59.1	78.81
Parallelogram-Mode (ours)	100	100	100	100

Tab. 11. Results of Zoo dataset for the second benchmarking

Methods	Accuracy	Purity	ARI	NMI
K-modes Huang (1997)	72.3	89.5	65.6	79.9
Fuzzy K-modes (1999)	67.93	83.97	62.1	75.51
ROCK (2000)	41.69	45.77	2.92	12.54
K-representatives (2004)	72.3	86.88	66.76	79.59
K-modes Cao (2013)	67.35	88.34	59.18	76.39
Genetic Fuzzy K-modes (2009)	41.69	47.81	10.5	22.16
MGR (2009)	91.25	92.42	95.63	93
EGA FMC (2018)	76.09	77.26	75.8	75.51
MFK-means (2018)	77.26	85.13	67.06	77.84
K-means like (2023)	84.26	93.29	77.84	86.3
K-PbC (2020)	83.38	88.05	79.59	83.67
LSH K-representatives (2021)	77.25	89.21	68.22	79.88
GT-Kmodes (2021)	80.18	86.3	80.47	85.13
MIS (2022)	84.26	89.21	90.38	88.05
Parallelogram-Mode (ours)	96.03	96.03	96.57	93.42

Tab. 12. Results of Congressional voting records dataset for the second benchmarking

Methods	Accuracy	Purity	ARI	NMI
K-modes Huang (1997)	85.81	86.07	51.75	44.09
Fuzzy K-modes (1999)	86.07	86.07	52.28	44.62
ROCK (2000)	61.78	61.78	1.06	0.8
K-representatives (2004)	87.39	87.39	56.24	49.64
K-modes Cao (2013)	86.65	86.65	53.16	44.12
Genetic Fuzzy K-modes (2009)	81.06	81.06	39.07	33.22
MGR (2009)	82.92	82.92	43.06	40.13
EGA FMC (2018)	86.11	86.11	51.03	43.85
MFK-means (2018)	77.08	77.61	34.02	29.24
K-means like (2023)	87.71	87.71	56.61	50.23
K-PbC (2020)	87.71	87.71	56.61	50.76
LSH K-representatives (2021)	87.44	87.44	55.81	49.17
GT-Kmodes (2021)	86.65	86.65	53.16	44.12
MIS (2022)	82.92	82.92	43.06	40.13
Parallelogram-Mode (ours)	90.57	90.57	65.63	53.98

Tab. 13. Results of Mushroom dataset for the second benchmarking

Methods	Accuracy	Purity	ARI	NMI
K-modes Huang (1997)	88.13	88.13	61.38	56.92
Fuzzy K-modes (1999)	71.61	71.61	26.75	24.92
ROCK (2000)	79.21	79.21	44.32	40.131
K-representatives (2004)	87.08	87.08	57.71	56.66
K-modes Cao (2013)	51.93	53.51	4.72	7.08
Genetic Fuzzy K-modes (2009)	73.71	73.71	30.95	28.33
MGR (2009)	86.03	86.3	55.08	45.11
EGA FMC (2018)	88.13	88.13	60.85	54.56
MFk-means (2018)	88.39	88.39	61.12	55.34
K-means like (2023)	75.02	75.02	27.54	25.97
K-PbC (2020)	74.75	74.75	27.02	22.82
Parallelogram-Mode (ours)	88.77	88.77	60.13	49.3

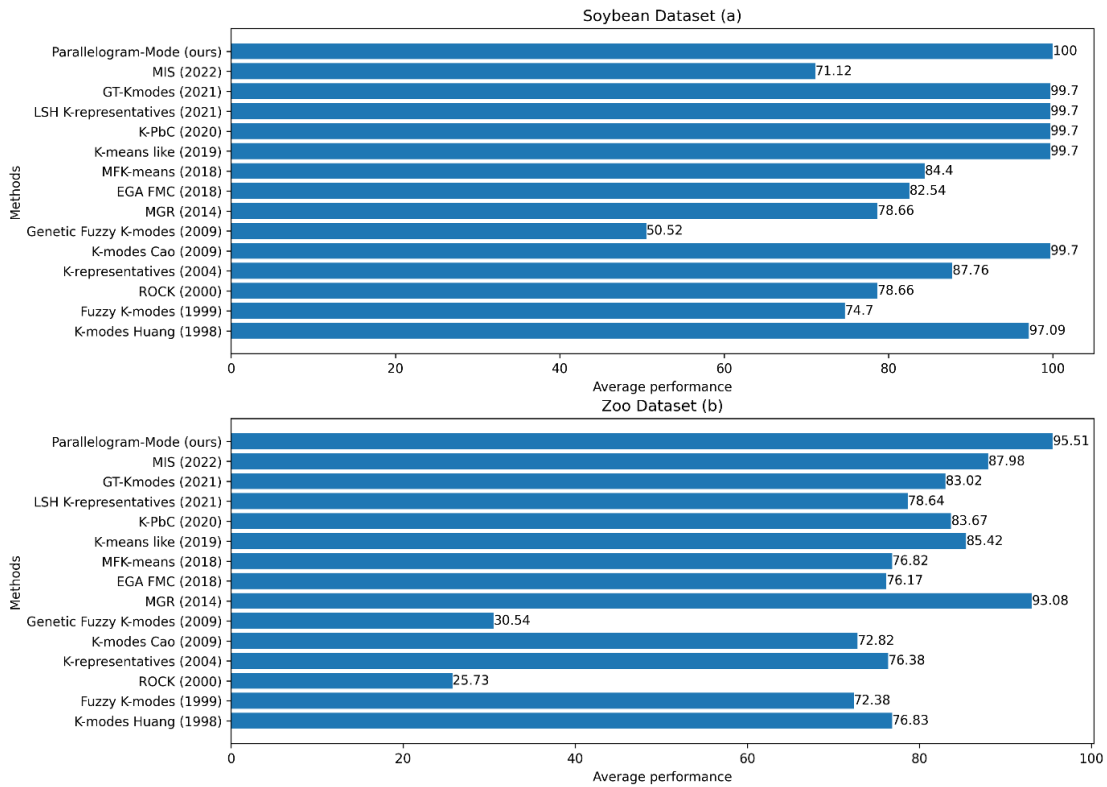


Fig. 2. Average performance on the datasets Soybean and Zoo for the second benchmarking

The average values of the selected four performance evaluation metrics (in tables 10-13) are presented in Figures 2 and 3. It can be seen that the Parallelogram-Mode method outperformed existing methods on Soybean (100 as the highest value) (see figure 2), Zoo (95.51 as the highest value) (see figure 2), and the Congressional voting records dataset (75.19 as the highest value) (see figure 3). On the other hand, the Parallelogram-Mode method shows relatively low performance compared to the existing methods on the Mushroom dataset (71.74, compared to 73.64 (the highest value)) (see figure 3). This low performance on the Mushroom dataset can be explained by the use of a large hyperparameter value α , which severely constrained the building of the generators. This finding implies that the Parallelogram-Mode method has a trade-off between scalability and clustering performance. The more the value of α is increased to improve scalability, the more the clustering performance is affected.

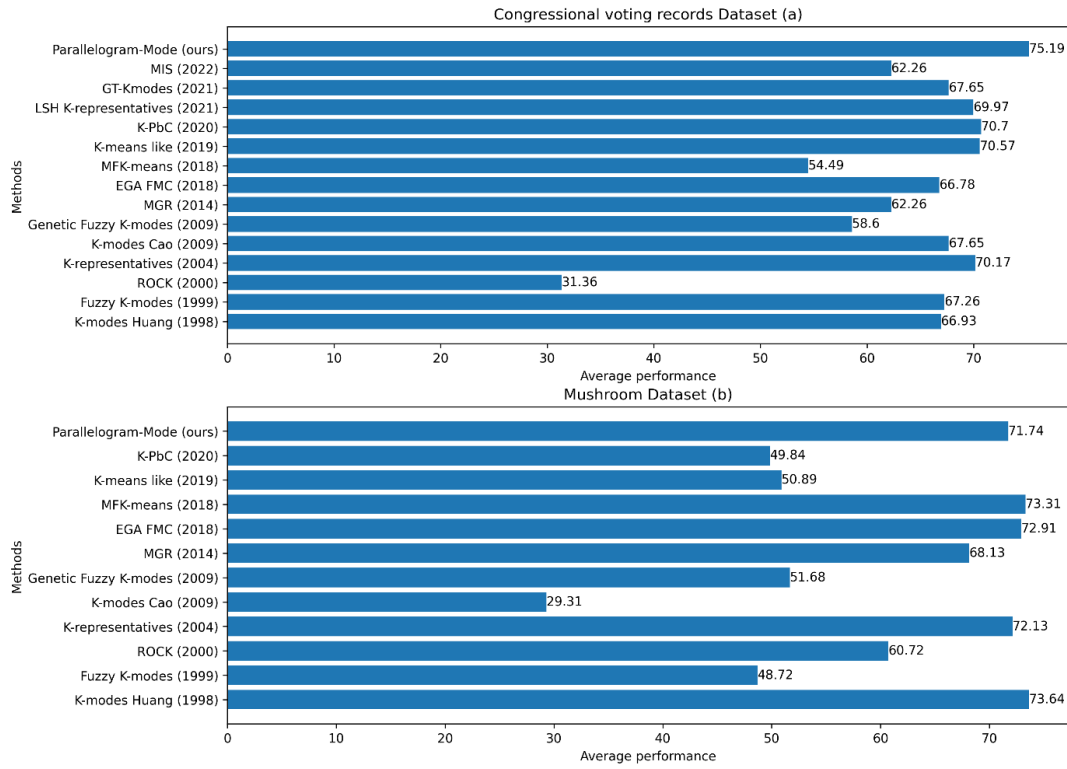


Fig. 3. Average performance on Congressional voting records and Mushroom datasets for the second benchmarking

This good performance of the Parallelogram-Mode method shows its capability to compete with categorical data clustering methods optimised for performance but lacking interpretability.

Regarding missing data, the experiment shows that the Parallelogram-Mode method handles them well. From the results presented in tables 8 and 12, it can be seen that the Parallelogram-Mode method applied to the Congressional voting record dataset gives competing clustering performance when trained with the missing values than when trained without the missing values (90.57 (with missing values) against 92 (without missing values)). This capability makes the Parallelogram-Mode method a good tool for clustering categorical data in context when missing-value imputation cannot be performed.

4.3. Scalability analysis

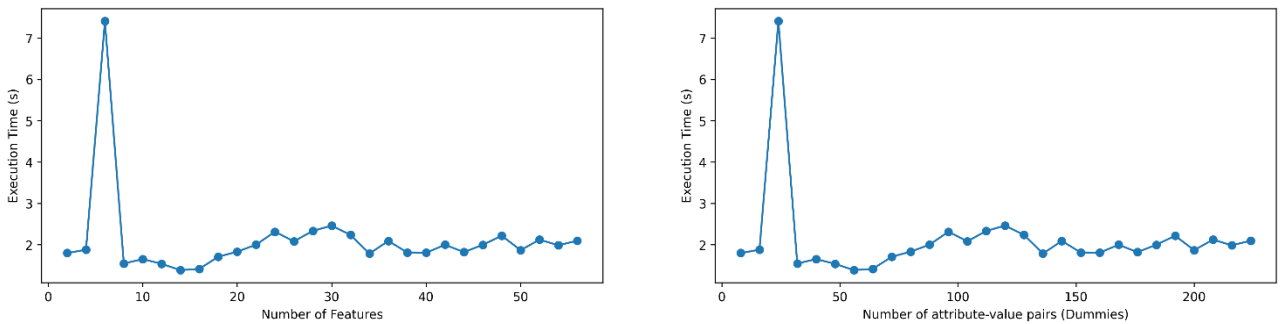


Fig. 4. Scalability analysis with respect to execution time (in seconds) using the Promoter Sequences dataset

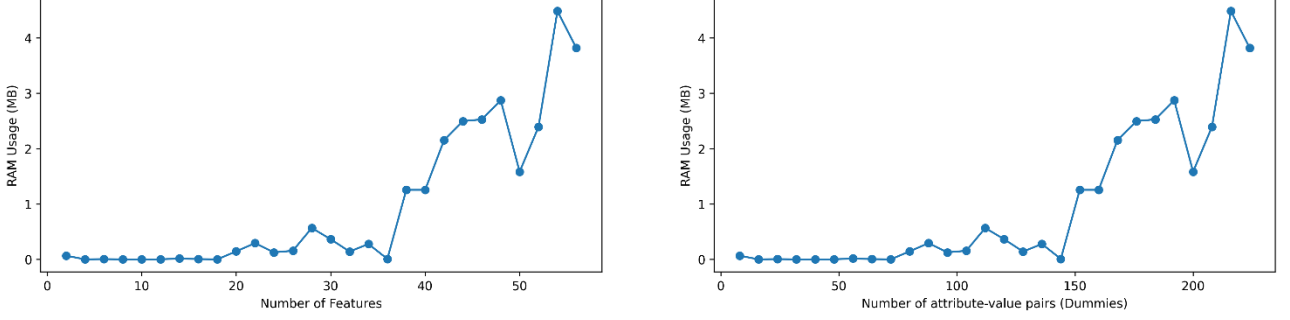


Fig. 5. Scalability analysis with respect to RAM usage (in MB) using the Promoter Sequences dataset

Figures 4 and 5 present the results of the scalability analysis done on the Promoter Sequences dataset (see Table 2). The experiment involved varying the number of features used to form the input dataset and measuring execution time (in seconds) and RAM usage (in MB) during CoSoBra's execution. This experiment was run on a Lenovo X1 Carbon 8th Gen with an i5 10th Gen CPU at 1.60 GHz and 16 GB of RAM.

As shown in Figure 4, the execution time varies almost linearly with the dataset size (number of features and number of attribute-value pairs). On the other hand, as shown in Figure 5, memory usage (RAM usage) varies linearly with dataset size (number of features and number of attribute-value pairs) for the first 36 features, then quadratically for 38 features and beyond.

These results prove (i) the high scalability of the proposed Parallelogram-Mode method with respect to execution time for a larger dataset and (ii) the relatively good scalability of the proposed Parallelogram-Mode method with respect to the memory usage for a medium dataset.

The unique feature that distinguishes Parallelogram-Mode from recent interpretable categorical clustering methods is the way the explanatory structure is discovered. CUBT constructs an unsupervised binary tree whose leaves are explained by paths of original-variable splits (Fraiman et al., 2013). SigDT constructs a categorical decision tree based on statistically significant two-sample tests, thus providing formal statistical justification for the splitting and stopping rules (Hu et al., 2025). The Parallelogram-Mode method does not build a hierarchy or rely on hypothesis testing; however, it describes attribute-value pairs by empirical co-occurrence vectors, ranks relations by cosine cohesion, slices to prune weak relations, and combines surviving relations into explicit IF-THEN generators. So, its unique contribution is a direct rule-search mechanism built around vectorised computation, not a new tree or statistical-testing framework.

Parallelogram-Mode is unlike optimisation- and embedding-based approaches. Methods such as jointly optimising cluster assignments and rule explanations through mixed-integer optimisation (Carrizosa et al., 2023) and graph-embedding approaches (Bai & Liang, 2022) learn numerical representations that are then used for clustering. Conversely, the vectors in Parallelogram-Mode are not latent features learned by a neural or graph-embedding objective; they are deterministic co-occurrence representations directly derived from the original categorical observations and are used only as a computational substrate for discovering symbolic relations. The final cluster description is returned to the original categorical domain as an interpretable generator. The novelty, therefore, lies in the direct interpretability of categorical rules combined with a scalable vectorised search architecture, capable of execution on GPUs or multi-core systems.

Another difference is that Parallelogram-Mode embeds interpretability into the clustering itself, rather than providing an explanation only at the end of an opaque partitioning process. This is consistent with the generator formulation in the manuscript, where the final rules are expressed directly in terms of observed attribute-value conditions. The approach thus complements approaches such as Mutual Information Scoring (Sankhe et al., 2022) that focus on improving the interpretability of cluster summaries by leveraging relational cohesion to help in the discovery of the clusters themselves.

5. CONCLUSIONS

In this paper, we introduced Parallelogram-Mode, a novel rule-based method developed to enhance scalability in interpretable categorical data clustering. Parallelogram-Mode offers many features that facilitate efficient computation of the clustering process while guaranteeing human understanding of the clustering process: (1) It enables fast computation of clusters by processing rules building tasks in the continual high-

dimensional vectorial space, taking advantage of modern computational hardware like GPU. (2) It accelerates the clustering process by trimming the feature set. (3) It produces cluster generators composed of If-THEN rules that are fully interpretable. (4) It provides flexibility in adjusting the scalability-accuracy trade-off through the finetuning of one hyperparameter. Two benchmarking experiments on 14 real-world datasets empirically demonstrate that the proposed Parallelogram-Mode method is competitive with the state-of-the-art and classical categorical-data clustering algorithms in terms of clustering quality.

The Parallelogram-Mode method faces a challenge: it lacks a feature that allows users to set the number of clusters in advance. Even if the hyperparameter α allows adjusting the number of discovered clusters, the lack of the ability to manually set the number of clusters at the start of an experiment remains a limitation to address. Another limitation to acknowledge is the lack of extensive evaluation of interpretability due to the difficulty of accessing the source code of existing methods. Future works can investigate these limitations and provide an improved version of this work.

The capabilities of Parallelogram-Mode make it a great tool in the hands of researchers and practitioners for whom the interpretability of categorical data clustering outcomes is as important as clustering accuracy. These researchers and practitioners who work in high-stakes fields such as healthcare, precision agriculture, and finance can use the proposed method to make categorical data clustering less of a black box.

Acknowledgements

We acknowledge the support of Covenant Applied Informatics and Communication Africa Centre of Excellence (CApIC-ACE) for providing the facility and sponsorship to complete this work.

Data Availability Statement

The data used in this study is publicly available. All data and materials supporting the results or analyses in this paper are under an open CC-BY licence, permitting reuse. The data used can be found in the open public repository <https://archive.ics.uci.edu>.

Conflicts of Interest

The authors declare no conflict of interest.

REFERENCES

- Aha, D. (1991). *Tic-Tac-Toe endgame* [Dataset]. UCI Machine Learning Repository. <https://doi.org/10.24432/C5688J>
- Andritsos, P., Tsaparas, P., Miller, R. J., & Sevcik, K. C. (2004). LIMBO: Scalable clustering of categorical data. In *Lecture Notes in Computer Science* (Vol. 2992, pp. 123–146). Springer. https://doi.org/10.1007/978-3-540-24741-8_9
- Bai, L., & Liang, J. (2022). A categorical data clustering framework on graph representation. *Pattern Recognition*, 128, Article 108694. <https://doi.org/10.1016/j.patcog.2022.108694>
- Ben Salem, S., Naouali, S., & Chtourou, Z. (2018). A fast and effective partitional clustering algorithm for large categorical datasets using a k-means based approach. *Computers & Electrical Engineering*, 68, 463–483. <https://doi.org/10.1016/j.compeleceng.2018.04.023>
- Bhattacharjee, P., & Mitra, P. (2021). A survey of density based clustering algorithms. *Frontiers of Computer Science*, 15(1), Article 151308. <https://doi.org/10.1007/s11704-019-9059-3>
- Bohanec, M. (1988). *Car evaluation* [Dataset]. UCI Machine Learning Repository. <https://doi.org/10.24432/C5JP48>
- Cao, F., Liang, J., Li, D., & Zhao, X. (2013). A weighting k-modes algorithm for subspace clustering of categorical data. *Neurocomputing*, 108, 23–30. <https://doi.org/10.1016/j.neucom.2012.11.009>
- Carrizosa, E., Kurishchenko, K., Marín, A., & Romero Morales, D. (2023). On clustering and interpreting with rules by means of mathematical optimization. *Computers & Operations Research*, 154, Article 106180. <https://doi.org/10.1016/j.cor.2023.106180>
- Congressional voting records*. (1987). [Dataset]. UCI Machine Learning Repository. <https://doi.org/10.24432/C5C01P>
- Dinh, D.-T., & Huynh, V.-N. (2020). k-PbC: An improved cluster center initialization for categorical data clustering. *Applied Intelligence*, 50(8), 2610–2632. <https://doi.org/10.1007/s10489-020-01677-5>
- Dinh, T., Wong, H., Fournier-Viger, P., Lisik, D., Ha, M. Q., Dam, H. C., & Huynh, V. N. (2025). Categorical data clustering: 25 years beyond K-modes. *Expert Systems with Applications*, 272, Article 126608. <https://doi.org/10.1016/j.eswa.2025.126608>
- Downey, A. (2025). *Think stats: Exploratory data analysis*. O'Reilly Media.
- Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2), 139–172. <https://doi.org/10.1023/A:1022852608280>

- Forsyth, R. (1990). *Zoo* [Dataset]. UCI Machine Learning Repository. <https://doi.org/10.24432/C5R59V>
- Fraiman, R., Ghattas, B., & Svarc, M. (2013). Interpretable clustering using unsupervised binary trees. *Advances in Data Analysis and Classification*, 7(2), 125–145. <https://doi.org/10.1007/s11634-013-0129-3>
- Gormley, I. C., Murphy, T. B., & Raftery, A. E. (2023). Model-based clustering. *Annual Review of Statistics and Its Application*, 10(1), 573–595. <https://doi.org/10.1146/annurev-statistics-033121-115326>
- Guha, S., Rastogi, R., & Shim, K. (2000). Rock: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5), 345–366. [https://doi.org/10.1016/S0306-4379\(00\)00022-3](https://doi.org/10.1016/S0306-4379(00)00022-3)
- Hayes-Roth, B., & Hayes-Roth, F. (1977). *Hayes-Roth* [Dataset]. UCI Machine Learning Repository. <https://doi.org/10.24432/C5501T>
- He, Z., Xu, X., & Deng, S. (2002). Squeezer: An efficient algorithm for clustering categorical data. *Journal of Computer Science and Technology*, 17(5), 611–624. <https://doi.org/10.1007/BF02948829>
- Hu, L., Jiang, M., Liu, X., & He, Z. (2025). Significance-based decision tree for interpretable categorical data clustering. *Information Sciences*, 690, Article 121588. <https://doi.org/10.1016/j.ins.2024.121588>
- Huang, Z. (1997). A fast clustering algorithm to cluster very large categorical data sets in data mining. *Research Issues on Data Mining and Knowledge Discovery*, 1–8.
- Huang, Z., & Ng, M. K. (1999). A fuzzy k-modes algorithm for clustering categorical data. *IEEE Transactions on Fuzzy Systems*, 7(4), 446–452. <https://doi.org/10.1109/91.784206>
- Iliopoulos, V. (2015). *The Quicksort algorithm and related topics*. arXiv. <https://doi.org/10.48550/arXiv.1503.02504>
- Janosi, A., Steinbrunn, W., Pfisterer, M., & Detrano, R. (1989). *Heart disease* [Dataset]. UCI Machine Learning Repository. <https://doi.org/10.24432/C52P4X>
- Katakam, A. (2025). Mann-Whitney test, and Wilcoxon. *Translational Gastroenterology*, 135.
- Khoei, T. T., & Singh, A. (2025). Data reduction in big data: A survey of methods, challenges and future directions. *International Journal of Data Science and Analytics*, 20(3), 1643–1682. <https://doi.org/10.1007/s41060-024-00603-z>
- Kita, F. J., Rao, G. S., & Kirigiti, P. J. (2025). Optimizing clustering of electronic health tabular data: Generative adversarial networks and Dirichlet process mixture models for advance healthcare analytics. *IIEE Transactions on Healthcare Systems Engineering*, 15(3), 224–240. <https://doi.org/10.1080/24725579.2025.2510966>
- Kumar, K. R. N., Lahza, H., Sreenivasa, B. R., Shawly, T., Alsheikhy, A. A., Arunkumar, H., & Nirmala, C. R. (2023). A novel cluster analysis-based crop dataset recommendation method in precision farming. *Computer Systems Science and Engineering*, 46(3), 3239–3260. <https://doi.org/10.32604/csse.2023.036629>
- Kutbay, U. (2018). Partitional clustering. In *Recent Applications in Data Clustering*. IntechOpen. <https://doi.org/10.5772/intechopen.75836>
- Lahitani, A. R., Permanasari, A. E., & Setiawan, N. A. (2016). Cosine similarity to determine similarity measure: Study case in online essay assessment. In *Proceedings of 2016 4th International Conference on Cyber and IT Service Management (CITSM 2016)* (pp. 1–6). IEEE. <https://doi.org/10.1109/CITSM.2016.7577578>
- Lange, F. J. D., Wilcke, J. C., Hoffmann, S., Herrmann, M., & Boulesteix, A. L. (2025). On “confirmatory” methodological research in statistics and related fields. *Statistics in Medicine*, 44(25–27). <https://doi.org/10.1002/sim.70303>
- Lenses. (1990). [Dataset]. UCI Machine Learning Repository. <https://doi.org/10.24432/C5K88Z>
- Leydesdorff, L., & Vaughan, L. (2006). Co-occurrence matrices and their applications in information science: Extending ACA to the web environment. *Journal of the American Society for Information Science and Technology*, 57(12), 1616–1628. <https://doi.org/10.1002/asi.20335>
- Li, J., Zhang, C., Zhang, J., Qin, X., & Hu, L. (2022). MiCS-P: Parallel mutual-information computation of big categorical data on Spark. *Journal of Parallel and Distributed Computing*, 161, 118–129. <https://doi.org/10.1016/j.jpdc.2021.12.002>
- Mar San, O., Huynh, V.-N., & Nakamori, Y. (2004). An alternative extension of the k-means algorithm for clustering categorical data. *International Journal of Applied Mathematics and Computer Science*, 14(2), 241–247.
- Mau, T. N., & Huynh, V. N. (2021). An LSH-based k-representatives clustering method for large categorical data. *Neurocomputing*, 463, 29–44. <https://doi.org/10.1016/j.neucom.2021.08.050>
- Michalski, R. (1980). *Soybean (small)* [Dataset]. UCI Machine Learning Repository. <https://doi.org/10.24432/C5DS3P>
- Mukhopadhyay, A., Maulik, U., & Bandyopadhyay, S. (2009). Multiobjective genetic algorithm-based fuzzy clustering of categorical attributes. *IEEE Transactions on Evolutionary Computation*, 13(5), 991–1005. <https://doi.org/10.1109/TEVC.2009.2012163>
- Mushroom. (1981). [Dataset]. UCI Machine Learning Repository. <https://doi.org/10.24432/C5959T>
- Narasimhan, M., Balasubramanian, B., Kumar, S. D., & Patil, N. (2018). EGA-FMC: Enhanced genetic algorithm-based fuzzy k-modes clustering for categorical data. *International Journal of Bio-Inspired Computation*, 11(4), 219–228. <https://doi.org/10.1504/IJBIC.2018.092801>
- Nguyen, T. H. T., Dinh, D. T., Sriboonchitta, S., & Huynh, V. N. (2023). A method for k-means-like clustering of categorical data. *Journal of Ambient Intelligence and Humanized Computing*, 14(11), 15011–15021. <https://doi.org/10.1007/s12652-019-01445-5>
- Oliveira, G. S. de, Silva, F. A., & Ferreira, R. V. (2025). Explainable clustering: A solution to interpret and describe clusters. *Journal of Information and Data Management*, 16(1), 170–180. <https://doi.org/10.5753/jidm.2025.4663>
- Pang, N., Zhang, J., Zhang, C., Qin, X., & Cai, J. (2019). PUMA: Parallel subspace clustering of categorical data using multi-attribute weights. *Expert Systems with Applications*, 126, 233–245. <https://doi.org/10.1016/j.eswa.2019.02.030>
- Ran, X., Xi, Y., Lu, Y., Wang, X., & Lu, Z. (2023). Comprehensive survey on hierarchical clustering algorithms and the recent developments. *Artificial Intelligence Review*, 56(8), 8219–8264. <https://doi.org/10.1007/s10462-022-10366-3>
- Rao, R. K., & Mandhala, V. N. (2025). Comparative analysis of clustering algorithms for financial fraud detection. *International Journal of Safety and Security Engineering*, 15(4), 777–785. <https://doi.org/10.18280/ijss.150414>
- Sankhe, P., Hall, S. F., Sage, M., Rodriguez, M. Y., Chandola, V., & Joseph, K. (2022). Mutual information scoring: Increasing interpretability in categorical clustering tasks with applications to child welfare data. In *Lecture Notes in Computer Science* (Vol. 13558, pp. 165–175). Springer. https://doi.org/10.1007/978-3-031-17114-7_16

- Sedighi, M. (2016). Application of word co-occurrence analysis method in mapping of the scientific fields (case study: The field of Informetrics). *Library Review*, 65(1–2), 52–64. <https://doi.org/10.1108/LR-07-2015-0075>
- Siegler, R. (1976). *Balance scale* [Dataset]. UCI Machine Learning Repository. <https://doi.org/10.24432/C5488X>
- Towell, G. G., Shavlik, J. W., & Noordewier, M. O. (1993). *Molecular biology (promoter gene sequences)* [Dataset]. UCI Machine Learning Repository. <https://doi.org/10.24432/C5S01D>
- Wolberg, W. (1990). *Breast cancer Wisconsin (original)* [Dataset]. UCI Machine Learning Repository. <https://doi.org/10.24432/C5HP4Z>
- Yang, Y., Guan, X., & You, J. (2002). CLOPE: A fast clustering algorithm for transactional data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 682–687). ACM. <https://doi.org/10.1145/775047.775149>
- Zhang, Z., & Chen, S.-M. (2021). Optimization-based group decision making using interval-valued intuitionistic fuzzy preference relations. *Information Sciences*, 561, 352–370. <https://doi.org/10.1016/j.ins.2020.12.047>
- Zwitter, M., & Soklic, M. (1987). *Primary tumor* [Dataset]. UCI Machine Learning Repository. <https://doi.org/10.24432/C5WK5Q>
- Zwitter, M., & Soklic, M. (1988). *Lymphography* [Dataset]. UCI Machine Learning Repository. <https://doi.org/10.24432/C54598>