

HYBRID SORT – A PATTERN-FOCUSED MATRIX REORDERING APPROACH BASED ON CLASSIFICATION

Celmar Guimarães da Silva

School of Technology – University of Campinas, Brazil

ABSTRACT

Matrix reordering algorithms permute rows and columns of an input matrix for unveiling patterns that are hidden due to inappropriate permutations. Most reordering algorithms are not pattern-focused and provide low-quality results for revealing some particular patterns. On the other side, few algorithms focus on revealing specific patterns with high-quality results, but they must know a priori that one of these patterns is hidden in the matrix. In this work, we define a matrix reordering algorithm that focuses on revealing a pattern from a pattern set while providing high-quality results. Based on permutation-invariant features of matrices, the algorithm Hybrid Sort uses an empirically-created classifier to classify an input matrix into a canonical data pattern. After that, the algorithm uses this classification to select a pattern-focused reordering algorithm to reorder the input matrix. Therefore, Hybrid Sort inherits the good output quality of the selected reordering algorithms. We report that the classifier reached F2-measures greater or equal to 85% to all patterns in our tests. Besides, we present some examples of this new method applied to synthetic and real-world matrices.

KEYWORDS

Reorderable Matrix, Heatmaps, Pattern Detection

1. INTRODUCTION

Visualization techniques aim to help users to amplify cognition (Card et al. 1999) to better understand datasets. Laying out marks appropriately is a way of reaching this goal, given that proper layout may unveil patterns, tendencies, and outliers in the dataset. Providing a good layout in matrix visualizations (such as heatmaps) means to find row and column permutations that show patterns (if they exist) in the matrix to the user. These permutations can help to perform tasks such as: defining the relative chronological order among a set of artifacts in archaeology according to their similarities, better presenting the results of sociometric tests and other social network data, displaying clusters or gradual changes in the composition of species (Liiv, 2010), and analyzing similarities between students and subjects in undergraduate courses (Barros et al., 2019).

The visualization literature presents a set of patterns that may be found in matrices and that are useful to enhance user understanding about the underlying dataset (Behrisch et al. 2016, Wilkinson 2005). Despite the existence of these patterns, most matrix reordering algorithms, such as the ones surveyed by Behrisch et al. (2016) and Liiv (2010), do not focus on discovering particular patterns in the input matrix. We found only a few pattern-focused reordering algorithms. Most of them try to discover only one or two particular patterns in a matrix (Rocha et al. 2017, Santos et al. 2016, Silva 2019, Silva et al. 2017). Also, we did not find any algorithm that combines reordering algorithms to focus on a particular set of patterns. As an exception, the SMB method (Medina et al. 2016) tries to unveil four patterns, but it does not combine a set of reordering algorithms.

In this paper, we demonstrate that it is possible to construct a hybrid reordering algorithm. Given an $n \times m$ input matrix whose cells have real values, our reordering algorithm (*Hybrid Sort*) combines a matrix classifier and some pattern-focused reordering algorithms to reveal canonical data patterns that may be hidden in this input matrix. Inspired on Medina et al. (2016), we created our own empirical classifier and selected three pattern-focused algorithms that are able to unveil all canonical data patterns. We connected properly the classifier and the selected algorithms. In this paper, we show that our classifier surpasses SMB's

one and also works with one more pattern. We also present how Hybrid Sort reorders some synthetic and real-world matrices.

This paper is organized as follows. Section 2 briefly presents some concepts regarding matrix patterns. Section 3 discusses works related to pattern-focused reordering algorithms. Section 4 details our hybrid algorithm, with focus on the construction of the matrix classifier. Section 5 exemplifies how our algorithm reorders synthetic and real-world matrices. Besides, it presents a complexity analysis of the algorithm and briefly discusses limitations of this research. Section 6 concludes the paper and points out future work.

2. THEORETICAL BACKGROUND

Behrisch et al. (2016) and Wilkinson (2005) defined nine relevant patterns that may be present in some permutation of a matrix. In a broad analysis, these patterns help users to overview a dataset, its topology, its partitioning and grouping, and the existence of connections among groups. They also aid to inspect adjacencies and connectivity in graphs. This work considered five patterns, known as *canonical data patterns* (Wilkinson 2005): Simplex, Equi, Band, Circumplex, and Block (Figure 1(a-e)). A *Simplex* pattern happens if cell values tend to increase from a matrix corner to its opposite corner. An *Equi-correlation* (a.k.a. *Equi*) pattern happens if cell values tend to increase in bottom-up or top-down directions. The *Band* pattern has its highest values in the main diagonal, and the values decrease from this diagonal to the corners of the secondary diagonal. The *Circumplex* pattern is similar to a Band pattern, but the matrix is taken as a torus, and the band has no start or end. The layout of a *Block* pattern is similar to a heatmap of a sequence of binary numbers with k digits, where each digit defines the values of a rectangular part of the matrix. The literature indicates other patterns that are more related to matrix-based representations of graphs (*Blocks*, *Off-diagonal block*, *Line/Star*, and *Bands*) (Figure 1(f-i)), or anti-patterns (*Bandwidth* and *Noise*) – matrices from which users cannot extract useful information (Behrisch et al. 2016, 2020).

A useful terminology about patterns is the “pre-pattern” terminology. As far as we know, its first use is in the work of Wilkinson (2005). Given a pattern P , a “pre- P ” matrix is a matrix that can be permuted to reveal the pattern P . E.g., a pre-Band matrix can be properly permuted to show a Band pattern.

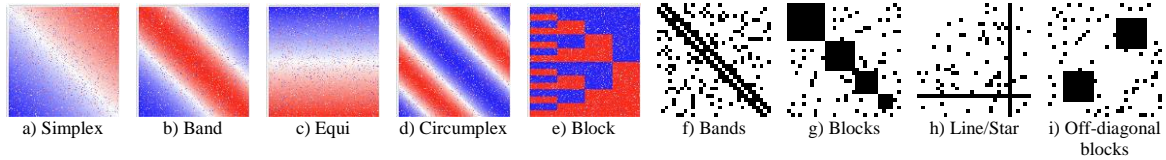


Figure 1. Examples of matrix patterns. (a) to (e): Wilkinson’s canonical data patterns; red, white and blue cells stand for high, intermediate and low values, respectively. (f) to (i): Binary patterns by Behrisch et al. (2017); black and white stand for 1 and 0, respectively. Source of the binary patterns: <http://magnostics.dbvis.de/#/benchmark>

3. RELATED WORK

In this section, we highlight a set of reordering algorithms that may be classified as *pattern-focused* algorithms. Each of them tries to find permutations of rows and columns to reveal a particular set of matrix patterns.

FVS (Silva et al. 2017) reorders the rows (and columns) of a matrix according to their mean values. This simple but useful approach can reveal Simplex and Equi patterns with time complexity $O(n^2)$ (for an $n \times n$ matrix). *Circumplex Sort* (Rocha et al. 2017) sorts the rows of a pre-Circumplex matrix according to index-weighted means of the values of each column. After that, the algorithm creates two stacks of rows. At each iteration, a row r is added to the stack whose top row is most similar to r . At the end, the algorithm concatenates both stacks. This process is repeated for columns. The method unveils a Circumplex pattern with time complexity $O(n^2)$. For pre-Band and pre-Circumplex input matrices, *Polar Sort* (Silva 2019) calculates two multidimensional projections – one for rows, and other for columns. In each projection, a barycenter of the projected points (representing rows or columns) is used to define a circular ordering of

these points. This ordering is then replicated into the matrix, unveiling Band and Circumplex patterns with time complexity $O(n^3)$. The *Block Reordering* method (Santos et al. 2016) receives a pre-Block matrix as input and returns a matrix with a Block pattern. The general strategy of this algorithm is (a) to choose iteratively a pivot column (i.e., a column with the lowest noise level of the set of columns considered in a single iteration), (b) to group columns that are similar to it, and (c) to order this groups. The order of the rows is defined by sorting each column of a set of representative columns of each group. The time complexity of the method is $O(kn^2)$, where k is a configuration parameter of the Block pattern.

SMB and MAGNOSTICS are the most similar works to our proposal. SMB (Medina et al. 2016) uses multiple binarized versions of an input matrix to calculate an appropriate permutation of rows and columns. It defines binarization thresholds according to the matrix pattern that underlies the input matrix. Finding what is this pattern is a relevant point of this algorithm. It calculates linear regressions of sorted vectors with maximum and minimum values of rows and columns of the input matrix. The linear coefficients of these regressions form feature vectors that are used to classify the underlying canonical data pattern of the matrix (except Block pattern). Our classifier and feature vectors are inspired by SMB's ones. Our feature vectors have other coefficients than SMB, which enhanced our classification results. Besides, our classifier can detect the Block pattern, which was not the focus of SMB.

The authors of MAGNOSTICS (Behrisch et al. 2017) selected six feature vectors/descriptors (from a set of 31 ones) that can help to classify a given input matrix into one of the following matrix patterns/anti-patterns: Blocks, Off-Diagonal Blocks, Star/Lines, and Bands patterns; and Bandwidth and Noise anti-patterns (Behrisch et al., 2016). However, their classifier was not used to choose an algorithm to reorder a matrix, as we propose in the current paper. Besides, it does not deal with canonical data patterns.

4. HYBRID SORT

Hybrid Sort is based on pattern-focused reordering methods and a matrix classifier. It has three main steps: (1) classify the input matrix M into a pattern p ; (2) select a reordering algorithm r according to p ; (3) use r to reorder M . The following subsections discuss the construction of the classifier of Step 1 and the algorithm selector of Step 2.

4.1 Permutation-Invariant Features of Matrices

The proposed matrix classifier aims to classify a matrix into one of the canonical data patterns. More specifically, we defined our pattern set as: pre-Band, pre-Simplex, pre-Circumplex, pre-Equi, and pre-Block. The prefix "pre-" in the pattern names reinforces that the input matrix probably does not show the pattern in its current permutation. Therefore, a classifier must be based in permutation-invariant features, i.e. matrix properties that do not depend on positions of columns and rows.

Based on the work of Medina et al. (2016), we observed that we could define four vectors for each matrix $M_{m \times n}$: $minRows[1..m]$ and $maxRows[1..m]$ have, respectively, the minimum and maximum value of each row of M ; and $minColumns[1..n]$ and $maxColumns[1..n]$ have, respectively, the minimum and maximum value of each column of M . Given that these vectors are dependent of the permutation order of rows and columns, we removed this dependence by sorting these vectors in ascending order.

Figure 2 exemplifies four of the five patterns of our pattern set. (The pre-Block pattern is not shown because it needs a separate treatment.) At the right and below each pattern, linear heatmaps represent the four vectors of that pattern, before and after sorting. Note that the set of sorted vectors of one pattern has not the same behavior in the other patterns. Therefore, these vectors are of interest to our classification approach.

Another problem to overcome is the dependence of the matrix dimension. Matrices with different sizes would return vectors with distinct sizes, which would hamper the classification. For that reason, we opted by replacing each vector by a linear equation of its values, generated by linear regression. This approach generated a list of four equations (and eight coefficients) that summarizes these vectors and, by extension, the input matrix. The equations have the format $y_\phi = a_\phi x + b_\phi$, where ϕ should be replaced with $minRows$, $maxRows$, $minColumns$, or $maxColumns$. The linear regression also helps to deal with noise in the input matrix. Low noise ratios tend to cause little changes in the equation coefficients.

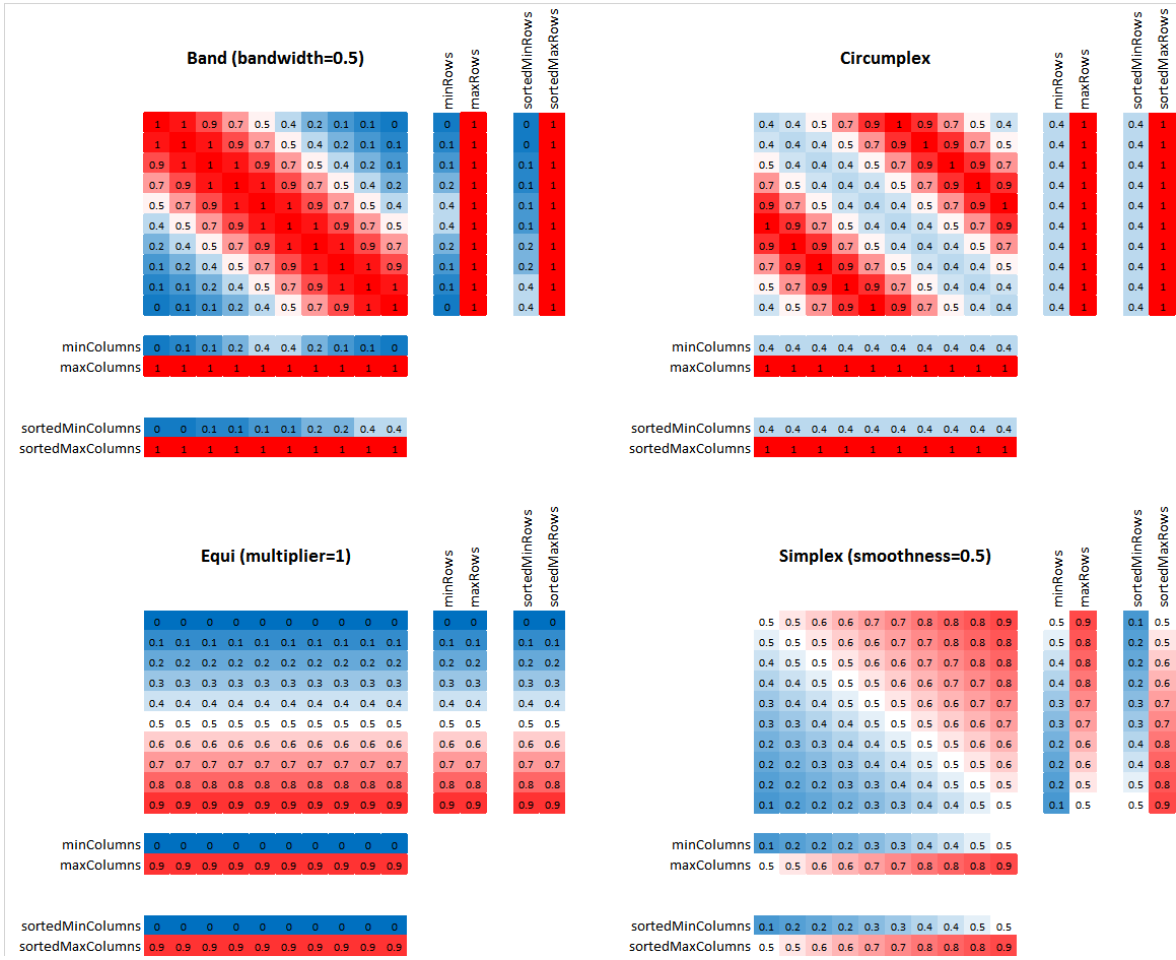


Figure 2. Examples of Band, Circumplex, Equi, and Simplex patterns in 10×10 matrices, and their respective vectors with minimum and maximum values per row and per column

Medina et al. (2016) opted by using only the angular coefficients of these equations for classifying an input matrix into one of the four patterns. In our current work we initially used all coefficients to try to produce a more reliable classification. During the work, we reduced this set as necessary.

We also noted that these coefficients do not help to identify the pre-Block pattern. However, we defined two other permutation-invariant boolean coefficients that aid in the identification of this pattern:

- *BalancedColumns*. Given a matrix normalized in values from 0 to 1, we call “high values” the values greater or equal to 0.5, and “low values” the ones lower than 0.5. Pre-Block and pre-Equi matrices have “balanced columns”, i.e., columns whose proportion of high (and low) values is about 50%. Noise may increase or decrease this proportion, so we consider a column as balanced if it has a proportion of high values between 40 and 60%. If a matrix has at least 80% of balanced columns, then *BalancedColumns* is *true*, which is quite indicative that the matrix is pre-Block or pre-Equi.
- *FewIntermediaryValues*. In a matrix normalized in values from 0 to 1, a Block pattern has only cells with value 0 or 1. Therefore, the occurrence of other values would reveal a tendency that the matrix is not pre-Block. However, noise can change this characteristic, and this verification must accept some other values closer to 0 or to 1. We define that values in the intervals $[0, 0.2]$, $]0.2, 0.8[$, and $[0.8, 1]$ are “low”, “intermediary”, and “high values”, respectively. We count the number of rows with at least 70% of their cells having only low or high values. If the matrix has 70% or more rows that have this characteristic, then *FewIntermediaryValues* is *true*, and this suggests that the matrix has a pre-Block pattern.

It is worth noting that the definitions of both coefficients are empirical. Nonetheless, they worked well in the experiments that we will present later on this paper.

In summary, we have ten coefficients that form a vector of permutation-invariant features of a matrix: $[a_{minRows}, b_{minRows}, a_{maxRows}, b_{maxRows}, a_{minColumns}, b_{minColumns}, a_{maxColumns}, b_{maxColumns}, BalancedColumns, FewIntermediaryValues]$. This feature vector is the basis for our classifier.

4.2 A Matrix Classifier Based on an Empirical Decision Tree

Based on the presented feature vector definition, we constructed a matrix classifier algorithm. We follow the assumption that our input matrices must contain one of the following patterns: pre-Band, pre-Equi, pre-Simplex, pre-Circumplex, or pre-Block.

First, we studied if our feature vectors would provide enough separability between the first four matrix patterns. We considered in this study a set of feature vectors derived from an ensemble of 1,026 matrices configured with distinct patterns, parameters and sizes. Our ensemble is divided into 342 groups of 3 matrices with the same configuration per group. The configurations were:

- A size from the set $\{100 \times 100, 150 \times 150, 300 \times 300\}$;
- A noise ratio from the set $\{0\%, 1\%, 5\%, 10\%, 30\%, 50\%\}$;
- A pattern from the set: Band with bandwidth values $\{0.1, 0.25, 0.5, 0.75, 1, 1.5\}$; Circumplex; Equi with multiplier values $\{0.5, 1, 2\}$; Simplex with smoothness values $\{0.1, 0.25, 0.5, 0.75, 1, 1.5\}$; and Block with values of parameter k belonging to the set $\{3, 5, 7\}$;

We used the software Tableau to study visually the relationship between feature vectors and matrix patterns. We used no statistical tool to help in this process. In our analysis, we concluded that we need only three coefficients of the defined feature vector to classify a matrix into one of the four patterns: $a_{minRows}$, $a_{maxRows}$, and $b_{minColumns}$. Figure 3 (top) shows a scatterplot of $a_{minRows}$ values per matrix pattern. We identified four data intervals by numbers from 1 to 4. These intervals indicate situations in which we have only Circumplex (1 and 3) or Band (2 and 4) patterns. Figure 3 (bottom) helps to analyze the remaining feature vectors, i.e., those with $a_{minRows} > 0.015$. We identified other five data intervals that are strongly related to a pattern: region 5 for Band pattern, regions 6 and 9 for Simplex, and regions 7 and 8 for Equi.

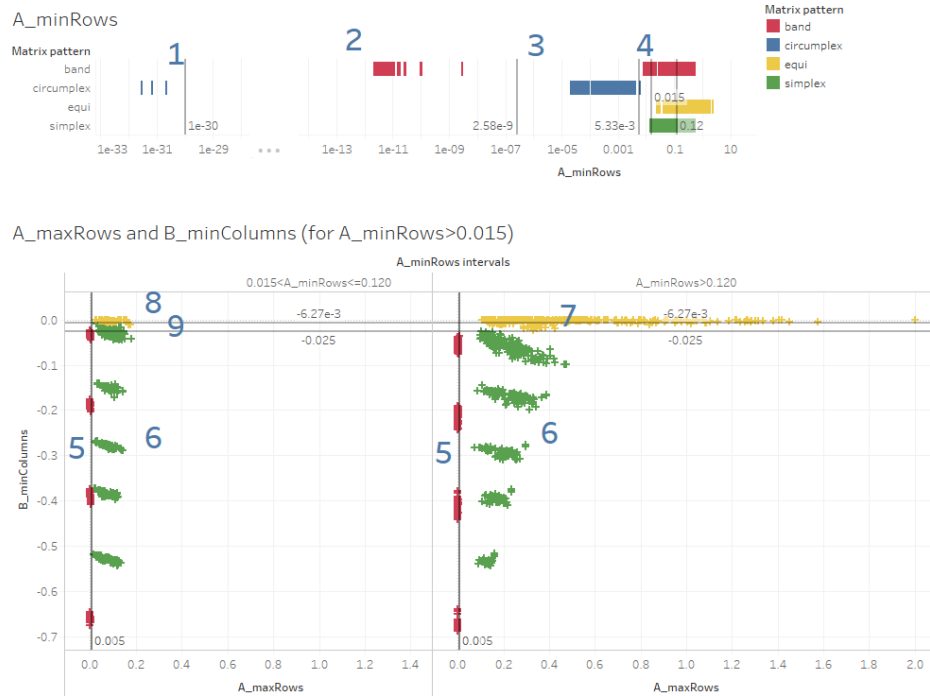


Figure 3. Values of feature vectors' variables per matrix pattern. Top: variable $a_{minRows}$ ($A_{minRows}$); bottom: variables $a_{maxRows}$ ($A_{maxRows}$) and $b_{minColumns}$ ($B_{minColumns}$). Blue numbers indicate regions of interest in each graphic

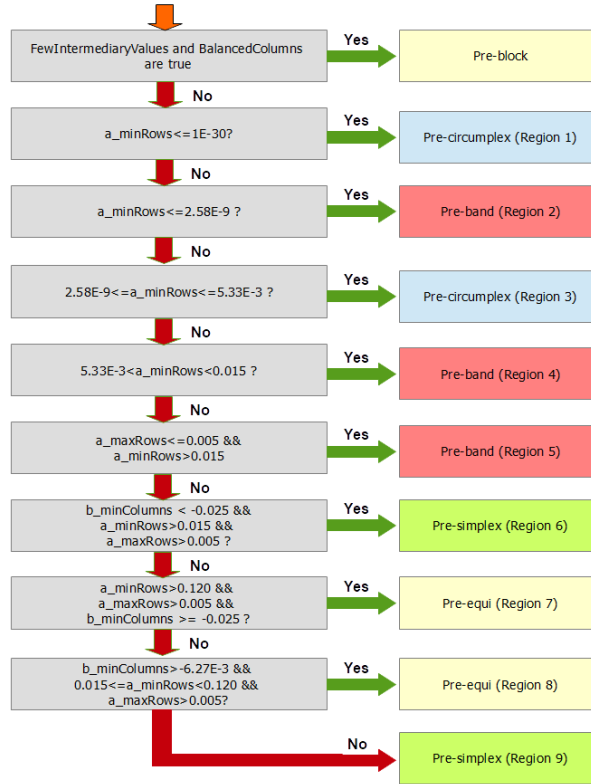


Figure 4. Empirical decision tree of the matrix classifier

We defined most of the decision tree in Figure 4 based on this previous analysis. This decision tree is the heart of the algorithm because a wrong classification would lead to choosing an inappropriate reordering algorithm.

Regarding pre-Block pattern, we noted that matrices whose properties `BalancedColumns` and `FewIntermediaryValues` are true could be classified as pre-Block. Consequently, this decision was added to the root of the decision tree in Figure 4. Section 5 will discuss the quality of the proposed classifier.

4.3 Selection of Pattern-Focused Algorithms

The Step 2 must select an algorithm that is good at unveiling the pattern informed by Step 1. According to the works presented at Section 3, we defined the following algorithms to be selected for each pattern: FVS (Silva et al. 2017) for pre-Equi or pre-Simplex, Block Reordering (Santos et al. 2016) for pre-Block, and Polar Sort (Silva 2019) for pre-Circumplex or pre-Band. These choices considered the high-quality of the output of these reordering algorithms and their fast running time in comparison to other algorithms, as presented in their respective papers.

5. RESULTS AND DISCUSSION

In this section, we first discuss the quality of our classifier. Next, we present how Hybrid Sort reorders some synthetic and real-world matrices. We also analyze the time complexity of this reordering method.

We tested our classifier with a set of 45,600 matrices. This set was divided into 50-matrices groups, and matrices of the same group have the same configuration. The set of matrix sizes was: $\{25 \times 25, 50 \times 50, 75 \times 75, 100 \times 100, 150 \times 150, 200 \times 200, 300 \times 300, 500 \times 500\}$. We considered the same noise ratios, patterns and pattern configurations used to construct the classifier. The test compared the actual patterns of

each matrix with the patterns assigned by the classifier. We opted by creating matrices that have the data patterns in its current permutation, instead of scrambling its rows and columns. This decision makes sense because sorting the vectors of Section 4.1 makes the classifier immune to the ordering of rows and columns of the input matrix. Thus, we saved experiment time that would be lost with unnecessary permutations.

Figure 5(a) presents the results of this analysis as a confusion matrix. Its diagonal values are higher than 91% for most patterns. The exception is for pre-Block matrices, with 82.9% of correct classification, and with 13.7% of the input matrices incorrectly classified as pre-Circumplex. Nevertheless, we believe that these values are considerably good for an empirically-defined classifier. Besides, the F2-measure has good scores for all patterns: 95%, 93%, 92%, 86%, and 85% for Band, Simplex, Equi, Circumplex, and Block, respectively.

For comparison, Figure 5(b) shows a confusion matrix for the SMB's classifier. This matrix does not have a "block" column or a "block" row because this classifier does not deal with pre-Block matrices. However, it was created by the same process that create the Hybrid Sort confusion matrix, except by the absence of pre-Block matrices. We noted that SMB's classifier did many errors when it classified pre-Band, pre-Circumplex or pre-Equi matrices as pre-Simplex ones (fifth column). Besides, the main diagonal of this confusion matrix has many values lower than 50%, which indicates that the classifier has low precision. The high number of matrices classified as "other" is also a concern. Therefore, we perceived that the results of Hybrid Sort's classifier are better than SMB's one.

In summary, we observed that it was possible to construct a classifier whose classification results surpass SMB's ones. Also, we noted that it is possible to improve our classifier.

Actual Class	Assigned Class				
	band	block	circumplex	equi	simplex
band	96,1%	0,0%	1,0%	0,7%	2,2%
block	2,5%	82,9%	13,7%	1,0%	0,0%
circumplex	6,4%	0,0%	92,3%	0,0%	1,3%
equi	3,3%	0,0%	0,0%	91,8%	4,9%
simplex	6,4%	0,0%	0,4%	1,2%	92,0%

a) Hybrid Sort

Actual Class	Assigned Class				
	band	circumplex	equi	simplex	(other)
band	11,8%	4,9%	0,0%	47,1%	36,2%
circumplex	11,0%	43,9%	0,0%	36,7%	8,4%
equi	0,0%	0,0%	19,8%	73,7%	6,5%
simplex	0,0%	0,0%	0,0%	100,0%	0,0%

b) SMB
















Figure 5. Confusion matrices of (a) Hybrid Sort (b) and SMB's classifiers

Table 1 exemplifies the output of Hybrid Sort for five synthetic input matrices with distinct patterns. Its first row indicates pattern names and configurations, and the second one ("Original") shows an instance of these patterns. All instances have noise ratio 0.1 and size 200×200 . The third row presents randomly permuted versions of these instances. The last row shows these shuffled matrices after being reordered by Hybrid Sort. In all patterns, there is a high visual similarity between matrices in the second and fourth rows.

As a first real-world example, we present a matrix that compares pairwise similarity of 12 mood-related adjectives (Browne 1992). This matrix has the same adjectives in both columns, and each cell shows how 472 persons correlate two adjectives. Figure 6(a) shows this table in its original order. Figure 6(b) shows how a scrambled version of this table is ordered by Hybrid Sort. Note that the rows and columns of (a) and (b) are in the same circular ordering. Both versions unveil a Circumplex pattern that shows similarities of groups of adjectives, such as "relaxed" and "calm", or "drowsy" and "dull".

A second real-world example is a matrix that correlates crimes and offenses (Thurstone 1927). For each pair of elements in a set of 15 crimes and offenses, 266 persons informed which of these elements they consider more serious. These answers were summarized as a set of triples $[e1, e2, p]$, where p indicates the proportion of answers "e2 is more serious than e1". In the resulting table, row labels have values of $e1$, and column labels have values of $e2$. Each cell (r, c) has the proportion in which the crime (or offense) in column label c is more serious than the one in the row label r . Figure 6(c,d) shows this table in alphabetical order, and the same table after being reordered by Hybrid Sort. In Figure 6(c), homicide and rape have red columns that make evident the seriousness of these crimes for the interviewed individuals. When Hybrid Sort received this table as input, the algorithm classified the table as a pre-Simplex matrix, and consequently, it ordered the matrix with FVS. In the resulting heatmap in Figure 6(d), the Simplex-like pattern highlights an overall ordering of the people's opinions about the seriousness of the analyzed crimes and offenses. The ordering provided by Hybrid Sort also evidences that rape, homicide, abortion, seduction, and kidnapping were considered as the most serious crimes.

Table 1. Examples of Hybrid Sort’s output when reordering matrices with canonical data patterns

Pattern configuration	Band (bandwidth =0.5)	Block (k=0.5)	Circumplex	Equi (multiplier =1.0)	Simplex (smoothness =0.5)
Original					
Shuffled					
Reordered					

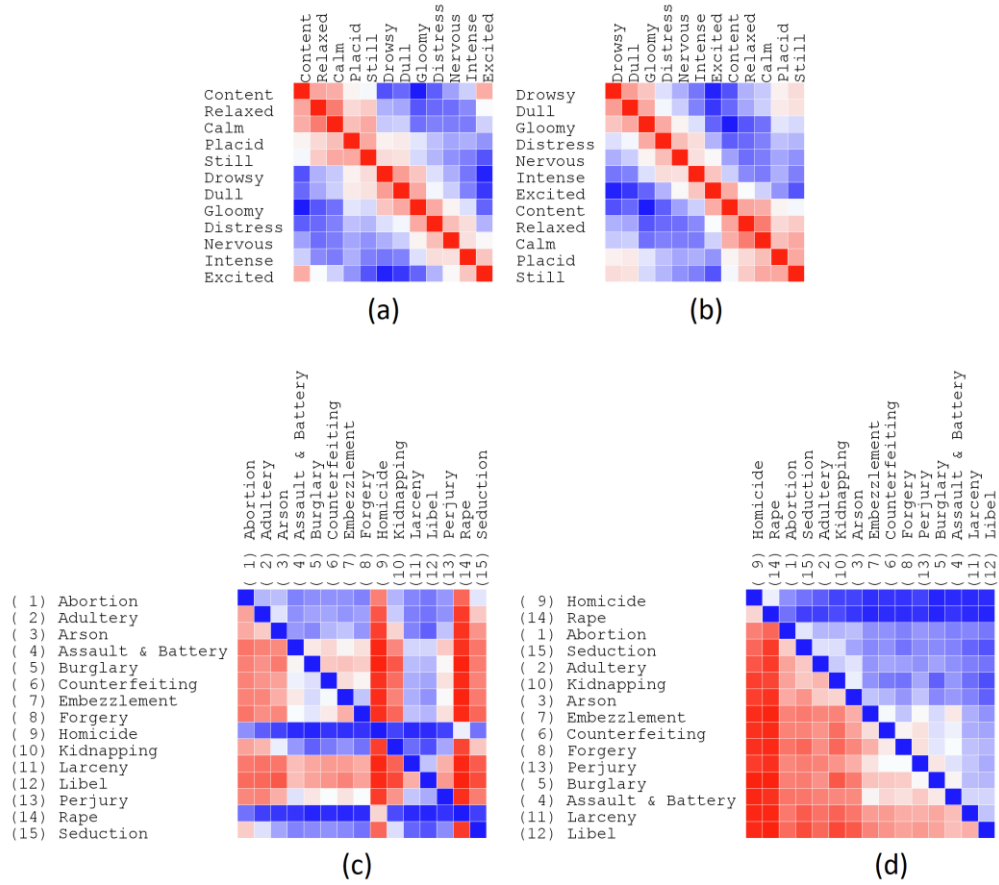


Figure 6. (a) Original and (b) Hybrid Sort’s ordering of the “mood adjectives” matrix; (c) original and (d) Hybrid Sort’s ordering of the “Crimes” matrix

The asymptotic time complexity of Hybrid Sort is as follows. The classification step is dominated by activities that browse the entire matrix, so it has time complexity $O(n^2)$. The algorithm selection is straightforward and has time complexity $O(1)$. The reordering step depends on the selected algorithm, and the one with the worst time complexity is Polar Sort ($O(n^3)$). Therefore, we conclude that Hybrid Sort has time complexity $O(n^3)$. This complexity could be improved if we replace Polar Sort by a faster algorithm with output quality as good as it has for pre-Band and pre-Circumplex matrices.

The main limitation of this research is that the decision tree used by the matrix classifier supposes that every input matrix is pre-Band, pre-Circumplex, pre-Simplex, pre-Equi or pre-Block. Therefore, attempts to use Hybrid Sort to reorder matrices that do not have one of the five patterns considered by the decision tree will possibly result in not appropriate orderings of these matrices.

It is worth noting that we do not present in this paper experiments with synthetic matrices that could justify the quality of the output matrices reordered by Hybrid Sort. We opted by doing this because the output quality of the reordering methods used by Hybrid Sort was already analyzed and attested in their respective papers.

6. CONCLUSION

We presented in this work a matrix reordering method – Hybrid Sort – based on three steps: input classification, selection of a reordering algorithm, and execution of this algorithm. Its classifier uses a decision tree based on feature vectors extracted from the input matrix. It classifies the input matrix into one of the five canonical data patterns. The algorithm selector uses the inferred class of the matrix to choose FVS, Polar Sort or Block Reordering algorithms for the third step. The last step uses the selected algorithm to reorder the input matrix. The classifier reached good precision measures that overcome the SMB's classifier. The overall solution has time complexity $O(n^3)$. Future work includes: replacing Polar Sort by a faster multidimensional projection algorithm, aiming to reduce Hybrid Sort's time complexity; improving the classifier by using machine learning strategies for pattern detection; and expanding the algorithm to consider other patterns.

ACKNOWLEDGEMENT

We thank the grant #2224/20 from FAEPEX/PRP/Unicamp.

REFERENCES

- Barros, R.T.C. et al., 2019. Use of reorderable matrices and heatmaps to support data analysis of students transcripts. Companion Proceedings of the 32nd SIBGRAPI – Conference on Graphics, Patterns and Images. Rio de Janeiro, Brazil, pp. 219–222.
- Behrisch, M. et al., 2016. Matrix reordering methods for table and network visualization. *Computer Graphics Forum* 35(3), pp. 693–716.
- Behrisch, M. et al., 2017. MAGNOSTICS: Image-based search of interesting matrix views for guided network exploration. *IEEE Transactions on Visualization and Computer Graphics* 23(1), pp. 31–40.
- Behrisch, M. et al., 2020. GUIRO: User-guided matrix reordering, *IEEE Transactions on Visualization and Computer Graphics* 26(1), pp. 184–194.
- Browne, M. W. 1992. Circumplex models for correlation matrices. *Psychometrika* 57(4), pp. 469–497.
- Card, S. K. et al. 1999. *Readings in Information Visualization: Using Vision to Think*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Liiv, I., 2010. Seriation and matrix reordering methods: An historical overview, *Statistical Analysis and Data Mining* 3(2), pp. 70–91.
- Medina, B. F. et al., 2016, Smoothed Multiple Binarization – Using PQR tree, smoothing, feature vectors and thresholding for matrix reordering. *Proc. of 20th Int. Conf. Information Visualisation (IV)*. Lisbon, Portugal, pp. 88–93.
- Rocha, M. M. N. et al., 2017. Circumplex Sort: a two-phase method for revealing circumplex data patterns in reorderable matrices. *Proc. Int. Conf. Computer Graphics, Visualization, Computer Vision and Image Processing 2017 and Big Data Analytics, Data Mining and Computational Intelligence 2017*. Lisbon, Portugal, pp. 181–188.
- Santos, A. M. et al., 2016, Block Reordering: Um algoritmo para evidenciar padrão Block em matrizes reordenáveis. *Workshop of Undergraduate Works (WUW) in the 29th Conference on Graphics, Patterns and Images (SIBGRAPI'16)*, São José dos Campos, Brazil.
- Silva, C. G. et al., 2017. A fast feature vector approach for revealing simplex and equi-correlation data patterns in reorderable matrices. *Information Visualization* 16(4), pp. 261–274.
- Silva, C. G., 2019. Polar Sort: combining multidimensional scaling and polar coordinates for matrix reordering. *Proc. Int. Conf. Interfaces and Human Computer Interaction 2019; Game and Entertainment Technologies 2019; and Computer Graphics, Visualization, Computer Vision and Image Processing 2019*, Porto, Portugal, pp. 239–243.
- Thurstone, L. L., 1927. The method of paired comparisons for social values. *The Journal of Abnormal and Social Psychology* 21, pp. 384–400.
- Wilkinson, L., 2005. *The Grammar of Graphics*. Springer Science & Business Media. New York, USA.