Visualizing Overlapping Biclusterings and Boolean Matrix Factorizations

THIBAULT MARETTE · PAULI MIETTINEN · STEFAN NEUMANN

MOTIVATION

- Finding biclusters (sets of items closely related to each other) is a well-studied problem
- Visualization helps *assessing* outputs of biclustering algorithms

Research Questions:

- How can we visualize a bipartite graph given a set of biclusters?
- Can we define metrics to compare visualizations?

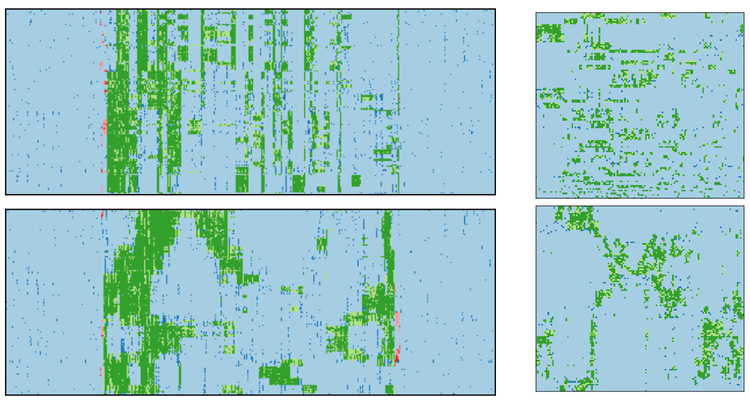


Figure 1: Visualization of the same biclustering using permutation from ADVISER [2] (top) and from our TSP-based heuristic (bottom).

Related work. Jin et al. [3] and Colantonio et al. [2] studied the visualization of a given set of overlapping biclusters as a biadjacency matrix.

- Colantonio et al. [2] proposed a greedy heuristic called ADVISER to minimize gaps in the visualization of each bicluster.
- However, similar but non-overlaping biclusters are not visualized close to one another.

PRE-PROCESSING

Problem setup. Let $G = (R \cup C, E)$ a bipartite graph, and $((R_1, C_1), \ldots, (R_k, C_k))$ a biclustering of G, where $R_i \subseteq R$ and $C_i \subseteq C$ for all i. We Compute a row permutation σ_R and a column permutation σ_C of the biadjacency matrix A to *optimally* visualize the biclustering. In the visualization, we set $A_{\sigma_R(r),\sigma_C(c)} = 1$ iff $(r, c) \in E$.

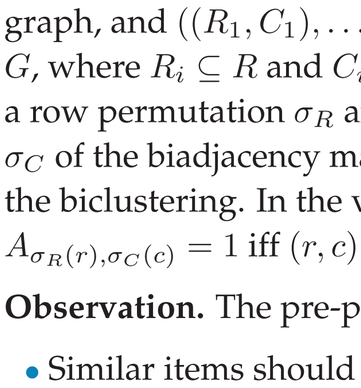
Observation. The pre-processing stems from:

- Similar items should be close to each other in the final permutation

⇒ Rows and columns from the same clusters should be contiguous in the final permutation.

We partition the rows and columns into *row blocks* and *column blocks*. We ensure that each cluster can be expressed as the union of a set of blocks.

Formally, for $r \in [m]$, let $clusters_R(r) = \{i : r \in R_i\}$ denote the set of indices of all row clusters that contain row r.



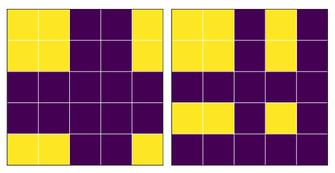
Now, the *row block of* r is given by $block_R(r) = \{r' : clusters_R(r) = clusters_R(r')\}$, i.e., it is the set of all rows r' that are contained in exactly the same row clusters as r.

⇒ We only have to find permutations of the *row and column blocks*.

OBJECTIVE FUNCTIONS

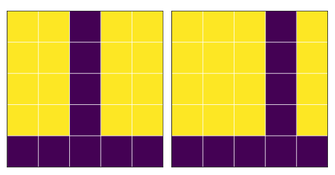
The three criteria that we study are as follows:

Proximity: Rows and columns of each bicluster should be close to each other.



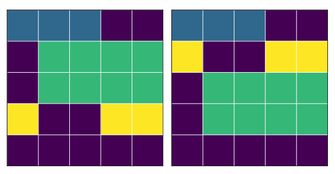
$$S_{\text{prox}}(R_i, C_i) = [\max\{R_i\} - \min\{R_i\} + 1] \cdot [\max\{C_i\} - \min\{C_i\} + 1]$$

Size of the consecutive cluster areas: Rows and columns of each bicluster should form large consecutive areas.



$$S_{\text{clArea}}(R_i, C_i) = \sum_{(X,Y) \in cons(R_i) \times cons(C_i)} |X \times Y|^2.$$

Size of uninterrupted areas: Areas that belong to biclusters should form large uninterrupted areas (not limited to individual biclusters).



$$S^R_{\text{uninter}}(b^R_i) = \sum_{Y \in cons(nonzero(b^R_i))} |b^R_i \times Y|^2.$$

- All three objective functions capture a different aspect of a good visualization
- They represent a *global* state of the visualization
- They are costly to compute

DEMERIT

We introduce *demerit* as a fast-to-compute local distance function between two blocks. If two blocks are dissimilar, their demerit is high.

Formally, the *demerit* for row block b^R and column blocks b^C_i and b^C_j is given by:

$$demerit(b^R; b^C_i, b^C_j) = \begin{cases} |b^R| \cdot (|c_1 \cup c_2| + 1) \\ \qquad \text{if } c_1 = \emptyset \text{ or } c_2 = \emptyset, \\ |b^R| \cdot (|c_1 \cup c_2| - |c_1 \cap c_2|) \\ \qquad \text{otherwise,} \end{cases}$$

where $c_1 = clusters_R(b^R) \cap clusters_C(b^C_i)$, and $c_2 = clusters_R(b^R) \cap clusters_C(b^C_j)$.

To measure the *demerit of the column block permutation* σ_C, we set

$$demerit(\sigma_C) = \sum_{b^R \in \mathcal{B}^R} \sum_{i=1}^{t-1} demerit(b^R; b^C_{\sigma_C(i)}, b^C_{\sigma_C(i+1)}).$$

- The *demerit* is easier to optimize than the other three objective functions
- Can be computed locally
- Only have to consider consecutive pairs of column blocks $b^C_{\sigma_C(i)}$ and $b^C_{\sigma_C(i+1)}$
- Defined for all pairs of row and column blocks

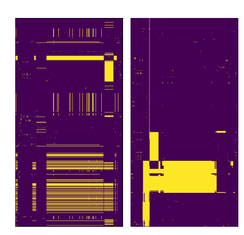
TSPheuristic. We use a TSP solver in order to find a permutation minimizing *demerit* between blocks. The permutation returned by the TSP solver is the permutation used in the visualization.

EXPERIMENTS

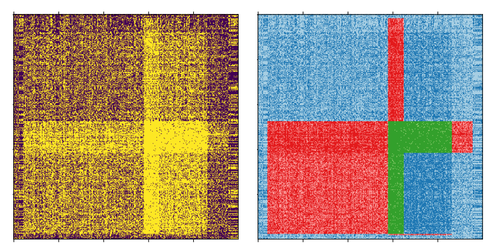
The code of our visualization tool is open-source and available online.

github.com/tmarette/biclusterVisualization

We picked our **color scheme** using colorbrewer [1] to ensure readability and accessibility.

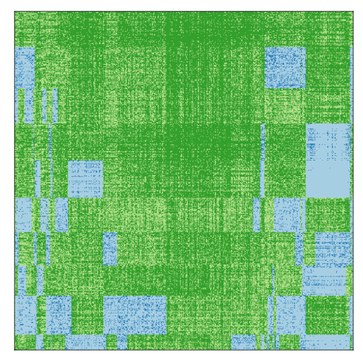


Ordering

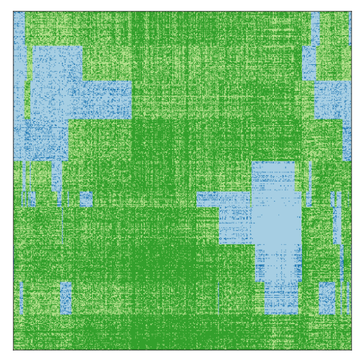


Color scheme

Qualitative Evaluation. We notice an increase in the quality of the visualization by comparing our method to the baseline algorithm ADVISER.

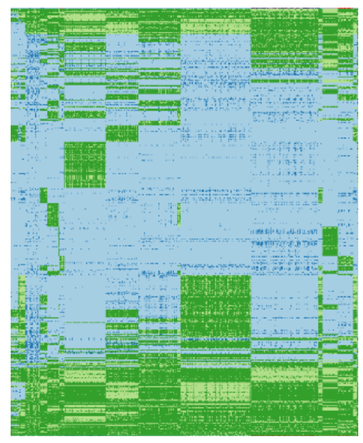


ADVISER

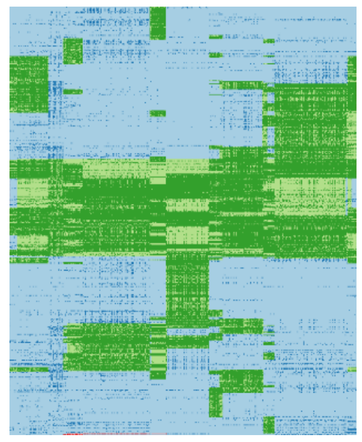


Our method

Increased size of uninterrupted areas.



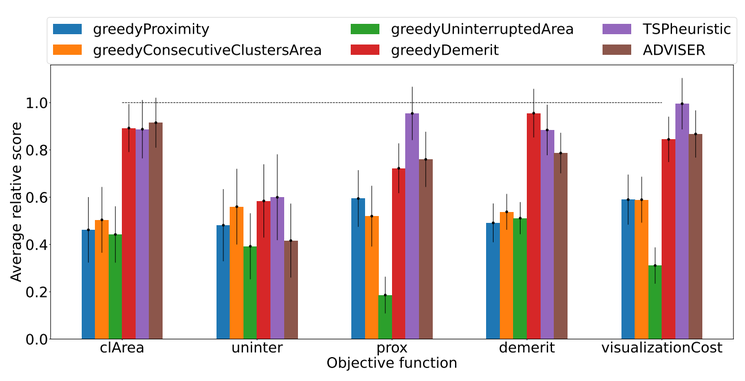
ADVISER



Our method

Increased coherence.

Quantitative Evaluation. We gathered objective functions values of all methods across multiple datasets, using varying numbers of clusters.



⇒ Minimizing the demerit achieves very high quality across all objective functions.

REFERENCES

[1] C. A. Brewer, M. Harrower, B. Sheesley, A. Woodruff, and D. Heyman. Colorbrewer 2.0: color advice for cartography. *The Pennsylvania State University. http://colorbrewer2. org/*.

[2] A. Colantonio, R. Di Pietro, A. Ocello, and N. V. Verde. Visual role mining: A picture is worth a thousand roles. *IEEE Trans. Knowl. Data Eng.*, 24(6):1120–1133, 2011.

[3] R. Jin, Y. Xiang, D. Fuhry, and F. F. Dragan. Overlapping matrix pattern visualization: A hypergraph approach. In *IEEE Int. Conf. Data Min.*, pages 313–322, 2008.

ACKNOWLEDGEMENT

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.