A Summarization-Based Pattern-Aware Matrix Reordering Approach

Zihan Zhou[1], Jiacheng Pan[1], Xumeng Wang[2], Dongming Han[1], Fangzhou Guo[1], Minfeng Zhu[✉3], and Wei Chen[✉1]

[1]State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China
[2]College of Computer Science, Nankai University, Tianjin, China
[3]Zhejiang University, Hangzhou, China
{zhouzihan, panjiacheng, dongminghan, guofangzhou, minfeng_zhu, chenvis}@zju.edu.cn
wangxumeng@nankai.edu.cn

Abstract

Matrix-based graph visualization is effective in revealing relationships among entities in graphs. The visibility of structural patterns depends on the ordering of rows/columns in matrices. Most existing approaches mainly settle on an ideal ordering according to quality metrics, which emphasize certain types of patterns but ignore others. This paper proposes a summarization-based pattern-aware reordering approach to highlight multiple patterns simultaneously. First, a pattern-aware graph summarization utilizes the Minimum Description Length (MDL) technique to identify various types of patterns from the input graph. Second, we propose a coarse-to-fine reordering mechanism to generate matrix-based visualizations that maintain the structure of all identified patterns. Experimental results of two comparative studies and a user study on several datasets demonstrate that our approach simultaneously highlights more types of patterns than other approaches and performs well across multiple quality metrics.

Keywords: Graph summarization. Graph visualization. Matrix-based visualization. Matrix reordering.

1. Introduction

Graph visualization is beneficial for depicting relationships among entities, which are concerned with multiple scenarios, ranging from interpersonal relationship analysis in sociology [37, 4, 34, 8] to protein structures analysis in biology [36]. Matrix visualization is one of the most classical approaches in graph visualization to uncover topology structures. In a matrix, each row/column corresponds to an entity in the graph data. The relationship between two entities is encoded by drawing a cell at the intersection of the two entities' corresponding rows/columns. Matrix visualization is free from node overlapping and edge crossing issues that trouble node-link diagrams [7]. However, meaningful patterns in matrices (e.g., diagonal and off-diagonal blocks) can only be observed when rows/columns are arranged in a specific order. Therefore, the order of rows/columns plays a significant role when implementing analysis tasks with matrix visualizations.

In real-world scenarios, analysis tasks, such as exploring multiple information diffusion patterns within a specific population, have a need to simultaneously analyze multiple patterns. However, it is challenging to find an ordering scheme that can capture and highlight multiple patterns. This is because the various orders that may be employed to highlight different patterns may also conflict with one another, making it both necessary and challenging to identify multiple patterns within a matrix. Existing reordering algorithms mainly recommend an ordering scheme through an optimization process driven by metrics designed for certain patterns (e.g., blocks, stars, and bands, as shown in Tab.1). However, those metrics appreciate certain patterns but ignore others and even suppress them. For instance, MinLA [31] pursuits a high linear arrangement (LA) metric to highlight diagonal blocks. However, this approach results in low scores for cells further away from the diagonal, ultimately disrupting the visualization of off-diagonal patterns (see the second column of Tab.1). Similarly, Moran's I (MI) [35] (see the last column of Tab.1) emphasizes block patterns and star patterns but dislikes sparsely connected bands. Analysts always have to employ diverse reordering algorithms to explore patterns. Nevertheless, selecting reordering schemes needs expertise in reordering algorithms, and browsing multiple matrices could be exhausting.

To address this challenge, we propose a pattern-aware approach for highlighting multiple patterns in symmetric binary matrices. The proposed approach consists of two steps: pattern-aware graph summarization and coarse-to-fine ma-

trix reordering.

	Pattern	PR	LA	BW	MI
(a)	Block	0.78	0.90	0.70	0.81
(b)	Star	0.47	0.72	0.40	0.70
(c)	Off-diagonal Block	0.33	0.40	0.10	0.89
(d)	Bands	0.64	0.80	0.80	0.40
	Anti Pattern	**PR**	**LA**	**BW**	**MI**
(e)	Anti Pattern	0.38	0.65	0.20	0.40
(f)	Bandwidth Anti Pattern	0.47	0.77	0.60	0.56

Table 1. Patterns and anti-patterns with four common metrics. All metrics are calculated according to their definitions and normalized to a range of [0, 1]. They are positively correlated with quality. It is noteworthy that the band pattern is characterized as a chain, resulting in a lower MI score due to the emphasis of MI on connectivity.

First, our approach describes a graph as a set of base patterns (e.g., Tab.1a-d) with corrections to emphasize patterns with salient structures. We search for the description with minimum length and utilize it as the pattern-aware graph summarization. Second, we design a reordering mechanism to generate matrix visualizations according to the graph summarization. To preserve the identified patterns, we consider each pattern as a group of nodes, namely a supernode, and reorder supernodes to arrange the matrix from a coarse level. Then, we fine-tune the inner structure of each pattern to determine the order of each row/column at the fine level. Following the above two steps, salient patterns can be highlighted in priority. Overall, the graph summarization techniques sketch out a graph by searching a combination of salient patterns, which provides a solution for pattern conflicts. The ordering scheme derived from the summarization can, therefore, restore the visual patterns of more salient structures than other algorithms, as demonstrated in the two quantitative experiments. We also implemented a user study to evaluate the effectiveness of our approach in terms of end-user pattern awareness.

The contributions of this paper include:

1) A summarization-based pattern-aware matrix reordering approach that summarizes salience patterns from graphs and reorders corresponding matrices to highlight them;
2) A series of quantitative experiments designed to evaluate our summarization algorithm and reordering mechanism.

2. Related Work

2.1. Graph Summarization

With the rapid growth of real-world datasets, the corresponding graph visualization suffers from visual clutter, posing challenges for data analysis. To solve this issue, graph summarization provides an overview of a graph by aggregating nodes and edges.

Existing studies mainly use bit-compression approaches to summarize graphs [25] without information loss. Bit-compression-based approaches utilize the two-part MDL principle to minimize the description length of the graph. For instance, Navlakha et al. [27] presented a representation for graphs as a <summary, corrections> pair and used the MDL principle to yield a coarse level representation. The following work [10] proposed bipartite graph mining with MDL (BL-MDL). It follows a bottom-up and greedy approach to find a better summary graph. The Vocabulary-based summarization of Graphs (VoG) [22] summarizes a graph based on its distinct subgraphs of certain fixed types (cliques, bi-cliques, stars, chains) and decomposes the input graph into possible subgraphs. The MDL principle is then utilized to identify the type of each subgraph and define the graph summary of the subgraph composition, which may non-redundantly describe the graph.

Applying MDL-based graph summarization approaches can losslessly compress visualizations into a contracted representation with summarized structures or patterns [22, 10]. For matrix visualizations, arranging rows/columns according to the summarization can aggregate cells into patterns to achieve a simplified representation. Existing summarization approaches rarely consider the matrix reordering problem or only highlight patterns without optimizing the matrix quality metrics [22]. There is still a gap between graph summarization approaches and the matrix reordering problem. We propose an approach to fill the gap by introducing the two algorithms, Greedy [27] and VoG [22], to improve the quality of matrices while enabling them to highlight meaningful patterns.

2.2. Matrix Reordering

Node-link diagrams and matrix visualizations are two major approaches to visual graph data. Unlike node-link diagrams suitable for topology-based tasks, matrix visualization is deemed more appropriate for the task of identifying patterns for users [29]. However, the usefulness of matrix visualizations is always limited by unsuitable reordering methods for rows/columns. To address this issue, various matrix reordering strategies have been proposed. The typical reordering process consists of three steps: 1) convert data into a representation (e.g., adjacency matrix), which relates to the problem space, 2) iteratively rearrange the permutation in the problem space until the quality metric meets the desired threshold, and 3) reorder the matrix accordingly.

Problem Space Definition. Available space definitions include distance matrices [19], Laplacian matrices [2], and Eigenvector spaces [16, 20]. The general goal of these problem spaces is to cluster similar rows together and thus generate meaningful patterns.

Permutation Determination. The permutation determination procedures can be divided into six major categories: Robinsonian algorithms, spectral algorithms, dimension reduction algorithms, heuristic algorithms, graph-theoretic algorithms, and biclustering algorithms [6]. First, Robinsonian approaches [32, 3] mainly reorganize a matrix through clustering of similar rows/columns while separating dissimilar ones. The Bipolarization algorithm [19] upholds a similar principle and effectively illustrates the bi-polar organization (Tab.1a,c,e). Second, spectral methods [16] use eigenvalues and eigenvectors to calculate a permutation with efficient implementations. They project each row/column into the eigenvector space and use distances between eigenvectors to calculate the permutation. For example, Chen's rank-two ellipse reordering method [11] tends to recognize off-diagonal block patterns (Tab.1c). Third, dimension reduction techniques [30] produce a one-dimension representation that captures the (non-)linear relationships in data, which prioritize high-level patterns (Tab.1a) over detailed matrix patterns (Tab.1b). Fourth, heuristics [9] are able to transform reordering problems into alternative problem spaces. For example, Niermann et al. [28] utilized a genetic algorithm to obtain the best permutation which has the minimal number of bits to encode the matrix. This resulted in a tendency towards displaying orderly patterns. Fifth, graph-theoretic approaches [26] rely on leveraging the underlying graph topology structure to compute a linear order and optimize the layout cost function, such as profile, connectivity, similarity or shortest path. Finally, biclustering [13] conducts simultaneous clustering in two dimensions, detecting groups of rows/columns that exhibit analogous activity patterns. Visual block patterns with continuous colors (Tab. 1a,c) are hence highlighted.

Although patterns are regarded as an important goal of matrix reordering algorithms [6, 5, 23], they only highlight limited patterns. The metrics they aim to optimize focus on placing similar rows/columns together and gathering cells to form continuous blocks such as diagonal and off-diagonal blocks. To highlight a wider range of patterns, we propose a new approach that quantifies the pattern salience in matrices and selects the most salient patterns to highlight. Moreover, it is less limited by metrics because its pattern identification does not depend on metrics optimization.

3. Background

We employ mathematical symbols to explain the problem of matrix reordering in Tab.2. A graph $G = (V, E)$ can be represented in the form of adjacency matrix $A = (n \times n)$, where $n = |V|$ and each cell $A_{ij} = 1/0$ denotes whether nodes i and j are connected by an edge. The matrix of an undirected and unweighted graph is symmetric and binary and can be visualized by an $n \times n$ bitmap, namely a matrix visualization.

Symbol	Description
$G(V,E)$	Graph with a node-set V and an edge-set E
A	The adjacency matrix of G
$\mathbb{G}(\mathbb{V},\mathbb{E})$	Summary graph with super-nodes $\mathbb{V}$ and super-edges $\mathbb{E}$
$\mathbb{A}$	The corresponding adjacency matrix of $\mathbb{G}$
$d(G)$	The density of G
$n(\mathbb{V})$	Original nodes in the super-node $\mathbb{V}$
Ω	Vocabulary of structure types
$\omega \in \Omega$	One structure in the vocabulary Ω
$\oplus$	Exclusive OR, the symmetric difference of two sets
C	Corrections for restoring $\mathbb{G}$ to G
$C_\omega(u)$	Corrections of encoding u with structure ω
$\Gamma(v)$	Neighbors of a node v
$\Pi_{u,v}$	All node pairs between super-nodes u and v
$A_{u,v}$	Connected node pairs between super-nodes u and v
L_G	The length of encoding G with the two-part MDL
$cost(u)$	The cost of describing a node u
$cost_\Omega(u)$	The cost of describing a node u with the vocabulary Ω
$cost(e_{u,v})$	The cost of describing an edge $e_{u,v}$
$s(u,v)$	The cost reduction of merging nodes u & v

Table 2. Definition of symbols.

Different ordering schemes can significantly affect the effectiveness of matrix visualizations. Reordering approaches aim to generate an ordering for all rows/columns that can clearly represent patterns. In this section, we introduce the following perspectives for assessing an ordering scheme.

3.1. Patterns

Matrix reordering approaches mainly consider the following six types of visual patterns (see Tab.1) summarized by existing studies [6, 5, 23].

- **Block patterns** refer to a contiguous group of cells on the main diagonal of the matrix, with a size of at least 2×2 cells. They correspond to cliques where nodes are densely connected in topology structures.
- **Off-diagonal block patterns** are blocks of cells placed symmetrically off the diagonal. Their topology structures can be regarded as bi-cliques, where one set of nodes connects with another set while nodes within the same set do not connect.
- **Star patterns** have a horizontal line and a vertical line that is connected, with or without discontinuities.

They represent star structures where one node (hub) is connected to several other nodes (spokes) while spoke nodes are not connected.

- **Band patterns** refer to parallel lines in a matrix that follow the diagonal. Their topology structures can manifest as chain structures or cycles.
- **Noise anti-pattern** indicates a lack of apparent topology structures in the matrix. Or the reordering approach fails to capture the underlying topology structure, which is undesirable.
- **Bandwidth anti-pattern** does not correspond to any meaningful topology structure in the input graph. They often arise as a typical outcome of optimizing for the bandwidth metric.

3.2. Quality Metrics

Existing studies proposed three typical distance-based measures: the profile [15], linear arrangement [21], and bandwidth [17], to measure the quality of the matrix. Given a one-dimensional alignment of the nodes ϕ, the three measurements calculate the distance between two nodes (i.e., u and v) in G through the following equations.

1) **Profile (*PR*)** counts the sum of the distance from the diagonal to the farthest-way non-zero cell for each column.

$$PR(\phi,G)=\sum_{u\in V}(\phi(u)-\min_{v\in\{u\}\bigcup\Gamma(u)}\phi(v)), \tag{1}$$

where $\Gamma(u)=\{v\in V:(u,v)\in E\}$.

2) **Linear arrangement (*LA*)** calculates the sum of every two connected nodes' distances in the alignment ϕ.

$$LA(\phi,G)=\sum_{(u,v)\in E}\lambda((u,v),\phi,G), \tag{2}$$

where $\lambda(u,v)=|\phi(u)-\phi(v)|$.

3) **Bandwidth (*BW*)** measures the maximum distance between every two connected nodes along rows.

$$BW(\phi,G)=\max_{(u,v)\in E}\lambda((u,v),\phi,G). \tag{3}$$

To evaluate the salience of patterns in matrices, **Moran's I (*MI*)** [35] is proposed to quantify the continuous block patterns, without specifically aiming to specify what a pattern actually constitutes.

$$MI(\phi,G)=-1+2\sum_{i=1}^{n-1}s(G,\phi(i),\phi(i+1)), \tag{4}$$

where $s(G,u,v)=c_B(G)\cdot B(G,u,v)+c_W(G)\cdot W(G,u,v)$ measures the neighborhood similarity between u and v. Behrisch et al. [5] presented a pattern-driven quality measurement called Magnostics. Magnostics assess how a reordering result can reveal a group of patterns based on the similarity between the matrix image and base patterns. However, it regards the entire matrix as a global pattern and does not consider local patterns that are more common because graphs are usually composed of multiple substructures (local patterns). Consequently, this metric is less suitable to compute benchmarks [35] and we did not include it in our subsequent analyses.

Observing how these measures work can help us summarize the goal of matrix reordering because they are the optimization targets of multiple existing reordering algorithms. We list the quality score for each pattern and anti-pattern in Tab.1 on the four metrics (*PR*, *LA*, *BW*, and *MI*). After calculating the scores based on their definitions, we normalize them and correlate them positively with the quality (higher scores indicate better quality):

$$\hat{PR}=1-\frac{PR}{|V|\times(|V|-1)/2},\quad \hat{LA}=1-\frac{LA}{|V|\times|E|},$$
$$\hat{BW}=1-\frac{BW}{|V|},\quad \hat{MI}=\frac{MI+1}{2}. \tag{5}$$

Optimizing distance-based metrics (*PR*, *LA*, *BW*) gathers cells to the diagonal and hence forms block patterns. They were proven to be less responsive to other patterns such as stars and bands and may produce anti patterns [35]. Although Beusekom et al. [35] introduced *MI* to alleviate such phenomenon, the *MI* scores of sparsely connected patterns are relatively lower (Tab.1), such as bands (0.40).

3.3. Graph Summarization

Graph summarization describing patterns in a graph can be used to inspect the representation effectiveness of a matrix with an ordering scheme. Here, we introduce two generation approaches to graph summarization: Greedy [27] and VoG [22].

3.3.1 Greedy

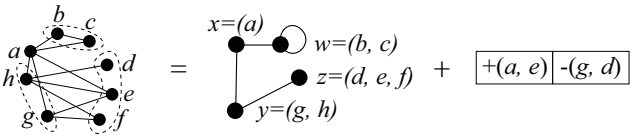


Figure 1. An example shows how Greedy [27] works. Greedy summarizes the original graph with a summary graph. The summary graph can be restored to the original graph by adding or deleting correction edges.

Greedy [27] aims at generating a brief description to summarize graph structure. It shortens the description of a graph by aggregating similar nodes and edges into super-nodes and super-edges, as shown in Fig.1. Greedy first identifies super-nodes by merging neighbor nodes with similar connections to others. For example, nodes h and g have almost identical neighbors (d, e, and f). Thus, Greedy merges them into two super-nodes *y={g, h}* and *z={d, e, f}* and adds a super-edge *(y, z)* to indicate that nodes d, e, and f all connect to nodes g and h. However, the original graph does not

contain the edge *(g, d)*. Hence, Greedy increases a correction *-(g, d)*. Instead of describing the original four edges among nodes *h, g, d, e*, and *f*, the summary only has one super-edge with one correction edge *-(g, d)*. It saves the description length.

Following such a principle, Greedy compresses the original graph $G(V,E)$ into the summary graph $\mathbb{G}$ with corrections C. The summary graph $\mathbb{G} = (\mathbb{V},\mathbb{E})$ consists of super-nodes $\mathbb{V}$ and super-edges $\mathbb{E}$. Each super-node contains one or more original nodes. Each super-edge retains edges across different super-nodes. The corrections C consist of nodes and edges not expressed by $\mathbb{G}$: $C = \mathbb{G} \oplus G$.

The goal is to compute an optimal representation of nodes (V) and edges (E), which minimizes the description length, defined as the cost sum of all super-nodes:

$$L_G = \sum_{u \in \mathbb{V}} cost(u). \tag{6}$$

The *cost* of a super-node u and a super-edge $e_{u,v}$ are defined as:

$$cost(u) = \sum_{v \in \Gamma(u), u \in \mathbb{V}} cost(e_{u,v}), \tag{7}$$

$$cost(e_{u,v}) = min(|\Pi_{u,v}| - |A_{u,v}| + 1, |A_{u,v}|), \tag{8}$$

where $\Pi_{u,v}$ means all original node pairs (a,b) from super-nodes u to v ($a \in u$ and $b \in v$), and $A_{u,v} \subseteq \Pi_{u,v}$ means *connected* original node pairs. Greedy chooses a cheaper encoding scheme to express the connections between u and v. If nodes in u are tightly connected to nodes in v, namely $|\Pi_{u,v}| - |A_{u,v}| + 1 > |A_{u,v}|$, Greedy builds the super-edge $(+1)$ and encoding the missing edges $(\Pi_{u,v}| - |A_{u,v}|)$ as the corrections. Otherwise, Greedy removes the super-edge and encodes the existing edges $(|A_{u,v}|)$ as the corrections.

Greedy approximates the global goal (the minimum cost of the graph) with a local optimization goal. It utilizes a greedy process to iteratively merge the pair of nodes with the largest cost reduction:

Step 1: For every two nodes within a two-hop distance, merge them and compute their costs before and after the merge. Store the node pair into a max heap if the cost decreases.

Step 2: Pop out and merge the node pair with the max cost reduction s from the heap. Then update the costs of their neighbors within two hops and remove node pairs without cost reduction.

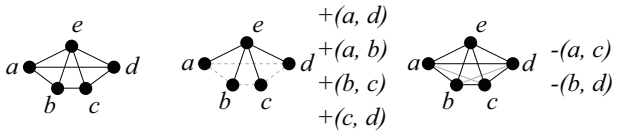
Step 3: Repeat step 2 until the heap is empty.

The cost reduction s of merging u and v into a super-node w is formally defined as:

$$s(u,v) = (cost_u + cost_v - cost_w)/(cost_u + cost_v). \tag{9}$$

3.3.2 VoG

VoG [22] defines the form of graph summarization as predefined structures with corrections. It chooses the structure with the least encoding cost to encode a subgraph. The subgraph in Fig.2a can be represented as a star (Fig.2b) with four corrections (adding edges (a,b), (b,c), (c,d), and (d,a)). However, encoding it as a clique (Fig.2c) has fewer corrections (removing edges (a,c) and (b,d)). Thus, VoG chooses to encode the subgraph as a clique.



(a) Original Graph (b) Encoding as a Star (c) Encoding as a Clique

Figure 2. An example shows how VoG [22] works. (a) is the original graph. (b) VoG first encodes it as a star, which requires four correction edges to recover the original graph. (c) VoG encodes the original graph as a clique, which requires only two corrections.

VoG first uses SlashBurn [24] to split the graph into several subgraphs. To find the best structure for encoding a subgraph, VoG proposes a pre-defined "vocabulary" of structures ($\Omega = \{clique, star, chain, bi\text{-}clique, \ldots\}$) with their cost definitions. VoG assigns a structure type for each subgraph with the least cost by traversing the "vocabulary".

4. Pattern-aware Matrix Reordering

A convenient and efficient reordering approach should break the limitation of pattern types while preserving the key ideas of optimization goals: **G1**) reducing isolated black/white cells by arranging cells with the same color vertically or horizontally (*MI*) or gathering black cells close to the diagonal (*PR*, *LA*, and *BW*), and **G2**) forming important patterns on a broader range of pattern types (e.g., stars and bands).

To achieve the above two goals, we propose a pattern-aware matrix reordering approach to highlight multiple patterns following two steps. In the first step, we employ a pattern-aware graph summarization to extract salient patterns from the graph data and minimize isolated edges. Moving to the next step, our approach uses a coarse-to-fine matrix reordering mechanism to transform the graph summarization into a matrix and reorder the matrix at both coarse and fine levels. By doing so, we are able to effectively preserve and highlight the identified patterns. Our approach absorbs advantages from two state-of-the-art techniques, Greedy [27] and VoG [22], and overcomes their shortcomings by combining them into a unified framework.

4.1. Pattern-aware Graph Summarization

Summarizing above, minimizing isolated edges and selecting salient patterns can benefit from the minimum description length calculation. We first analyze the strengths and shortcomings of two graph summarization approaches to motivate ours.

Greedy. Minimizing the cost defined by Greedy (Eq.6) leads to three results: 1) nodes in one super-node are sparsely connected, 2) two sets of nodes belonging to two connected super-nodes are densely connected, and 3) nodes in one self-looped super-node are densely connected. When analogizing to the matrix ordering problem, the above three results lead to 1) empty blocks along the diagonal, 2) off-diagonal blocks, and 3) diagonal blocks, respectively. Besides these structures, other isolated edges are regarded as corrections, which cause higher costs. Thus, minimizing the cost equals reducing isolated edges. Based on this observation, Greedy can be used to reduce the isolated edges, which conforms to **G1**. However, Greedy supports only three types of patterns due to limited considerations in the cost definition.

VoG. VoG proposes several cost definitions of different structure types in a "vocabulary". This helps to identify the best structure type to represent a subgraph. Less encoding costs lead to more precise matches. When analogizing to the matrix ordering problem, VoG can help identify the base pattern of given blocks. When the given block (subgraph) matches one base pattern (topology structure), its encoding cost minimizes. It quantifies the salience of a pattern, which conforms to **G2**. However, the cost calculation of VoG needs a pre-defined subgraph division. The subgraph division approach [24] employed in VoG considers neither the "vocabulary" nor the cost calculation. Therefore, VoG could still miss significant patterns in matrices when the subgraph division is not the optimal segmentation.

To overcome their shortcomings, we propose a pattern-aware graph summarization algorithm, which coordinates Greedy and VoG into a unified framework. Specifically, we take advantage of both Greedy and VoG to re-define the cost of a super-node u:

$$cost(u) = \underbrace{cost_{\Omega}(u)}_{\text{cost of structure}} + \underbrace{\sum_{v \in \Gamma(u)} cost(e_{u,v})}_{\text{cost of connected edges}}, \qquad (10)$$

where Ω represents the "vocabulary", and $cost_{\Omega}(u)$ defined by VoG represents encoding a super-node u as a structure $\omega \in \Omega$. Minimizing $cost_{\Omega}(u)$ gathers original nodes into structures defined in $\omega \in \Omega$ (**G2**), which also reduces isolated nodes (**G1**). Meanwhile, minimizing $\sum_{v \in \Gamma(u)} cost(e_{u,v})$ gathers original edges into fully connected super-edges, which reduces the number of isolated edges (**G1**).

We compute $cost_{\Omega}(u)$ by encoding u as a structure $\omega \in \Omega$ with the minimum description length:

$$cost_{\Omega}(u) = \min_{\omega \in \Omega} \{1 + L(C_{\omega}(u))\}, \omega \in \Omega, \qquad (11)$$

where 1 is the cost of the structure ω and $L(C_{\omega}(u))$ is the cost of corrections. VoG gives several heuristics for finding the minimum corrections of encoding a super-node u as ω. We improve two of them (heuristics for finding chains and bi-cliques) to reach more accurate costs. Note that the "vocabulary" is extensible for customized patterns.

Algorithm 1 Correction of encoding a graph as a bi-clique

Require: $G(V,E)$;
Ensure: C: the correction, a set of edges;
Init correction $C = \varnothing$;
$lc \leftarrow MaxDegree(V)$; ▷ left center
$rc \leftarrow \varnothing$; ▷ right center
$deg_{max} \leftarrow 0$;
for $v \in \Gamma(lc)$ **do** ▷ neighbors of left center
  $edges \leftarrow A_{v,\Gamma(lc)}$; ▷ edges between v and $\Gamma(lc)$
  $deg \leftarrow degree(v) - |edges|$;
  $rc \leftarrow v$ if $deg >= deg_{max}$;
end for
$l \leftarrow \{lc\}, r \leftarrow \{rc\}, V \leftarrow V - \{lc, rc\}$;
for each node $v \in V$ **do**
  $cost_l \leftarrow |A_{v,l}| + |\Pi_{v,r}| - |A_{v,r}|$; ▷ encoding v in left
  $cost_r \leftarrow |A_{v,r}| + |\Pi_{v,l}| - |A_{v,l}|$; ▷ encoding v in right
  if $cost_l < cost_r$ **then**
    add v to l;
    add $+A_{v,l}$ and $-(\Pi_{v,r} - A_{v,r})$ to C;
  else
    add v to r;
    add $+A_{v,r}$ and $-(\Pi_{v,l} - A_{v,l})$ to C;
  end if
end for
return C;

1) **Clique**: Corrections for encoding a super-node with the clique are the missing edges. Finding them requires counting unconnected node pairs.
2) **Star**: Encoding a super-node as a star requires two parts of corrections: 1) the lack of edges between the hub and spokes and 2) redundant edges among spokes. Heuristically, VoG regards the node with the highest degree as the hub, and other nodes are thus spokes.
3) **Chain**: VoG's heuristic first finds the furthest node of a random node using Breadth-First Search (BFS). Regarding the furthest node as the beginning of the chain, it uses BFS again to find the subsequent furthest node as the end of the chain. It regards the rest edges as corrections, which leads to a higher cost. We improve it by iteratively finding chains in the remaining subgraph and attaching them to the last chain.
4) **Bi-clique**: We propose a heuristic to find the largest bi-clique (Algorithm1). It first regards two nodes as the cores of the bi-clique's two parts. Other nodes are classified into two parts according to their connections.
5) **Empty**: The empty structure is the opposite of the clique. It means that there is no edge existing between any pair of nodes. Thus, non-empty cells are repre-

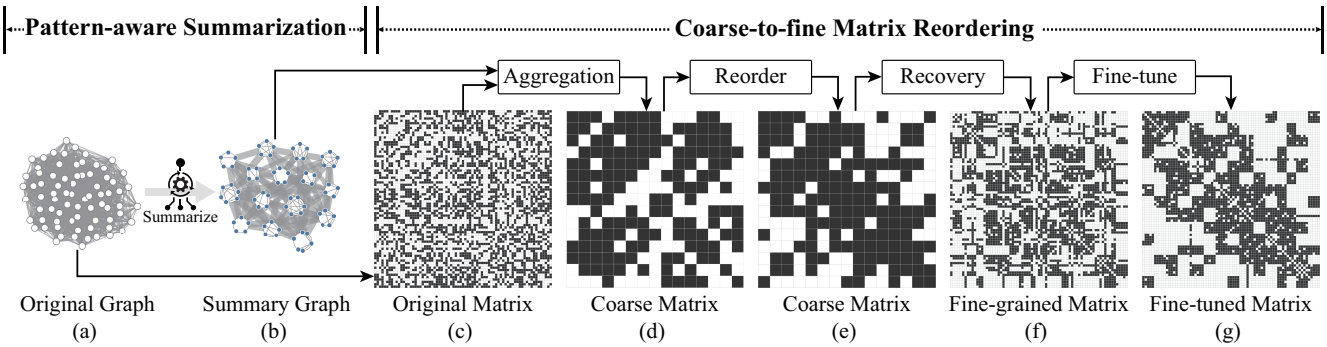


Figure 4. The pipeline of our pattern-aware matrix reordering approach. It is composed of two components. The pattern-aware summarization approach identifies patterns in (a) the original graph and generates (b) a summary graph, where each super-node in the summary graph corresponds to a pattern. The coarse-to-fine matrix reordering mechanism transforms the summary graph into the matrix representation. Given (c) the original matrix, our approach first aggregates its rows/columns into (d) a coarse matrix according to the summary graph. (e) To optimize quality metrics, we first reorder the matrix at the coarse level. (f) A fine-grained matrix is restored by reversing the aggregation phase. (g) We fine-tune the restored matrix by reordering the sub-matrix corresponding to each pattern.

sented as corrections. Forming empty patterns reduces isolated cells.

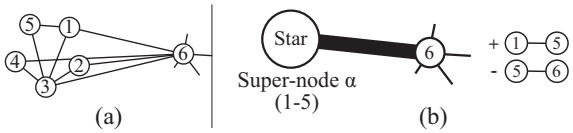


Figure 3. An example shows how our unified pattern-aware cost definition works. (a) the original subgraph; (b) the summary graph with two corrections is generated following our two-part MDL principle.

Example. Fig.3 shows an example of using our pattern-aware cost definition. Nodes 1-5 form a super-node α because they nearly formulate a star, which is one structure in our "vocabulary" Ω. By testing every structure in Ω, we finally choose a star to encode α because it only has one correction edge: $+(1,5)$. Thus, the structure cost of the super-node α is 2. Then, a super-edge is created between the super-node α and node 6 because it only costs 2 (one super-edge and one correction edge $-(5,6)$). Thus, the total cost of the subgraph is 4. The correction set $C = \{+(1,5), -(5,6)\}$ means that it requires adding edge $(1,5)$ and removing edge $(5,6)$ from the summary graph to restore the original graph.

Pattern-aware summarization. Our algorithm minimizes the total costs of all nodes, which selects the salient pattern and reduces corrections. Fig.4a-b show the input and the output of our pattern-aware summarization algorithm. Given the original graph (Fig.4a), our approach follows the optimization process of Greedy, which iteratively merges two nodes with the largest cost reduction. Finally, it outputs a summary graph (Fig.4b), where each super-node presents a structure (pattern). Seeking high performance, we implement a faster version of Greedy, named Randomized, which sacrifices some accuracy to improve the performance. Randomized randomly picks a node u to merge with its one/two-hop neighbor v with the largest cost reduction. The process is iteratively performed until the cost is no more reduced.

Time Complexity. Our pattern-aware summarization requires merging all 2-hops node pairs, resulting in a time complexity of $(O(d_{av}^3(d_{av} + logn + logd_{av})))$, where d_{av} is the average node degree). Since the Randomized method selects random nodes and their one/two-hop neighbors for merging, utilizing this approach can reduce the time complexity to $(O(d_{av}^3))$. We leave it as a future work that may be solved by: 1) using locality sensitive hashing (LSH) [10] to accelerate the neighborhood searching; 2) using parallel computing to calculate cost reductions of different node pairs simultaneously; and 3) using a progressive pattern cost computation, which avoids recalculating the cost of a super-node in each merge.

In summary, the proposed graph summarization approach unblocks the structure limitation of Greedy by introducing an extensible "vocabulary" including different types of structures. Simultaneously, our approach employs a locally optimal segmentation, which can generate shorter descriptions than those yielded by VoG.

4.2. Coarse-to-fine Matrix Reordering

To highlight identified patterns in matrices, we need to convert the identified patterns into the matrix representation while maintaining matrix qualities. Thus, we propose a coarse-to-fine matrix reordering mechanism that makes use of existing reordering algorithms to maintain matrix qualities while emphasizing patterns through a two-level architecture. It is composed of four phases (Fig.4c-g).

Phase 1: Aggregation. In order to retain patterns, we first aggregate each recognized pattern into a coarse row/column (Fig.4c-d) so that rows/columns belonging to the same pattern are not disturbed during reordering. Therefore, the original matrix is aggregated into a coarse matrix

(Fig.4d), which corresponds to the summary graph. Note that the coarse matrix is weighted since each coarse cell is composed of multiple original cells, whose weight is defined as the number of original cells.

Phase 2: Coarse Matrix Reordering. Our approach reorders the coarse matrix to optimize metrics, preserving matrix qualities (Fig.4d-e). We update the metric definitions into a weighted version to adapt to coarse matrices:

$$\begin{aligned} PR &= \sum_{u \in V} (\phi(u) - \min_{v \in \{u\} \bigcup \Gamma(u)} \phi(v)) \times w(u,v), \\ LA &= \sum_{(u,v) \in E} \lambda((u,v), \phi, G) \times w(u,v), \\ BW &= \max_{(u,v) \in E} \lambda((u,v), \phi, G) \times w(u,v), \end{aligned} \tag{12}$$

where $w(u,v)$ is the weight of cell (u,v). Because MI can handle weighted matrices, we do not modify it. During our preliminary tests, we found that minimizing LA consistently yielded superior quality measurements compared to other methods we considered, such as Optimal Leaf Ordering. This empirical observation led us to select MinLA [31] for ordering the coarse matrix, as it aligned well with our research objectives of enhancing the clarity and interpretability of the matrix patterns. We acknowledge that MinLA [31] can be computationally intensive, but its efficacy in our context justified its selection.

Phase 3: Recovery. The recovery is the inverse of the **aggregation** process (Fig.4e-f), which converts the coarse matrix into a fine-grained matrix.

Phase 4: Fine-tuning. In the fine-tuning phase (Fig.4f-g), we obtain a sub-matrix for each pattern by tailoring the corresponding rows/columns. We employ MinLA again to reorder each sub-matrix individually, based on its proven effectiveness in our initial tests. The final ordering for the entire matrix is then obtained by concatenating the node orderings in the individual sub-matrices. We would like to emphasize that the reordering algorithms utilized in Phases 2 and 4 are interchangeable, allowing for adaptability and exploration of alternative methods in future research endeavors. This flexibility is crucial as it enables the adaptation of our approach to various contexts and requirements.

5. Evaluation and Results

We evaluated our approach in three aspects. First, we compared the pattern summarization precision between our approach and VoG [22] on a synthetic dataset. Second, we compared our approach to existing matrix reordering algorithms on four quality metrics. Finally, we conducted a user study to evaluate the effectiveness of our approach in highlighting patterns, which compared the efforts of end-users in recognizing patterns using different approaches.

Implementation. All experiments were performed on a PC with an i7-9700K CPU (3.6 GHz) and 32 GiB RAM. The specific implementation and results of all experiments are available on GitHub[1].

5.1. Comparison of Pattern Summarization Precision

Our pattern-aware summarization is inspired by the concept of VoG [22]. As previously analyzed, our approach offers some advantages over VoG. To further compare the performance of our method to VoG, we evaluated the precision of pattern summarization for both approaches.

Algorithms. Our approach provides two implementations: either Greedy (GRD) and Randomized (RDM) can be utilized to minimize the total cost. Therefore, we compared both implementations (GRD and RDM) with VoG..

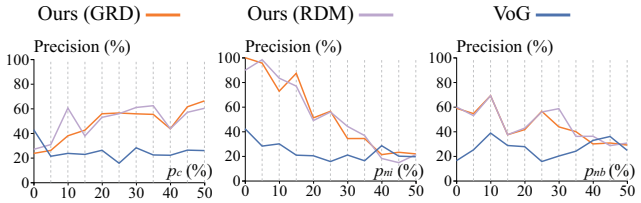
Datasets. Due to the lack of datasets with labeled patterns, we generated a synthetic dataset in accordance with [12]. It is comprised of several graphs. In each graph, the four pattern types outlined in Sec.3 were chosen. We randomly assigned five to ten nodes to each pattern. Each type of pattern was added five times to the graph. We linked two patterns with a probability p_c. For two connected patterns, edges were deleted at random with a probability p_{nb} (noise between patterns). In each pattern, existing edges would be deleted and two disconnected nodes would be connected with a probability p_{ni} (noise in patterns). We iteratively fixed two of three probabilities to 25%, while the third was increased from 0% to 50% with a 5% increment, yielding 33 graphs. We introduced noise in the patterns to simulate real-world scenarios where patterns may not be clear-cut. The noise was carefully calibrated to assess the robustness and adaptability of our method in recognizing patterns amidst uncertainties. The impact of noise on pattern recognition was meticulously analyzed to understand the thresholds beyond which the noise significantly alters the identified patterns.

Collected Data and Statistical Analysis. We computed precision as the proportion of correctly summarized nodes, where a node is considered correctly summarized if its assigned pattern in the summarization result matches the ground truth pattern. A formal mathematical definition of precision is provided in Appendix. We performed a Conover's test [14] to determine the significance between our approach and VoG. We adjusted the p-value with Holm's method [18]. We considered the introduced noise levels while computing precision to ensure that the evaluation is reflective of the method's capability to handle uncertainties and variations in the data. The significance tests were conducted with due consideration to the noise levels, providing insights into the method's reliability under different noise conditions.

Result. The results indicate that the pattern summarization precision of GRD (μ=.48, σ=.21) and RDM (μ=.49, σ=.20) is greater ($p < .01$) than VoG (μ=.21, σ=.06). As

[1] https://github.com/pattern-aware-reordering/Implementation

shown in Fig.5, our techniques behave better than VoG on most configurations. Whereas, when the between-patterns noise p_{nb} and the within-patterns noise p_{ni} are relatively high (equivalent to 50%), our techniques resemble VoG. Because the introduced noise transforms the original patterns into other patterns or even noise, neither approach can correctly recognize them.



(a) p_{ni}=25%, p_{nb}=25% (b) p_c=25%, p_{nb}=25% (c) p_{ni}=25%, p_c=25%

Figure 5. Comparison of the precision of pattern summarization between GRD, RDM, and VoG [22], while varying the connection probability p_c, the within-patterns noise p_{ni}, and the between-patterns noise p_{nb} respectively. (a) varies p_c from 0 to 50% with p_{ni} and p_{nb} fixed at 25%; (b) varies p_{ni} from 0 to 50% with p_c and p_{nb} fixed at 25%; (c) varies p_{nb} from 0 to 50% with p_c and p_{ni} fixed at 25%.

Dataset	$\|V\|$	$\|E\|$	$d(G)$	Description
chesapeake	39	170	0.229	road network
everglades	69	880	0.375	ecology network
lesmis	77	254	0.087	characters network
radoslaw	167	3,250	0.235	email network
jazz	198	2,742	0.141	collaboration network
visbrazil	222	336	0.014	tree-like network
sch	242	7,882	0.270	student interaction
econ-wm2	257	2,375	0.072	economic network
netscience	379	914	0.013	collaboration network
dwt_419	419	1,572	0.018	plannar structure
Caltech36	769	16,656	0.056	social network
asoiaf	796	2,823	0.009	characters network
petster-hamster	921	4,032	0.010	social network
price_1000	1,000	999	0.002	tree-like network
bn-mouse	1,029	1,559	0.003	brain network
wiki_talk_br	1,049	2,330	0.004	Wikipedia messages
wiki_edit_eu	1,135	2,585	0.004	Wikipedia edition
bio-grid-mouse	1,450	1,636	0.002	biological network
bio-grid-plant	1,717	3,098	0.002	biological network
bn-fly	1,781	8,911	0.006	brain network

Table 3. Overview of the dataset

5.2. Comparison of Matrix Reordering Measures

This section compared our approach to seven other reordering algorithms based on four existing quality metrics.

Datasets. We conducted experiments on various datasets, selected for their diverse sizes and characteristics. The basic statistics and description of each dataset can be found in Tab.3.

Algorithms. We compared our approach to seven widely employed matrix reordering algorithms. Six of them were chosen according to the six categories summarized in [6]; the collection-aware leaf ordering includes an algorithm that specifically optimized for Moran's I, which was recently proposed and has not been surveyed by the taxonomy. We re-implemented these seven algorithms, and the implementation details could be found on our GitHub page.

1) **MinLA** [31] (graph-theoretic approach);
2) **Biclustering** [6] (biclustering approach);
3) **Evolutionary** Reordering [28] (heuristic approach);
4) **Leaf Ordering** (**LO**, Robinsonian approach) [3];
5) **MDS** [33] (dimension reduction approach);
6) **Rank-Two** [11, 6] (spectral approach);
7) **Leaf Ordering using Moran's I** (**LO-δ_I**) [35, 1].

Metrics. For the resulting matrix ordering, we selected four existing metrics (*LA*, *PR*, *BW*, and *MI*). All metrics were scaled into $[0,1]$ and positively correlated to the quality. As indicators for further analysis, we employed the 95% confidence interval (CI) for each quality metric. We used a Friedman test to examine any significant differences among algorithms. We conducted comprehensive pair-wise comparisons based on the Conover's test [14] for any significant difference that emerged from the Friedman test. The p-value was adjusted using Holm's method [18].

Quantitative Results . As shown in Fig.6, we compare our two approaches (GRD and RDM) to the other seven baseline algorithms with the significance level of $p < .05$. In 39 (red cells with $p < .05$) out of ($2\ ours \times 7\ baselines = 56$) comparisons, our techniques significantly outperform baselines. The blue cells primarily correspond to the three baselines: MinLA, LO and LO-δ_I. Our two approaches and minLA perform similarly in terms of the *LA/PR/BW*, as evidenced by the similarity of histogram and confidence intervals shown in Fig.6b-d. However, our GRD significantly outperforms minLA on *MI* in Fig.6a. Likewise, our GRD shows comparable performance to LO and LO-δ_I on *MI*, but demonstrates superior performance on *LA* and *BW*. Our approach cannot outperform algorithms designed to minimize certain metrics within their territory (e.g., LO-δ_I for *MI* and MinLA for *LA*), but it still finds the position: 1) no algorithm is significantly better than our approach on any metric, and 2) no algorithm is superior to our approach on all metrics. It suggests that our pattern-aware matrix reordering approach shows substantial improvement on most metrics without negative influence, successfully preserving or even enhancing matrix qualities. Details on runtime performance are provided in Appendix.

Qualitative results We perform a qualitative analysis by highlighting patterns on matrices. Our GRD generates a summary graph for a given graph dataset in the first step (Fig.4b), which consists of supernodes and superlinks. These supernodes correspond to the five "vocabularies" mentioned in Sec.4.1, which are also reflected in the patterns shown in Tab.1. By leveraging this graph summa-

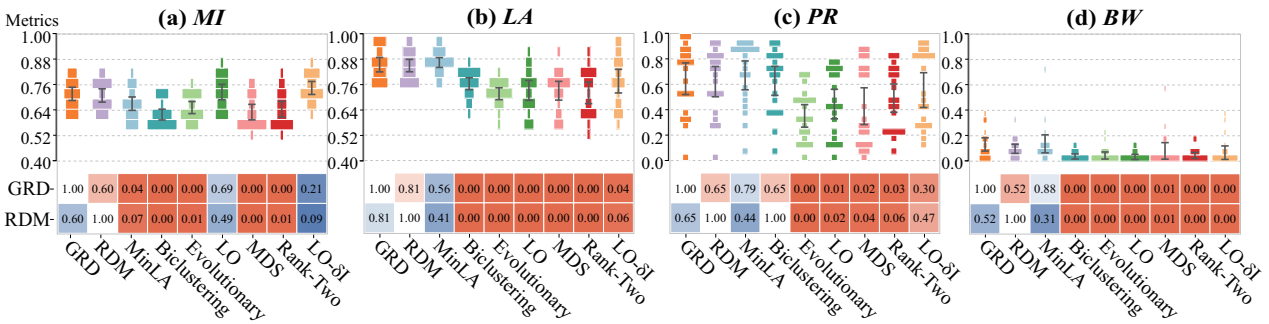


Figure 6. (a-d) are comparisons of four metrics (*MI*, *LA*, *PR*, and *BW*) between our techniques and baselines. The interval plots above show the 95% confidence interval (CI) of metric values. Histograms show distributions of metric values on individual datasets. The 2×9 pixel maps below demonstrate the p-value of the Conover's test. A smaller p-value means the difference is more significant. Blue cells mean baselines are better, while red cells mean ours are better. Cells with deeper color suggest greater significance.

rization, we are able to highlight patterns in all matrices. Specifically, if other reordering algorithms produce results that contain the same pattern we have identified, the corresponding area of the matrix representing the pattern will be colored accordingly. We highlight patterns in the matrices of the seven reordering algorithms across all the datasets, based on the graph summarization of GRD and RDM respectively. All of the result figures are available on https://osf.io/gb3wq/?view_only=3d3c4ac16d0444ad9815cfa22c91fd9f. Visualizations for this analysis are provided in Appendix (see Figure 2). The result demonstrates the comparison of our GRD with the three most competitive baselines (MinLA, LO and LO-δ_I) on two datasets (chessapeake and lesmis). First, our GRD effectively preserves the identified patterns, which are highlighted along the diagonal. This can be attributed to our approach of transforming the summary graph into a coarse level matrix, which has demonstrated superior performance in pattern preservation. Second, clear evidence indicates that other methods exhibit a significantly lower number of highlighted patterns compared to our GRD. This only means that the three baselines fail to identify all the patterns we summarized. Although all three baselines optimize their quality metric and are able to show certain visual patterns, their interpretability falls short when compared to our GRD. Third, as illustrated in the top row of the visualization, our GRD, LO, and LO-δ_I are capable of displaying the chain structure (blue cells). However, it is noteworthy that our GRD demonstrates a chain structure that is more consistent with human visual perception. Moreover, our GRD excels in highlighting the bi-cliques (yellow cells).

5.3. User Study

The purpose of this user study is to evaluate our approach's ability in pattern highlighting. Detailed settings of the user study are introduced as follows.

Algorithms. According to the findings of our experiment on metric comparison, MinLA [31] and LO-δ_I [35] are the baselines that are most effective on optimizing metrics. Thus, in the user study, we used them as baselines to compare the effectiveness of pattern highlighting with the proposed techniques (GRD and RDM).

Participants. We recruited 15 participants (4 females and 11 males; aged 21-25). They are all researchers or students with a background in computer science. 5 participants reported that they have practical experience with graph visualization and matrix visualization. The others said they had taken the necessary courses to equip them with basic visualization knowledge. Each participant received $15 before the user study.

Stimuli. Due to the lack of labeled datasets, we generated a synthetic dataset for this user study using the same procedure in Sec.5.1. Each synthetic graph consists of a star, a clique, a bi-clique, a chain, an empty, and a small amount of noise. All patterns have eight nodes. We generated five graphs by randomly configuring $p_c \in [0\%, 20\%]$, $p_{ni} \in [0\%, 20\%]$, and $p_{nb} \in [80\%, 100\%]$. Generated graphs were reordered by four algorithms (MinLA, LO-δ_I, GRD, and RDM). In summary, the user study used $5 \times 4 = 20$ matrices. Each of them has $5 \times 8 = 40$ rows and columns.

Task. Participants were asked to identify and label patterns from 20 matrices through a web system, as shown in Fig.7. Specifically, they needed to recognize patterns, select each recognized pattern by brushing the cells or point-and-click, and label the cells with a pattern type (clique, bi-clique, chain, and star). They were asked to execute their tasks as quickly and accurately as they could. Note that labeling more or fewer cells is considered incorrect.

Procedure. The study began by introducing the purpose, background knowledge of the patterns, the system interface, and the study task. Then, we provided participants with a practice matrix. Before moving to the next phase, we made sure that participants understand how to perform the task and how to use the interface. Subsequently, participants

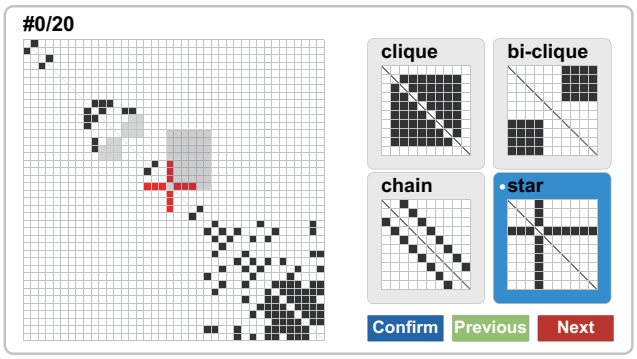


Figure 7. The interface of the user study. The number in the left top corner indicates the progress. The left matrix is the stimuli shown to participants. Participants can label patterns by brushing cells. Brushed cells are colored in red and labeled cells are colored in gray. Brushing one cell would also select its mirrored counterpart since the matrix is symmetric. After brushing one pattern, participants can choose one of four pattern types and click the blue button for confirmation. Participants can continue to the next activity by pressing the red button, or return to the previous task by clicking the green one.

can start the formal study. To counterbalance learning effects, the order of 20 matrices was randomly assigned to each participant. The study was conducted on a PC with a mouse, keyboard, and 27-inch display (with a resolution of 3840×2160). The formal study lasted around fifteen minutes.

Hypotheses. We hypothesized that our techniques would highlight more patterns. Consequently, with our techniques, participants would recognize patterns more efficiently than with the baselines. Due to the fact that our two techniques performed comparably in previous studies, we hypothesized that their effectiveness would make little difference.

H1: GRD is more effective than both MinLA and LO-δ_I for participants to perform pattern identification tasks.

H2: RDM is more effective than both MinLA and LO-δ_I for participants to perform pattern identification tasks.

H3: The effectiveness of GRD and RDM makes no difference.

Collected Data and Statistical Analysis. We recorded the pattern labeling results and their task completion time. The task of pattern identification is regarded as a multi-class classification task. We obtained the ground truth data through the generation step, which recorded which cells belong to which pattern. Subsequently, we updated the true positive, false positive, and false negative counts based on the user's labeling of each cell in the ground truth. Using this information, we calculated precision and recall measures for each pattern and aggregated them to determine the user's accuracy for the entire matrix. Consequently, the effectiveness of different algorithms can be measured by the micro F1-score (equals to the micro precision and micro recall) and the average labeling time per pattern. We employed the Conover's test to conduct pairwise comparisons with a significance level of .05. All tests were adjusted with Holm's method [18].

Quantitative Results. As shown in Fig.8a, our two techniques (GRD and RDM) enable participants to identify patterns more accurately than baselines (MinLA and LO-δ_I). There is no statistically significant difference ($p = .06 > .05$) between our two techniques. The average labeling time also reveals that our techniques accelerate participants' ability to identify patterns over baselines ($p < .05$). Our two techniques do not show a significant difference in terms of labeling time (Fig.8b).

In general, all results support **H1-3**. To further analyze the collected data, we break down the results by different patterns (Fig.8c-d). We observe that all algorithms highlight cliques similarly, but perform differently on other patterns. We observe that MinLA excels at highlighting cliques, as its design aligns bi-cliques along the diagonal, which naturally supports cliques. However, this disrupts bi-cliques, where our techniques outperform MinLA significantly. Similarly, LO-δ_I performs poorly on stars and chains, due to its optimization target (MI) being less responsive to these patterns (0.70 for stars, 0.40 for chains). Our techniques, on the other hand, achieve superior precision and recall on these patterns, maintaining global pattern integrity.

6. Discussion

We discussed our study from the following four aspects.

Significance. After decades of research, numerous reordering algorithms have been developed to improve the quality of matrix visualization. Nonetheless, highlighting multiple patterns in matrices is still challenging. By incorporating a pattern-aware graph summarization algorithm, our approach opens up new opportunities for highlighting patterns. On the one hand, customized patterns can be introduced into graph summarization to facilitate a variety of tasks. On the other hand, our approach preserves the unique characteristics of the employed reordering algorithms. For instance, our approach reduces users' pattern recognition efforts while retaining MinLA's capability to minimize linear arrangement.

Generalizability. The pattern-aware summarization is applicable to more than only matrices. It can be expanded to improve the readability of graph layouts. The pattern-highlighting property can also be applied to node-link diagrams. Using pattern-aware summarization can also benefit scatterplots. Similarities can be drawn between the proximity of data points in a scatterplot and the proximity of nodes in a graph. To construct a pattern-aware summary for scatterplots and node-link diagrams, it is necessary to investigate new pattern discovery heuristics.

Limitations. We list three additional limitations of our

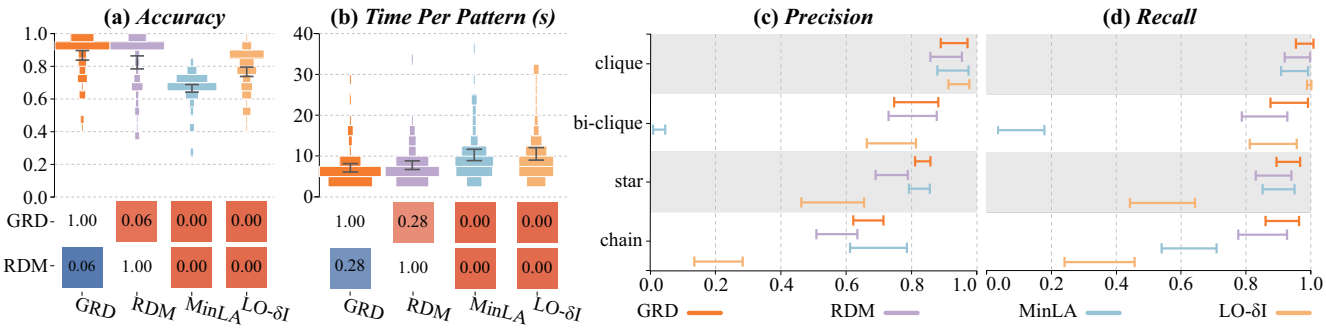


Figure 8. Comparisons of participants' effort on recognizing patterns with four algorithms. (a) and (b) compare the accuracy and the average time for labeling per pattern of different algorithms. The interval plots above show the 95% confidence interval (CI) of metric values. Histograms show distributions of metric values on individual datasets. The 2×4 pixel maps below demonstrate the p-value of the Conover's test. A smaller p-value means the difference is more significant. Blue cells mean baselines are better, while red cells mean ours are better. Cells with deeper color suggest greater significance. (c) and (d) show the 95% confidence interval (CI) of different algorithms' precision and recall on the four pattern types.

approach, which could be addressed in the future. First, patterns have no clear demarcations and may exhibit smooth transitions from one to another. Enforcing rigid and clear pattern boundaries could potentially mask the presence of other plausible patterns. It is noteworthy that our current method employs a hard segmentation approach, where a node is definitively assigned to a single pattern. However, a node may inherently belong to multiple patterns in reality. To address this, we plan to assign a probability to each node or edge, indicating its likelihood of belonging to a certain pattern. It might soften the "stiffness" of different patterns. Second, supporting custom patterns requires designing new heuristics, which is challenging for non-expert users. It may be alleviated by the community contributing more pattern discovery heuristics to our open-source implementation. Third, our approach stops when merging two nodes reduces no cost, which may fall into a local minimum without generating larger patterns. We plan to employ the concept of simulated annealing to escape from saddle points, which tolerates a degree of cost rise.

Future Work. We identified several potential extensions of our work. First, our approach can be extended to address the non-binary matrix reordering problem. In Sec.4.2, we provided several metrics' variants to assess the qualities of weighted matrices. It could be the initial step of extending existing reordering algorithms to non-binary matrices, particularly algorithms aimed at optimizing metrics. In addition, reordering non-binary matrices requires surveying patterns on non-binary cells, namely weighted patterns. New heuristics for searching weighted patterns should be incorporated into our approach to facilitate their highlighting. Second, our cost definition can be extended to measure the sensitivity of the proposed techniques to different patterns. Magnostics [5] evaluates the global pattern sensitivity but is unable to measure local patterns. Our pattern-aware summarization algorithm can merge adjacent rows/columns to form local patterns. The cost definition is able to quantify the local pattern salience. Utilizing our cost definition in the future will allow us to measure the sensitivity of existing techniques to different patterns. Third, in the current process of obtaining graph summarization, we have assigned equal weight to both the summary and corrections. However, the potential impact of varying weight on pattern emphasis warrants further exploration in future research. For example, when analysts prioritize pattern discovery over correction identification, manual adjustment of weights can be employed to achieve the desired effect. Furthermore, different base patterns can be assigned distinct weights, allowing for greater flexibility in user interaction. As our approach successfully preserves identified patterns, users can selectively adjust the encoding costs of various patterns based on their analysis tasks, ultimately yielding their desired matrix visualizations.

7. Conclusion

We design and evaluate a summarization-based pattern-aware matrix reordering approach, which generates pattern-preserving matrix orderings. It has two stages: the data-level stage produces a pattern-aware summary, and the visual-level stage reorders the matrix using existing algorithms in a pattern-preserving mechanism. To evaluate our approach, we conducted three quantitative experiments: a comparison of pattern summarization precision, a comparison of matrix quality metrics, and a user study to examine the accuracy of end-user pattern recognition. The results demonstrated that our approach achieved better performance than existing algorithms comprehensively.

References

[1] Reorder.js. https://github.com/jdfekete/reorder.js/. Accessed: 2022-08-19. 9

[2] J. E. Atkins, E. G. Boman, and B. Hendrickson. A spectral algorithm for seriation and the consecutive ones problem. *SIAM Journal on Computing*, 28(1):297–310, 1998. 2

[3] Z. Bar-Joseph, D. K. Gifford, and T. S. Jaakkola. Fast optimal leaf ordering for hierarchical clustering. In *Proceedings of International Conference on Intelligent Systems for Molecular Biology*, pages 22–29, 2001. 3, 9

[4] L. Becchetti, P. Boldi, C. Castillo, and A. Gionis. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *Proceedings of ACM SIGKDD*, pages 16–24. ACM, 2008. 1

[5] M. Behrisch, B. Bach, M. Hund, M. Delz, L. von Rüden, J. Fekete, and T. Schreck. Magnostics: Image-based search of interesting matrix views for guided network exploration. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):31–40, 2017. 3, 4, 12

[6] M. Behrisch, B. Bach, N. H. Riche, T. Schreck, and J. Fekete. Matrix reordering methods for table and network visualization. *Computer Graphics Forum*, 35(3):693–716, 2016. 3, 9

[7] M. Behrisch, T. Schreck, and H. Pfister. GUIRO: user-guided matrix reordering. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):184–194, 2020. 1

[8] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Computer Networks*, 29(8-13):1157–1166, 1997. 1

[9] M. J. Brusco, H.-F. Köhn, and S. Stahl. Heuristic implementation of dynamic programming for matrix permutation problems in combinatorial data analysis. *Psychometrika*, 73(3):503–522, 2008. 3

[10] G. Y. Chan, P. Xu, Z. Dai, and L. Ren. ViBr: Visualizing bipartite relations at scale with the minimum description length principle. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):321–330, 2019. 2, 7

[11] C.-H. Chen. Generalized association plots: Information visualization via iteratively generated correlation matrices. *Statistica Sinica*, pages 7–29, 2002. 3, 9

[12] W. Chen, F. Guo, D. Han, J. Pan, X. Nie, J. Xia, and X. Zhang. Structure-based suggestive exploration: A new approach for effective exploration of large networks. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):555–565, jan 2019. 8

[13] Y. Cheng and G. M. Church. Biclustering of expression data. In *Proceedings of International Conference on Intelligent Systems for Molecular Biology*, pages 93–103. AAAI, 2000. 3

[14] W. J. Conover. *Practical Nonparametric Statistics*, volume 350. john wiley & sons, 1999. 8, 9

[15] N. E. Gibbs, W. G. Poole, and P. K. Stockmeyer. An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM Journal on Numerical Analysis*, 13(2):236–250, 1976. 4

[16] D. Harel and Y. Koren. Graph drawing by high-dimensional embedding. *Journal of Graph Algorithms and Applications*, 8(2):195–214, 2004. 2, 3

[17] L. H. Harper. Optimal assignments of numbers to vertices. *Journal of the Society for Industrial and Applied Mathematics*, 12(1), 1964. 4

[18] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979. 8, 9, 11

[19] L. Hubert. Some applications of graph theory and related non-metric techniques to problems of approximate seriation: The case of symmetric proximity measures. *British Journal of Mathematical and Statistical Psychology*, 27(2):133–153, 1974. 2, 3

[20] A. V. Knyazev. Toward the optimal preconditioned Eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM Journal on Scientific Computing*, 23(2):517–541, 2001. 2

[21] Y. Koren and D. Harel. A multi-scale algorithm for the linear arrangement problem. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 296–309, 2002. 4

[22] D. Koutra, U. Kang, J. Vreeken, and C. Faloutsos. Summarizing and understanding large graphs. *Statistical Analysis and Data Mining*, 8(3):183–202, 2015. 2, 4, 5, 8, 9

[23] I. Liiv. Seriation and matrix reordering methods: An historical overview. *Statistical Analysis and Data Mining*, 3(2):70–91, 2010. 3

[24] Y. Lim, U. Kang, and C. Faloutsos. SlashBurn: Graph compression and mining beyond caveman communities. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):3077–3089, 2014. 5, 6

[25] Y. Liu, T. Safavi, A. Dighe, and D. Koutra. Graph summarization methods and applications: A survey. *ACM Computing Surveys*, 51(3):62:1–62:34, 2018. 2

[26] M. Lozano, A. Duarte, F. Gortázar, and R. Martí. Variable neighborhood search with ejection chains for the antibandwidth problem. *Journal of Heuristics*, 18(6):919–938, 2012. 3

[27] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. In *Proceedings of the ACM SIGMOD*, pages 419–432. ACM, 2008. 2, 4, 5

[28] S. Niermann. Optimizing the ordering of tables with evolutionary computation. *The American Statistician*, 59(1):41–46, 2005. 3, 9

[29] M. Okoe, R. Jianu, and S. Kobourov. Node-link or adjacency matrices: Old question, new insights. *IEEE Transactions on Visualization and Computer Graphics*, 25(10):2940–2952, 2019. 2

[30] C. Pich. *Applications of multidimensional scaling to graph drawing*. PhD thesis, University of Konstanz, 2009. 3

[31] E. Rodriguez-Tello, J. Hao, and J. Torres-Jimenez. An effective two-stage simulated annealing algorithm for the minimum linear arrangement problem. *Computers and Operations Research*, 35(10):3331–3346, 2008. 1, 8, 9, 10

[32] R. Sakai, R. Winand, T. Verbeiren, A. V. Moere, and J. Aerts. dendsort: modular leaf ordering methods for dendrogram representations in R. *F1000Research*, 3, 2014. 3

[33] I. Spence and J. Graef. The determination of the underlying dimensionality of an empirically obtained matrix of proximities. *Multivariate Behavioral Research*, 9(3):331–341, 1974. 9

[34] S. Suri and S. Vassilvitskii. Counting triangles and the curse of the last reducer. In *Proceedings of International Conference on World Wide Web*, pages 607–614. ACM, 2011. 1

[35] N. van Beusekom, W. Meulemans, and B. Speckmann. Simultaneous matrix orderings for graph collections. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):1–10, 2022. 1, 4, 9, 10

[36] S. You, L. Gao, Y. Hua, M. Zhu, and M. Li. A visualization system for dynamic protein structure and amino acid network. In *Cooperative Design, Visualization, and Engineering*, pages 290–297. Springer, 2017. 1

[37] F. Zeng, R. Tao, Y. Yang, and T. Xie. How social communications influence advertising perception and response in online communities? *Frontiers in Psychology*, 8, 2017. 1