GUIRO: User-Guided Matrix Reordering

Michael Behrisch, Tobias Schreck, and Hanspeter Pfister

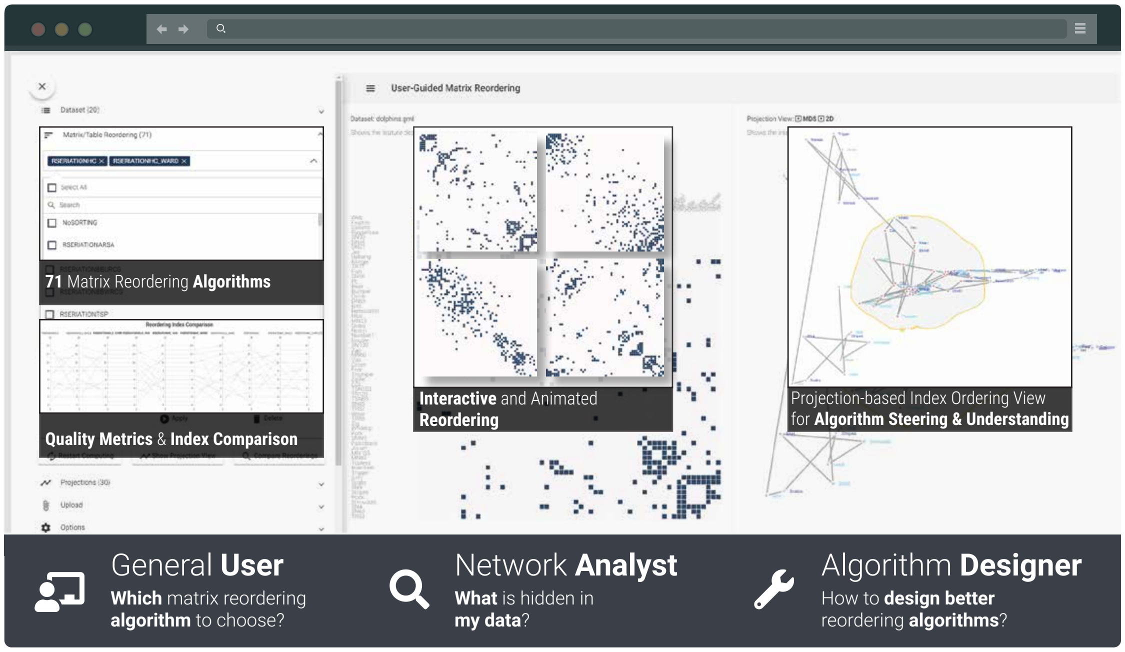

71 Matrix Reordering Algorithms
Quality Metrics & Index Comparison
Interactive and Animated Reordering
Projection-based Index Ordering View for Algorithm Steering & Understanding
General User
Which matrix reordering algorithm to choose?
Network Analyst
What is hidden in my data?
Algorithm Designer
How to design better reordering algorithms?
User-Guided Matrix Reordering

Fig. 1: With GUIRO we address three essential matrix analysis questions: (1) Which matrix reordering algorithm produces useful results? (2) Can we steer these algorithms to support analytical tasks? (3) How to compare reorderings quantitatively & qualitatively?

Abstract— Matrix representations are one of the main established and empirically proven to be effective visualization techniques for relational (or network) data. However, matrices—similar to node-link diagrams—are most effective if their layout reveals the underlying data topology. Given the many developed algorithms, a practical problem arises: *"Which matrix reordering algorithm should I choose for my dataset at hand?"* To make matters worse, different reordering algorithms applied to the same dataset may let significantly different visual matrix patterns emerge. This leads to the question of trustworthiness and explainability of these fully automated, often heuristic, black-box processes. We present GUIRO, a Visual Analytics system that helps novices, network analysts, and algorithm designers to open the black-box. Users can investigate the usefulness and expressiveness of 70 accessible matrix reordering algorithms. For network analysts, we introduce a novel model space representation and two interaction techniques for a user-guided reordering of rows or columns, and especially groups thereof (submatrix reordering). These novel techniques contribute to the understanding of the global and local dataset topology. We support algorithm designers by giving them access to 16 reordering quality metrics and visual exploration means for comparing reordering implementations on a row/column permutation level. We evaluated GUIRO in a guided explorative user study with 12 subjects, a case study demonstrating its usefulness in a real-world scenario, and through an expert study gathering feedback on our design decisions. We found that our proposed methods help even inexperienced users to understand matrix patterns and allow a user-guided steering of reordering algorithms. GUIRO helps to increase the transparency of matrix reordering algorithms, thus helping a broad range of users to get a better insight into the complex reordering process, in turn supporting data and reordering algorithm insights.

Index Terms—Visual Analytics, matrix, black-box algorithms, seriation, ordering, sorting, steerable algorithm, interaction, 2D projection

1 Introduction

Relational datasets, commonly referred to as graph datasets, are getting more and more important in the research and commercial sectors. Social networks, modeling individuals and their interactions, are typically used for researching sociological and socio-dynamic phenomena, but also find their commercial application in the planning and evaluation of effective advertisement campaigns [64, 85]. Biological networks help us understand the intertwined relationships between the degree of efficiency and environmental influences for drugs or depicts complex phylogenetic trees [11, 41]. Telecommunication networks are combining relational data and geographic considerations to drive the development of our ubiquitous wired and wireless networks. A wide range of application fields for network analysis, in general, are known in the area. The `NetworkRepository.com` [66] alone groups its more than 4,500 datasets into 27 categories, representing also less commonly associated fields, such as ChemInformatics, brain-, power network analysis, or ecology and economy networks research.

Generally, two visualization techniques for large-scale graph data have been proven effective: Node-link diagrams and matrix visualiza-

- *Hanspeter Pfister and Michael Behrisch are with the School of Engineering and Applied Sciences, Harvard University, United States. E-mail: {pfister, behrisch}@seas.harvard.edu.*
- *Tobias Schreck is with the Graz University of Technology, Austria. E-mail: tobias.schreck@cgv.tugraz.at.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

tions [25,57]. In this work, we will focus on matrix visualizations. Their outstanding property is that they allow us to perceive Overview+Detail in one plot. They follow a simple, yet powerful, construction mechanism: Items, i.e., vertices, are assigned to rows and columns and links between them, i.e., edges, are encoded by filling the cell at the intersection of the linking items' row and column index. This effective row/column layout circumvents overplotting and edge cluttering problems a n n nd—more importantly—can directly target overview questions, such as «Is the network big or small?» (few or many rows/columns) or «Is the network globally sparse or dense?» (few or many filled cells) or «Does the network represent a small-world phenomenon?» (globally sparse, but locally dense) [25]. For more detailed analytical questions, though, matrix visualization requires a *good* reordering of rows and columns to extract meaningful information, such as the presence of topological structures. Fig. 1 (middle) shows the same data (matrix), using a different order for rows and columns. All reordered plots let us perceive more or less distinct *interpretable* visual patterns. The problem is, however, that for a matrix with n rows and m columns, $(n! \times m!)/2$ possible non-redundant row/column permutations exist. Of this factorial number of permutations, we can assume that very few plots actually deliver useful information.

When using matrix visualizations for graph data analysis typically three central problems exist:

(1) It is not transparent, **which matrix reordering algorithm to choose.** From previous work [7, 48, 80], we learn that not only a wide range of algorithm choices exist, but also that their (non-apparent) algorithmic design decisions can have a tremendous impact on the visual result.
(2) It is not transparent, **which local structures are hidden in a globally optimized layout.** Most matrix reordering algorithms are designed to grasp solely the underlying top-level topology. Even worse, they are mostly designed "only" for graph partitioning tasks. While clusterings are important only few approaches are able to handle and demonstrate nested patterns.
(3) It is not transparent, **how algorithms are working.** Matrix reordering algorithms are mainly black-box algorithms; the user has no control over results beyond the choice and parametrization of quality criteria. Due to the large search space, these algorithms resort to heuristics and may return local optima. In addition, their complexity is such that multiple runs with different parametrization can be very time-consuming.

In this work, we present `GUIRO`, short for user-GUIded matrix Re-Ordering, a Visual Analytics system that helps users with different visualization literacy to work with matrix plots. Users can experiment with their own data through our web-based interface. We have implemented 70 matrix reordering algorithms; to our knowledge the biggest collection of directly accessible algorithms for this purpose. `GUIRO`'s primary purpose, however, is to help network analysts to retrieve and analyze local structures in global matrix reorderings. For this purpose, our technique centers on a similarity-preserving, 2D projection of the matrix (Index Ordering Visualization) in which analysts can invoke two novel interaction techniques for steering the row/column reordering process: (1) analysts can change the item-wise order through a drag-and-drop gesture and (2) they can reorder (one or multiple) selected item groups according to one or multiple distinct reordering algorithms. Combining (1) and (2) allows generating *composite* matrix reorderings, thus taking advantage of the benefits from its constituent sub-algorithms. Lastly, `GUIRO` can be used by algorithm designers, who want to compare qualitatively and—more importantly—quantitatively their newly developed algorithm to the state-of-the-art. We allow computing 16 quality metrics and visualize the pair-wise (dis)similarities for row/column index lists.

With `GUIRO`, our fundamental goal is to increase the transparency of so-called black box algorithms. We hope that showing comparative visualizations, the ability to apply reordering algorithms locally, and the freedom to intervene in the row/column reordering process, helps a broad range of users to get a better insight into the complex matrix reordering process, in turn leading to novel data and algorithm insights.

2 User, Goals & Usage Scenarios

Matrix visualizations are widely employed in research and industrial settings by a diverse set of users with different needs. Interestingly, neither the survey literature on the network/graph analysis [7, 56, 76], nor the central work on user tasks in this domain [25, 46, 62] explicitly categorizes or describes the users' capabilities. To fill this gap, we extrapolated three common user roles, their motivation, analysis focus, and typical usage scenarios.

2.1 Novices & Scholars

Although the design of matrices is straightforward, the interpretation of its visual features—and ultimately visual patterns—is not. Novices are therefore mostly concerned with acquiring a level of visualization literacy that enables them to explain the emerging visual patterns. In comparison, novices are generally not interested in understanding the algorithmic circumstances leading to the results, but rather try to reflect on the general idea of *"placing similar rows/columns next to each other."* Let us investigate this user group from an education perspective:

T1 **Compare result usefulness.** Matrix reordering algorithms are designed to promote certain visual patterns [7]. However, if the dataset simply does not contain the expected topology, these algorithms will likely produce cluttered results. In a typical exploration scenario the user will invoke a set of algorithms to compare the distinct outcomes visually.
T2 **Demonstrate patterns variability**. The high-level analysis tasks for graph data strongly differ from non-relational data analysis tasks. We will elaborate this point in Sect. 3.1. In an instructional setting, the scholar will raise the awareness that different algorithms will produce distinct dissimilar visual patterns.
T3 **Reveal the heuristic nature** of (many) algorithms. Due to the factorial increase of the search space many reordering algorithms are heuristic, sometimes even greedy algorithms. A comprehensive instructional session will illustrate this fact by applying several times the same algorithm on the same dataset, resulting in different visual results.

Combining the challenges of **T1**, **T2**, and **T3** leads to the quintessential question *"Which matrix reordering algorithm to choose?"*. In `GUIRO`, we help novices in answering these questions through an interactive comparative "Algorithm Chooser", described in Sect. 4.1.

2.2 Network Analyst

Network Analysts make up the most prevalent role of matrix visualization users. According to Kandogan [40] and Kandel [39] their interest in algorithmic details can be varying. However, we can safely assume that their primary goal is to understand, assess, and present the dataset's top-level data topology. A more *comprehensive* network analysis though will also examine the interaction between *local* patterns.

T4 **Global pattern analysis**. An appropriate matrix reordering algorithm will reflect the general data topology allowing the analyst to answer canonical questions, regarding connectivity and partitioning [46]. However, as described in [7] or [80], these algorithms can also fail to produce interpretable results, even if the dataset contains explainable structure. What is needed is a *structure-preserving meta visualization* that enables network analysts to estimate how much structure a dataset contains.
T5 **Nested pattern analysis**. The idealistic view—depicted in Fig. 2—does not hold for real-world scenarios. To give a practical example, biological networks often reflect complex and even nested visual pattern relationships [41, 47]. Almost all reordering algorithms, however, are designed for global pattern retrieval and interactive visualizations for applying these algorithms in a local context are mostly unexplored.
T6 **Detect (local) pattern relationships**. As mentioned before, visual patterns are not necessarily disjunct but form complex interactions. A network analyst has to distill this information, by retrieving, e.g., who is the central actor connecting two subgroups in a social network. Having found this influencing person the subsequent question would be *who* is the primary contact point *within* the groups to distribute the information further.

T7 Adapt visual results to emphasize analytic importance. Presenting the above complex relationships is quintessential for a successful network analyst. However, based on our experience, these analysts interestingly do not lean necessarily on one automatic matrix reordering result but sacrifice the global reordering on subparts to emphasize a point on other parts.

In GUIRO, we address the usage scenarios of structure "prescreening" with a projection-based index view (Sect. 4.2) and two interactions for a local pattern exploration, organization, and presentation (Sect. 5).

2.3 Algorithm Designer

Algorithm designers have a different set of tasks, compared to the network analyst and novice users. Their daily work is centered around quantitative evaluation schemes, justifiable algorithm design decisions, and stress tests.

T8 Inspect Quality Metrics. Algorithm designers are concerned, not only with the visual performance of their algorithms, but also quantitative quality metrics. Comparing a newly designed algorithm against the state-of-the-art is necessary for an objective comparative evaluation.

T9 Comparing/Weighing up of Design Decisions. Matrix reordering algorithms can be tweaked on two fronts: (a) the process of exploring the permutations space and (b) the notion of similarity between rows/columns. Giving the designer (visual) means to reflect on the impact of algorithm design decisions will lead to better, more stable algorithms for matrix reordering.

T10 What-if Analysis. Black-box algorithms work most of the time. Finding out, however, what went wrong in the error cases is nontrivial. One classic way of doing this kind of "debugging" is to exchange the input with structurally varying datasets and compare the ordering results.

In GUIRO, we give the algorithm designer various means to address their typical usage scenarios (see: Sect. 6). A performance metric view (**T8**) and the index-to-index comparison component (**T10**) enables a (visual) juxtaposition of matrix reordering results. Futhermore, we can overlay multiple reordering results in our projection-based index visualization, as we will describe in Sect. 4.2.

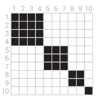
3 Matrices and Visual Patterns in Matrices

An adjacency matrix $\mathbb{M} = (n \times n)$ is a two-dimensional vector with $n = |N|$ with N being the vertices in an undirected network. A matrix *cell* c_{ij}; $0 \leq i, j < n$ denotes the existence of an edge between vertices i and j. In simplest case, $c_{ij} = 1$ if i and j are connected, otherwise $c_{ij} = 0$. In many real-world datasets, c_{ij} models data attributes, such as strength (e.g., amount of messages exchanged between friends in a social network), or relation type (e.g., highway vs. interstate road in a transportation network) resulting in a weighted adjacency matrix. *Matrix visualizations* reflect this layout scheme by applying color, texture, or glyphs to each to cell. Rows and columns can be freely rearranged, also called a row/column *permutation* π, without changing the underlying data. A "good" reordering allows us to perceive Overview and Detail in one plot. Statistically, however, $(n! \times n!)/2$ non-redundant permutations exists of which only few expose the *topological structure*.

3.1 Visual Patterns in Matrices

A *visual matrix pattern* is a perceivable structure in the matrix visualization revealing information about the underlying graph topology. Some work has been conducted [7, 53, 80] to collect the central visual patterns in matrix plots, along with their graph-theoretic interpretations. For the sake of simplicity, let us analyze the simple graph shown in Fig. 2 while generalizing our findings to a taxonomy of interpretable visual patterns for matrices.

The graph in Fig. 2 contains the complete/fully connected subgraph (clique) (v1 ↻ v2 ↻ v3). In our matrix representation, the red coherent rectangular block ▨ on the main diagonal represents this clique.



Generally, a **Block-diagonal pattern** consists of one or multiple blocks (at least 2 × 2 cells) on the main diagonal of the matrix. Note, that the main diagonal is the center matrix line, where $i = j$; $0 \leq i, j < n$, and is quintessential for a correct interpretation of matrix patterns. The density of the blocks lets us distinguish cliques (fully colored), strongly connected components or clusters (some uncolored matrix cells) and loosely connected components (less color density). Connected components are often the central visual pattern. It finds applications, e.g., in social network analysis representing groups of mutual friends or in bioinformatics groups of genes with their related expression characteristics.

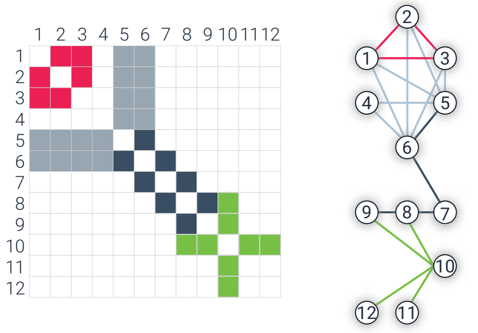
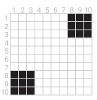


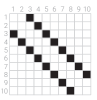
Fig. 2: An adjacency matrix (left) and a Node-Link diagram (right) allow interpreting a range of visual patterns. A network analysts' tasks is to (a) to find the row/column ordering that represents most topological structures best and (b) to represent the results effectively to the audience. Figure adapted from [54].

The next visual pattern in our graph connects v5 ↻ v1, v5 ↻ v2, v5 ↻ v3, v5 ↻ v4, and analogous v6 ↻ v1, v6 ↻ v2, v6 ↻ v3, v6 ↻ v4. The visually dense off-diagonal gray-blue rectangles ▨ show this connectivity pattern in matrices.



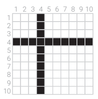
Albeit being similar in its visual appearance to block-diagonal forms, **Off-Diagonal Block patterns** do not overlap with the main diagonal; sometimes they are even at the corners of the matrix. Off-diagonal blocks refer to bi-cliques consisting of two sets of vertices where each vertex from the one set is connected to each vertex of the second set. Bi-cliques model for example, website visits and their user groups, or persons having visited a range of cities.

Next, our graph shows one four node path connecting v5 ↻ v6, v6 ↻ v7, v7 ↻ v8,and v8 ↻ v9, with its matrix representation being the two bands or lines parallel to the main diagonal ▨.



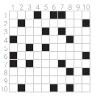
In undirected networks, this **Band pattern** consists of lines parallel to the matrix diagonal. These lines are mirrored and do not touch or cross the main diagonal. Bands indicate paths through the network, i.e., sequence of links from one to another vertex. Paths can represent a possible way of information flow through a computer network or a sequence of reactions in a biological networks.

Lastly, we find a fan-like connection of the vertex v10 ↻ v8, v10 ↻ v9, v10 ↻ v11,and v10 ↻ v11 in green. In the matrix, this motif will be strongly visible by a so-called **Star/Line Pattern** ▨.

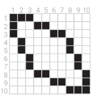


Generally, star or also called line patterns in matrices consist of one horizontal and one connected vertical line with or without discontinuities. This pattern represents a star graph motif, i.e., a highly connected vertex in the network, such as a famous person in a friendship network, a transportation hub, or a catalyst for reactions in a biological network.

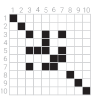
Our walk-through should not be considered as exhaustive, but rather as a general methodology for retrieving matrix patterns. The large variety of visual patterns results from the combination of the aforementioned base patterns and so-called anti-patterns: Two established anti-patterns exist in the literature [7, 80]:



The most obvious Anti-pattern is **Noise**. We can describe it as a distribution of black cells without inherent visual structure. The more noise a matrix contains, the less information it shows. Interestingly, noise has two interpretations: (1) There is no apparent structure in the underlying graph topology and (2) the pattern extractor, i.e., the matrix reordering algorithm, was not able to grasp this underlying topology [10, 53].



The second Anti-pattern is the **Bandwidth Anti-pattern**. Bandwidth patterns are solely an artifact of algorithms, which are examining the permutation search space in a breadth-first search, such as the graph-based matrix reordering algorithm Cuthill-McKee and its predecessors [15,24]. This breadth-first search finds its visual manifestation in a formed envelope, which does not add interpretable information unless the ordering *within* the formed envelope contains other patterns.



In this work, we want to establish a third general Anti-pattern: The **Noise-Cluster Anti-pattern**. Many matrix reordering algorithms (see also: Sect. 3.2) are designed to support partitioning tasks, i.e., their implementation tries (desperately) to reflect groupings/cluster in the data. However, two artifacts frequently clutter matrix visualizations: (1) the algorithm groups rows/columns next to each that are contradicting our humans' *gestalt-law* expectations and (2) the algorithm separates rows/columns that our humans' perception system would rather group together. Both visual artifacts can be traced back to the algorithm's implementation, e.g., thresholding decisions. Unfortunately, these perceptual and related cognitive biases aspect find little attention in the literature, yet they contribute significantly to our interpretability problems for matrices.

3.2 Matrix Reordering (or Seriation)

Matrix visualizations scale well with the size of the input data, but their usefulness heavily depends on a suitable reordering of rows/ columns [7, 53, 81, 83]. We can formalize the process reordering of an undirected network G, in accordance to our state-of-the-art survey on matrix reordering approaches [7], as computing one permutation $\pi \in S$ that maximizes or minimizes a quality criteria $q(\pi, G)$, such that: $\arg\min_{\pi \in S} q(\pi, G)$ For example, q can compute the sum of distances $d(x, y)$ between vertices according to the order π; The equation would find $\pi \in S$ that minimizes this sum. While a variety of automated reordering methods have been developed, we want to focus our attention in this work on the underlying mechanisms and different quality criteria that guide the process as a whole.

Automated Matrix Reordering: Quality metrics for matrix reordering regularly include (a) the optimization of quality criteria, e.g., the row/ column dissimilarity, and (b) the emergence of visible structures from the matrix. Regarding (a), several quality criteria have been suggested that are partially congruent with the graph drawing domain; c.f., Díaz et al. [17]. The prominent quality criteria for matrix reordering are bandwidth and profile optimization, column/row gradient measures, such presented by Hubert et al. [36], the Hamilton path length [13], or Stress measures [55]. The *linear arrangement (LA)* [43] is the de facto standard for comparing matrix reordering algorithms quantitatively. It relates all node-to-node distances (or edge weights) to their index-to-index distances in a given reordering. Minimizing the LA causes similar vertices to be placed close to each other. Finding a global optimum for this problem is computationally hard, but efficient approximations, such as the Multi-Scale approach [43] exist. Regarding (b), other reordering strategies may aim at exposing interpretable visual structures (c.f., Sect. 3.1). These effect ordering techniques [22] make use of *perception-driven* quality criteria, such as Anti-Robinson Events/Deviations [14, 65, 71], Anti-Robinson arrangements [61], the barycenter heuristics [50], or McCormick et al.'s "Measure of Effectiveness" [51, 52], which can be used to find locally dense areas off the diagonal (c.f., Off-Diagonal Block pattern). Note that `GUIRO` allows analysts to compare 16 quality scores from all mentioned categories.

Interactive Matrix Reordering: Automatic approaches are making (non-)deterministic, sometimes greedy, and most importantly *hard-coded decisions* at many points of their process, i.e., how to start the reordering sequence or how to traverse and prune the permutation search space. To address the shortcomings of certain algorithms, interactive reordering techniques have been suggested. Bertifier [60], TableLens [63], InfoZoom [70], or Siirtola and Mäkinen's "Reorderable Matrix" [69] on the one end of the spectrum, let the user drag-and-drop rows and columns manually, providing comparable functionality to Bertin's mechanical reordering device. On the other end of the spectrum, semi-assisted (steering) approaches, such as MatrixExplorer [32], COMBat [73], or PermutMatrix [13] let the user apply distinct matrix reorderings while helping between context switches. The first approach to a *semi-automatic* matrix reordering was presented by Henry and Fekete in [32], where a traveling salesman-based reordering algorithm can be iteratively (re-)applied on a selected subset of rows/columns.

`GUIRO` unifies many of the mentioned interactive and semi-interactive approaches in one user interface and lets the user experience, steer, and evaluate matrix reordering systematically. Please note that this section is supposed to give the reader a tutorial into the topic. For didactical reasons, we discuss `GUIRO` in the context of its applied techniques in the dedicated Related Work Section 8.

4 User-Guided Matrix Reordering

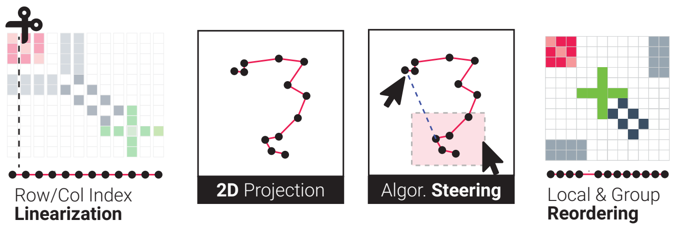


Fig. 3: Processing pipeline for our user-steerable matrix reordering approach: the rows, resp. columns, of a matrix are interpreted as high-dimensional (HD) vectors (1st image) and projected into a 2D space (2nd image) forming a set of vertices. Similar HD vectors are projected to similar 2D positions. A matrix reordering result is depicted by a path acyclically connecting all vertices. Selecting vertex groups allows a local application of reordering algorithms on submatrices (3rd image). The path can be manually modified, such that locally optimized submatrices or single vertices can be placed next to another (4th image).

In this section, we provide an overview of `GUIRO`. We illustrate how the steerable reordering process can increase user understanding of the matrix plot, as well as the measurable quality of the reordering.

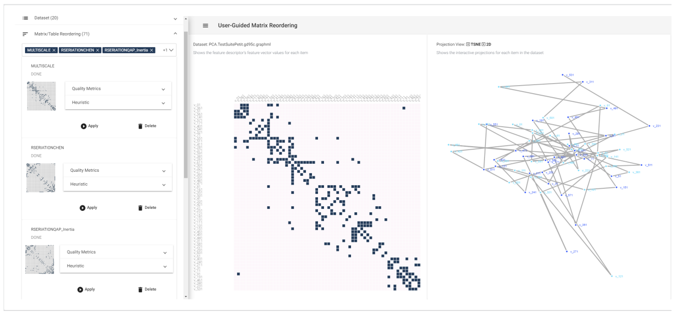


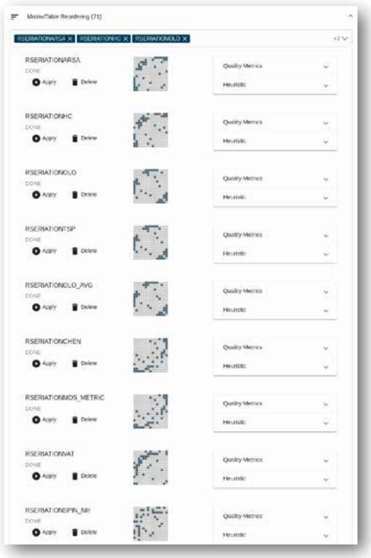
Fig. 4: A simple example demonstrating the basic design of `GUIRO`. The screenshot shows the Petit Test Suite G95c dataset with three reordering options (left). The projection space (right) depicts that distinct groups of rows/columns can be formed.

Implementation: We built `GUIRO` on top of a microservice infrastructure that encapsulates processing algorithms (matrix reordering, projections, and statistics) independent of their implementation language and makes them accessible through a `REST` API. The matrix reordering server, for example, bridges between Java (for the API) and its underlying algorithm implementations in R, Java, and C++. For all ten projection algorithms described in Sect. 4.2, we use the `Smile` (Statistical Machine Intelligence and Learning Engine) Java library [2]. Inter-service communication is implemented, for now, in REST. A central database stores all computed projections and matrix reordering results. `GUIRO`, as the visible client, is implemented in TypeScript. The interactive visualizations are written in D3.js. The client accesses all microservices through an API gateway, which enables horizontal scalability, service discovery, load distribution, health checks, and manages the authentication centrally. Source code, a demo instance, and a descriptive webpage will be available at `http://bit.ly/matrixreordering`.

4.1 Comparing Matrix Reordering Results

A multitude of matrix reordering algorithms have been developed with distinct design goals in mind. Consequently, the question arises which algorithm to choose (**T1**)? We claim that the research developments in the quality metrics for information visualization field [8] will soon allow us to approach this question with Visual Analytics means. For now, however, we have to rely on the user to make an informed decision.

Algorithm Chooser with 71 Choices (**T1, T2**)



In `GUIRO`, we let the user choose from 70 matrix reordering algorithms (+1 identity sort). All implementations are accessible through a central server, whose purpose is to proxy our own 15 Java implementations, 42 implementations from the outstanding R package `seriation` by Hahsler, Buchta, and Hornik [28, 29], nine R package `biclustering` implementations [37], two R package `corrplot` algorithms [79], and three algorithms from the C++ library `Boost` [1]. All references for the accessible implementations can be found in the appendix. Note that, nine reordering algorithms are reoccurring with district implementation facets, making them interesting study objects for algorithm designers (**T9**, **T10**).
We depict the results of all applied algorithms first in a preview manner, next to their quantitative quality metric scores; see Figure above. The list allows us (a) to experience the algorithms' visual quality and usefulness, i.e., produced amount of clutter (**T1**), and (b) demonstrates the algorithm's pattern variability given the same dataset (**T2**). For instance, in the screenshot above we can see that `OLO` shows a Star-Line form, but `ARSA` tries to capture a Block pattern. After having found an algorithm result, we can apply it to the main interactive matrix view to begin our in-depth analysis process (Sect. 4.3). Interestingly, it is even for experts challenging to distinguish deterministic from non-deterministic algorithms (**T3**). To aid this use case, we allow to reapply the same algorithm several times and see the results in a comparative display.

4.2 Representing Matrix Reorderings in the Projection Space

Our index ordering visualization relies on a projection of the matrix $\mathbb{M}$ onto the 2D plane, where the user can interact with the matrix elements by moving or grouping them (**T5**, **T6**, **T7**). We realize the projection by regarding the matrix rows or columns as a set of n distinct n-dimensional vectors with $n = |N|$ with N being the nodes in our network. Similar concepts have already been shown for 2D/3D graph layouts [30], dynamic network analysis [74], or topological graph comparison [12, 44]. In order to examine the typological structure of $\mathbb{M}$ (**T4**), we can allow all projection techniques that find a lower dimensional representation $\mathbf{y} = (y_1, \ldots, y_k)^T$ for each vector $\mathbf{x} = (x_1, \ldots, x_l)^T$ with $k \leq l$, such that $\mathbf{y}$ *preserves* the *HD row/column similarity*. In other words, that put visually similar rows/columns close to each other in the projection space. Note, for symmetric matrices (undirected graphs) rows and columns can be used interchangeably. We could indeed extend `GUIRO` for non-symmetric matrices, where rows and columns would need to be considered distinctively. In the current version of the system, we decided against this functionality, because it adds significant complexity to the interpretation of the projection space patterns.

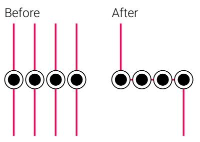
The many existing projection algorithms all have their own advantages and disadvantages. Linear projection techniques are typically fast, but only preserve linear structures. Non-linear projection techniques are typically slow, but can also take non-linear structures into account. In order to be projection-agnostic, `GUIRO` implements ten projection techniques: `PCA, MDS, t-SNE, IsoMap, Isotonic MDS, LLE, PPCA, KPCA, SammomMapping, LaplacianEigenMap`.

To depict one or multiple matrix reordering results, we connect the 2D projection points in this display through directed edges representing a given ordering of the rows or columns. The resulting path has $n - 1$ edges representing the matrix's row/column linearization. Its start and end point map to the first and last row or column in the matrix view. The path represents a model visualization, whose appearance (see: Sect. 4.3) allows us to scrutinize the effectiveness of a matrix reordering result.

4.3 Interpreting Reordering Results in the Projection Space

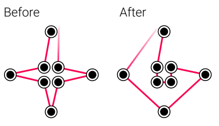
The 2D index visualization allows the user to perceive the reordering in terms of the similarity of rows or columns and gives access to interactive matrix reordering operations. We structure the interaction around visual patterns emerging from this space. These patterns prove to be a beneficial starting point for improving the matrix reordering. They visually represent a mismatch between close projection points, i.e., similar column-/row vectors of the matrix, and long connection edges, i.e., the sequential placement of dissimilar column-/row vectors by the reordering algorithm.

Vertex Groups



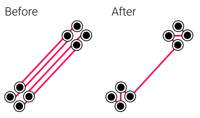
Vertex Groups: Incoherently connected vertex groups are a natural starting point for improving matrix reorderings. In case of an initial ineffective grouping, as illustrated left, a locally dense region is highly connected with vertices outside of the group. This corresponds to a matrix reordering, in which dissimilar vectors are arranged sequentially to improve the (global) optimization function.

Star Edges



Star Edges: The star edge pattern is a special case of the vertex group pattern, which can see often in matrix reordering results. As our illustration depicts, a dense group of vertices is connected to multiple satellite vertices. In order to optimize this pattern, a user or reordering algorithm would need to connect all satellite vertices linearly with one edge to the dense central vertex group.

Long Parallel Edges



Multiple parallel edges are another indication for optimization potential. Our illustration (left) represents the case when similar vertices are not connected appropriately to their neighborhood, i.e., are not arranged correctly into "local" subsequences. A fix would first consider the local groupings and then connect the two groups.

Discussion: The index ordering visualization projects similar rows/-columns in close proximity. Consequently, an intermediate result will appear visually cluttered (e.g., Fig. 4), due to the many, long, and unstructured index ordering edges. The users' task is to retrieve the above mentioned projection space patterns and invoke the local (group) reordering solutions described in the following subsections. Users will be confronted with a combination of patterns (see also the accompanying video), thus making this view to a typical Visual Analytics component, in which user understanding, prioritization and planning are of paramount importance for the success.

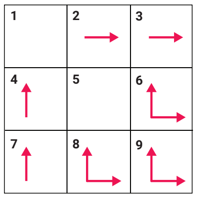
5 Interacting with Matrix Reorderings in the Projection Space

Once we retrieved the aforementioned structures in the 2D projection space we can start optimizing the reordering with two key interactions: (1) Invoke an *automated local reordering* on a selected group of vertices, and (**T4**, **T5**) (2) *Rearrange* vertices or groups by replacing one or more existing edges (**T6**, **T7**).

In the case of an automated local reordering, the user selects a group of vertices with a lasso selection tool and invokes thereby the matrix reordering algorithm of choice. The algorithm operates only on the selected matrix elements, effectively reordering a subregion of the selected matrix. This reordering approach can work recursively (**T5**), as user-selected groups can be organized into subgroups and sorted independently (see Sect. 5.1 for details). The second key interaction mechanism is the direct modification of the edge set. We enable users to reconnect edges in the projected matrix interactively. This is primarily used to modify the global order of locally reordered groups of matrix elements (**T4**, **T6**).

5.1 Local Group Reordering

We enable a submatrix reordering either with the help of an automated algorithm or a user-steered manual reordering decision (**T4**, **T5**). Obviously, this operation can only be applied if all row-/column projection vectors are in subsequent order thus forming a coherent submatrix. To establish this valid selection, we propose a so-called `PartSort` algorithm to *prepare* the matrix for a local reordering.



`PartSort` takes as an input a user selection, the current index permutation, and a reordering algorithm to be applied on the submatrix. It then linearizes the selected projection vertices sequentially *without* a notion of reordering quality. It ensures that the remaining 2D projection vertices, i.e., before and after the user selection, are logically connected by edges. After this initial step, all available matrix reordering algorithms can be applied on the selected submatrix as if the matrix stands on its own. For our example case, depicted in our matrix reassembling illustration (left), the user selection corresponds to the submatrix 5, which gets reordered independently. This reordering, i.e., the new ordering of the selection vertices, needs to be propagated into the full matrix. While the submatrix 1 remains untouched by the local change, all other submatrices have to be adapted. Depending on a submatrix's position relative to submatrix 5 either a horizontal (column-wise) or vertical (row-wise) or horizontal and vertical adaptations have to be applied. Submatrices 2 and 3, for example, will only be affected in the horizontal direction. The same applies to submatrices 4 and 7 in a vertical direction. Therefore, a column-/row swapping helps to reassemble the matrices correctly. Adaptations to submatrices 6, 8 and 9 must be applied in both the horizontal and vertical directions.

5.2 Edge Replacement Reordering Strategies

The second user intervention is the so-called edge replacement, depicted in Fig. 5. It rearranges a single row/column, respectively group of rows/columns relative to each other. This can be useful for detecting item-to-item, item-to-group, or group-to-group relationships (**T6**). Another use case is that an analyst may wish to emphasize a finding (**T7**), e.g., by "pulling" two influencing entities right next to each other.

A formally correct matrix reordering is present in the projection space if there exists precisely one acyclic path that connects all vertices. Whenever a user draws a new edge, its insertion will introduce cycles and ambiguities resulting from the temporarily two potential paths. Accordingly, every user intervention inevitably requires reestablishing our formal requirement. We need to deal with two edge manipulation cases depending on whether the user wants to put a vertex (or group) left/before or right/after another vertex (group):

1. **Forward Edge Modifications:** Forward edges are those edges whose linear reordering's start index (index of the inserted edge's outgoing vertex) is smaller than the end index (index of the inserted edge's incoming vertex). To make an example: if the user wants to place the 1st row-/column vector next to the 5th row-/column vector.
2. **Backward Edge Modifications:** Backward edges are those edges whose start index is larger than the end index. For example, if the user wants to place the 5th row-/column vector next to the 1st row-/column vector.

As a proof of concept and a point of further research, we implemented two different reordering strategies. We found that both edge modification types can ultimately have a significant impact on the matrix's overall reordering quality with respect to the linear arrangement.

Independent of the two cases we are reestablishing one acyclic path connecting all vertices by inserting so-called *"reconstruction edges"*. In our *forward edge modification algorithm* we split the vertex set into

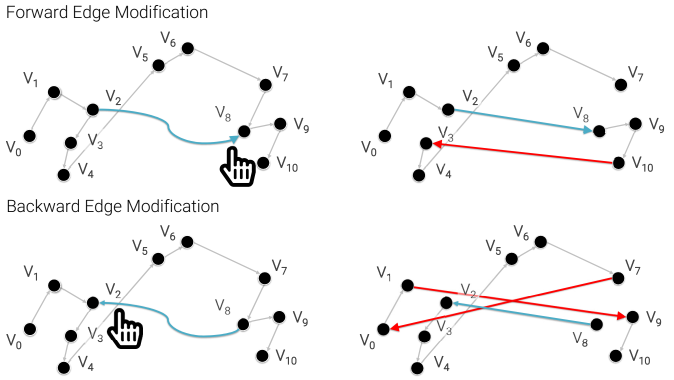


Fig. 5: Two matrix reordering strategies are possible: A forward edge modification (top) and a backward edge modification (bottom). Depending on the type of reordering strategy, different approaches to reestablish a formally correct linear arrangement have to be applied.

three disjunct subsets, as depicted in the upper part of Fig. 5. The first subset contains all vertices with an index before the insertion edge's starting point (v2) in the Set$_1$ = [v0, v1]. The second subset contains all edges with an index larger than the starting point and smaller than the endpoint (v8) in the Set$_2$ = [v3, v4, v5, v6, v7]. Finally, the third subset contains all edges with an index after the endpoint in the Set$_3$ = [v9, v10]. The new vertex sequence is calculated as follows: First, we start the permutation vector by retaining Set$_1$ = [v0, v1] and then add the insertion edge's start point and end point [v0, v1, **v2, v8**]. We append subsequently Set$_3$ = [v0, v1, v2, v8, **v9, v10**] to the list. Lastly, a new *reconstruction edge* is added to connect all items in Set$_2$, leading to the final linear reordering of [v0, v1, v2, v8, v9, v10, **v3, v4, v5, v6, v7**].

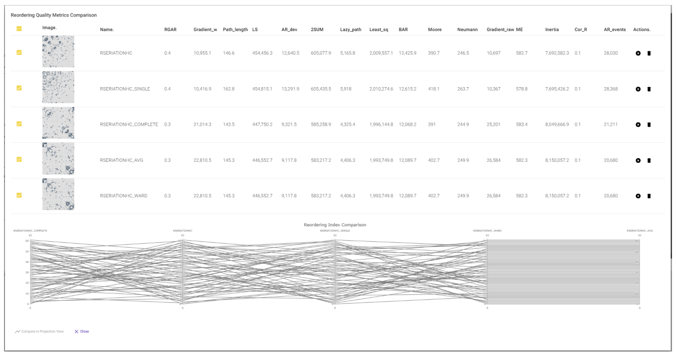
The *backward edge modification algorithm* is slightly more complicated and illustrated in the lower part of Fig. 5. We also split the vertex set into three subsets. In this case, Set$_1$ contains all vertices after the insertion edge's endpoint Set$_1$ = [v3, v4, v5, v6, v7]. In the second set Set$_2$, we collect the ordering from the initial linearization start until the insertion edge's endpoint Set$_2$ = [v0, v1]. Lastly, we store all vertices after the insertion edge's start point in Set$_3$ = [v9, v10]. The new permutation vector begins with the insertion edge sequence [**v8, v2**], followed by Set$_1$ = [v8, v2, **v3, v4, v5, v6, v7**] and a reconstruction edge. Subsequently, we append Set$_2$ = [v8, v2, v3, v4, v5, v6, v7, **v0, v1**] and another reconstruction edge to connect the missing vertices in Set$_3$. The final result is [v8, v2, v3, v4, v5, v6, v7, v0, v1, **v9, v10**].

A wide range of other implementations for the edge modifications are possible. While the algorithms presented above are just one heuristic, they perform well in establishing a better linear arrangement in our experiments. However, more sophisticated algorithms that minimize the reconstruction edges' length while retaining most of the input linear ordering are imaginable. These algorithms need to judge whether the user should be *overruled* for the sake of a better linear arrangement.

6 Comparing Performance Metrics for Reordering Algorithms

Another use case for `GUIRO` is the quantitative quality assessment. Algorithm designers constantly need to compare their designs to the state-of-the-art, similar approaches in the subfield, or algorithm modifications from the same algorithm family (**T8**).

We allow comparing 16 distinct quality metrics for matrix reordering algorithms, such as (Anti-)Robinson measures, graph-theoretic, stress or correlation quality metrics. Interestingly, we found that Wilkinson's idea to compute the similarity of matrix reordering results with the help of a Spearman correlation-based metric [80, p. 532] can be intuitively depicted with a parallel-coordinate plot inspired visualization. For example, in Fig. 6 we directly see that the hierarchical clustering algorithms `HC_Ward` and `HC_AVG` produce in the identical row/column list independent of their linkage method (**T9**, **T10**).



Reordering Quality Metrics Comparison
Reordering Index Comparison
RSERIATIONHC_COMPLETE
RSERIATIONHC
RSERIATIONHC_SINGLE
RSERIATIONHC_WARD
RSERIATIONHC_AVG
Compare in Projection View
Close

	Image.	Name.	RGAR	Gradient_w	Path_length	LS	AR_dev	2SUM	Lazy_path	Least_sq	BAR	Moore	Neumann	Gradient_raw	ME	Inertia	Cor_R	AR_events	Actions.
✓		RSERIATIONHC	0.4	10,955.1	146.6	454,456.3	12,640.5	605,077.9	5,165.8	2,009,557.1	12,425.9	390.7	246.5	10,697	582.7	7,692,582.3	0.1	28,030	
✓		RSERIATIONHC_SINGLE	0.4	10,416.9	162.8	454,815.1	13,291.9	605,435.5	5,918	2,010,274.6	12,615.2	418.1	263.7	10,367	578.8	7,695,426.2	0.1	28,368	
✓		RSERIATIONHC_COMPLETE	0.3	21,014.3	143.5	447,750.2	9,321.5	585,258.9	4,325.4	1,996,144.8	12,068.2	391	244.9	25,201	583.4	8,049,666.9	0.1	21,211	
✓		RSERIATIONHC_AVG	0.3	22,810.5	145.3	446,552.7	9,117.8	583,217.2	4,406.3	1,993,749.8	12,089.7	402.7	249.9	26,584	582.3	8,150,057.2	0.1	20,680	
✓		RSERIATIONHC_WARD	0.3	22,810.5	145.3	446,552.7	9,117.8	583,217.2	4,406.3	1,993,749.8	12,089.7	402.7	249.9	26,584	582.3	8,150,057.2	0.1	20,680	

Fig. 6: GUIRO allows algorithm designers to compare 16 quality metrics (**T8**), as well as the row/column index (dis-)similarity. We see that the hierarchical clustering algorithms HC_Ward and HC_AVG produce in the identical index list independent of their linkage method (**T9**, **T10**)

7 Evaluation

We designed GUIRO to support users with a wide range of visualization and algorithm literacy to explore matrix reordering as a topic. As discussed in the Background Section 3.2 and the Related Work Sect. 8, either automatic or interactive approaches were presented to tackle subparts of this problem. We claim that no single technique or software fulfills all our outlined requirements. Based on our experience though we also believe that it is possible to solve most tasks by switching back and forth between systems, such as presented in [13, 32, 60, 73].

Following the "Patterns for Visualization Evaluation" [20], we designed our evaluation procedure to reflect the three persona (novices, network analysts, algorithm designers) introduced in Sect. 2: (1) In a qualitative user study, we examine the usability and understandability of the system as a whole for novices and general users, incorporating all tasks, but put our focus on (**T1-3**). (2) We present a case study evaluation inspired by a typical network analysis scenario, demonstrating our proposed interaction techniques on a real-world dataset (**T4-7**). (3) We conducted an expert study with one of the leading researchers in the field of matrix reordering algorithm design and gathered valuable comments and thoughts on the design decisions, computational- and work flow aspects (all tasks, but particularly **T4-7** and **T8-10**).

7.1 User Study

Experiment Design: After a careful consideration, we decided yet against a formal comparative study between different systems, since this would only allow us to compare that systems like PermutMatrix (7), MatrixExplorer (2), or GUIRO (71) have a distinct ranges of applicable matrix reordering algorithms. MatrixExplorer, however, allows us to interact with the matrix (**T4** - **T6**) and retrieve consensus among algorithms (**T9**). While we value the latter in GUIRO only as a minor contribution, a comparative evaluation to our proposed interactive reordering would have been interesting, but practically challenging. Henry and Fekete use a coordinated matrix and a node-link diagram view, where the user should *"[...] explore and discover with matrices, and present with node-link diagrams"* [32, p. 682]. In a within-subject design we would confront the user with two similar setups, but entirely distinct purposes. In GUIRO, the user explores and discovers in the *index ordering view*, and then (semi-) automatically *applies* the gained knowledge to the *matrix plot*. A between-subject design is also inexpressive because it would just prove that different users have different understandings of what and when a pattern is visible. Given the complexity, we doubt that a fair contrasting juxtaposition is feasible in a between-subject or even within-subject study design. Consequently, we decided that it is more prudent to examine GUIRO's usefulness in a qualitative user study focused on assessing the usability and understandability for novice and less experienced users.

Participants: We recruited 12 paid participants (8 female) between 22 and 34 years old. All are researchers or students with a background in computer science, design, or applied mathematics. 58.3% indicated that they have no experience with heatmaps or matrix visualizations, two participants used matrices sometimes and three used them occasionally.

Tasks and Procedure: Prior to the actual study, we screened our participants for vision loss, gave them a short tutorial on how to interpret matrix patterns (similar to Sect. 3.1), and advised the participants to verbalize their thoughts and reasoning for their action (Think-Aloud study). The introduction phase took about 10min per participant and ended with our pre-study survey (see: Appendix).

During the study, which was subdivided into three consecutive analysis tasks (each approx. 10min), we captured the screen content, took notes on the participants' analysis approach, and answered conceptual or interaction questions. We based the tasks on our usage scenarios mentioned in Sect. 2. To reduce the mental complexity, we introduced the visual interface gradually. For user task 1 (UT1), we only showed the matrix and the reordering algorithm chooser. We asked the participants to explore the (socialnet_karate) dataset and visually assess and compare ten matrix reordering results with respect usefulness (**T1**), variability (**T2**), heuristic nature (**T3**). For UT2 and UT3, we added the projection view but restrained the subjects to a subset of the reordering and projection algorithm choices. In UT2, the subjects interpreted (**T4**), analyzed (**T5,6**) and interacted (**T7**) with a MDS-calculated projection space given a larger dataset with 62 entities (socialnet_dolphines [49]). In UT3, we enabled all algorithm and datasets choices, but told them to choose at least all five OLO algorithm variants. Their open-ended task was to find similar results visually in the matrix view (**T4-6**) and particularly in the quality metric view with its index (dis-)similarity plot (**T8-10**).

After each task, we asked the participants to assess the system's usability aspects and their subjective analysis confidence. We used the standardized *Single Ease Question (SEQ)* questionnaire [67]. After the study, we used the *NASA-TLX Workload Assessment Questionnaire* [31] to assess the subjective workload and collected feedback on the estimated usefulness for our anticipated user groups.Lastly, we concluded every session by asking open questions to collect detailed feedback and suggestions for improvements.

Results and Discussion: For most questions we used a 5-point Likert-scale rating ranging from strongly disagree (1) to strongly agree (5). We analyzed the survey results using a graphic qualitative method (Box-Plot) and report here the median ($\widetilde{x}$) and interquartile range (IQR). In order to compare the resulting ordinal distributions according to their population mean ranks, we applied the non-parametric Wilcoxon signed rank test. Our survey questions and answers can be found along with the analysis journal in the Appendix.

Overall, the majority of participants stated that the technique is useful ($\widetilde{x}$=4, IQR=1), well-integrated into the workflow ($\widetilde{x}$=4, IQR=1.25), and generally easy to understand ($\widetilde{x}$=4, IQR=0.5). When asked about the applicability for our target users the participants agreed that the system is useful for novices and scholars ($\widetilde{x}$=4, IQR=0.5). But, when taking the SEQ1 as an additional indicator and compare both distributions the Wilcoxon test reports only a weak statistical significance ($p < .0977$). The anticipated usefulness for expert users, however, can be statistically validated ($\widetilde{x}$=4, IQR=1). Comparing SEQ2 and this usefulness question reveals a strong statistical significance ($p < .0039$). The same holds for algorithm designers ($\widetilde{x}$=5, IQR=1). Comparing SEQ3 and this usefulness question reveals a strong statistical significance ($p < .0176$).

We asked the subjects to evaluate their confidence when interpreting the patterns in the projection view ($\widetilde{x}$=3, IQR=1) and whether they found it helpful for their analysis ($\widetilde{x}$=5, IQR=1). We claim that this is a typical result. Although the novice user might be challenged by understanding high-dimensional spaces and lower-dimensional projection techniques, they can still use them as a conceptual data analysis tool.

During our study we took notes on the participants' approaches and findings. When asked about the most useful result in UT1, many (75%) participants were judging OLO and CHEN [14] to be interpretable, while FPC and AOE (both spectral algorithms) were mostly (83%) described as less useful. As outlined in [7], this mostly congruent understanding of result usefulness can, right now, not be expressed with quality metric and remains an open research challenge. In UT2, all participants found the two satellite clusters and reordered them with our PartSort algorithm. The local algorithm choices were distinct and

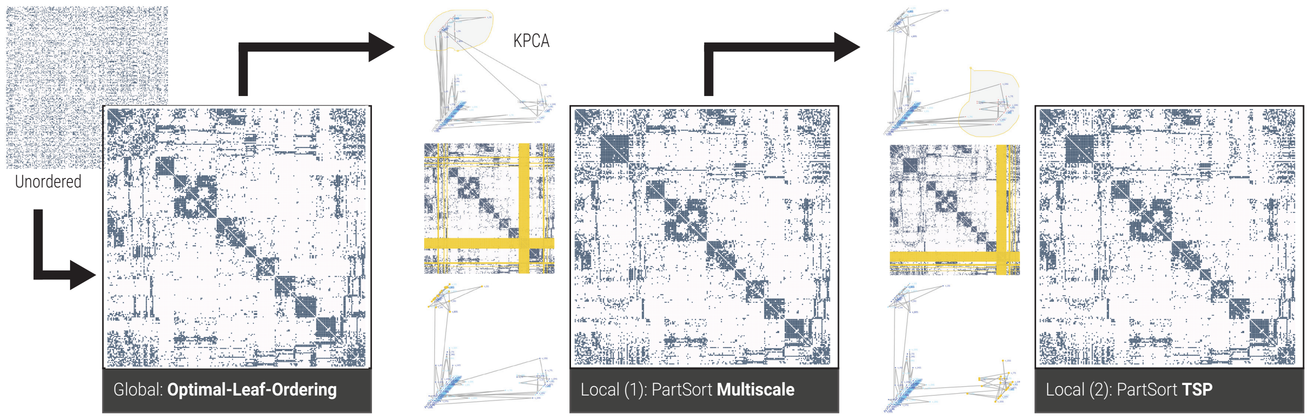


Fig. 7: In cases where global matrix reordering algorithms do not lead to satisfying reordering results, as depicted left, `GUIRO` can help to reveal hidden matrix substructures. We apply `Optimal-Leaf-Ordering`—a hierarchical clustering extension— to our dataset. The algorithm finds many distinct clusters on the diagonal, but fails to reveal structures in the regions on the top right and bottom left. The `Kernel PCA` projection depicts one large cluster and two subclusters. A lasso selection of the top left projection cluster reveals that many of the fragmented rows/columns on the top left are related to an existing, already coherent, block in the matrix. Reordering this submatrix with `MultiScale` has a major influence on the visual quality in the top right of the matrix, but reflects only mildly on the off-diagonal structures in this quadrant. Selecting and reordering the other projection cluster (for completeness) has only minor impact on the overall matrix appearance.

are not comparable, i.e., the subjects did not apply the same lasso selections. In UT3, which we estimated to be the most challenging task, all participants found a 100% match between `OLO` and `OLO_Complete`, but interestingly also agreed on the next two most similar rankings; a non-obvious choice. It will be interesting to correlate these rankings with quantitative ranking comparison metrics.

We asked the participants before ($\widetilde{x}$=3, IQR=1) and after the study ($\widetilde{x}$=4, IQR=1.25) to evaluate their confidence in assessing matrix patterns. Notably, their confidence increased on average by 1.27 points; the Wilcoxon signed rank test confirms this difference with a strong statistical significance ($p < .002$). Overall, the System Usability Scale (SUS) score was, as expected, on average $\mu = 62.95$ with a $\sigma = 14.629$; a typical result for a sophisticated Visual Analytics system.

7.2 Case Study on Submatrix Reordering

Figure 7 demonstrates the applicability of our approach by interactively improving the expressiveness of subparts of the matrix (**T4,6**). The dataset depicts a 236 × 236 matrix capturing social interactions in a primary school [23]. We begin our analysis with the `Optimal-Leaf-Ordering` algorithm in the single linkage variant (`OLO_Single`) [5]. Note, we depict all other reorderings in the Appendix. We interpret Fig. 7(1) globally (**T4**) and locally (**T5,6**) as follows: `OLO` tends to favor Block-Diagonals while having problems to grasp dense Off-Diagonal areas [7]. Nested patterns are not apparent.

Overall, we see a satisfactory reordering result (**T1**) with many distinct blocks on the diagonal (**T4**), i.e., the school's classes and class levels, although our algorithm seems to have problems in the upper left quadrant, respectively its associated rows and columns (**T6**). We want to focus our analysis, hence, on the hidden *local* patterns in this quadrant that the *global* algorithm was not able to express. For this, our first task is to find a suitable 2D projection representing the top-left submatrix's rows/columns distinctively. We choose `Kernel PCA` since it shows many long edges in the projection view connecting one central cluster (bottom left) with two satellite clusters (top right and bottom right) (**T6**). After applying a lasso selection to the (visually) larger cluster (top left), we see that our row/column vectors in the HD space are similar, but fragmented in the matrix plot. We inspect the reordering suggestions (depicted in the appendix) for the top left quadrant with two criteria: (1) visual quality; i.e., which additional visual patterns become apparent and (2) we take the quality metric Linear Arrangement into consideration (**T9,10**). Here, we choose `PartSort_MultiScale`. Reordering the rows/columns into one submatrix confirms our assumption, that the selected fragments indeed form a coherent bigger cluster with distinct off-diagonal patterns. Hence, our reordering steering has significantly improved the visual appearance of the upper left submatrix, revealing one of the largest and densest clusters in the network (**T7**). For the sake of completeness, we are reapplying the lasso selection on the lower right projection points, but can find that our reordering has little impact on the resulting quality.

7.3 Expert Review

Experiment Design: We employed an expert review as a method to evaluate our system's usefulness. Its goal is to complement, underpin, and contrast the aforementioned usability- and case study evaluation.
Participant: We were able to gather the feedback of an expert with over thirteen years of experience in the design and comparative evaluation of matrix reordering algorithms. Our expert is one of the leading researchers in the field with numerous publications on the topic.
Tasks and Procedure: We conducted two one-on-one interviews with the expert. The first session took around 17 minutes and introduced the topic and the general approach. After that, the expert had time to explore the online system on his own and collect questions or notes. The second, more comprehensive, session took 94 minutes and entailed a tutorial style walk-through in which all critical features were explained.
Results and Discussion: The responses were positive in general, especially about the aesthetics, the comprehensiveness of the algorithm collection, and the meaningful usage of animations.

However, many in-depth issues were discovered. For example, while the expert confirmed our expectations that `GUIRO` can be helpful for novices (**T1, 2**) and network analysts (**T4-6**), he mentioned that the system does not reflect the typical workflow of an algorithm designer. This user group, he stated, is constantly developing against a suite of unit tests assessing the correctness and quality of their approaches, i.e., (**T8**) is a purely automated process. On the other hand, though, algorithm designers might be interested in understanding result differences (a) in the projection space (**T9**) and (b) in the index (dis-)similarity plot (**T10**) to guide further improvement efforts. Furthermore, he suggested reordering the index plot with the help of established ranking-based correlation measures (Spearman and Kendall) or novel precedence invariant measures, such as the positional proximity coefficient [26] and to add a similarity visualization, such as presented in [27].

The second critical remark regards the choice of the projection technique. While he considered the overall idea meaningful that similar row/column vectors will be close in the projection space, he raised the concern that inexperienced users might choose "semantically" incorrect combinations. For example, the user might try to reason on a non-linear manifold in the dataset through a linear projection technique, such as `PCA`, and will wonder why a non-linear spectral matrix reordering actually produces visually pleasing results.

The third improvement idea relates to the guidance of users in the exploration of (local) patterns. The expert stated the vision, that users should be able to see descriptive, local highlights for automatically retrieved patterns. We support this vision; however, localized matrix pattern descriptors remain currently a research topic. One way to educate users in GUIRO, though, will be to provide a thorough selection of tutorial datasets each depicting one prominent visual pattern.

8 Related Work

For didactic purposes, we discussed the gap in the research literature throughout the paper (e.g., in Sect. 7) and embedded GUIRO into matrix research field in Sect. 3.2. This Section should give the reader a broader context, focusing on advancements in algorithmic and interactive relational data analysis.

8.1 Automatic Relational Data Analysis

Relational data analysis and visualization is unarguably challenging. Not only the dataset sizes range between a few hundred to many million or even billion items, but rather the linking and complex relationships among items is in the focus of the analysis [76]. These relationships can even be multivariate in nature or can change over time [6]. The research community devotes a significant portion of its efforts into finding solutions for typical analysis tasks, such as graph partitioning, connectivity or similarity analysis, finding relationships in multi-variate or high-dimensional spaces [42, 58, 77]. These approaches require not only sophisticated algorithmic techniques but also elaborate solutions for understanding the results. While the algorithmic field shows promising advances with, e.g., deep convolutional neural networks [16,38,45], the graph visualization domain is primarily concerned with externalizing graph topology characteristics [4, 34, 68] or designing effective overviews for large graph datasets, i.e., solving the hairball rendering problem [32, 35, 78].

Our proposed methods rely on projection to depict and interact with matrix reorderings. Many techniques exist to project high-dimensional data to 2D space [21, 59] and support the exploration process in data analysis tasks [84]. Most notably, our approach is similar to Harel and Koren's work on *Graph Drawing by High-Dimensional Embedding (HDE)* [30]. In HDE, the positions of vertices are first computed in an HD space and then projected to 2/3D for drawing. The projection technique is supposed to retain HD structures and reflect them in the low-dimensional space. However, HDE is well-known to be fast and frequently leads to unsatisfactory results. We still believe that the approach is valid and thus allow our users to examine different projection techniques. We share this claim with Turkay et al., who present a *Representative Factor Generation*, i.e., a pattern-finder for HD spaces [72]. Here, the global projection shows the overall HD space topology, while local artifacts can be resolved visually by user intervention.

It is also known that the humans' performance on a visually presented Traveling Salesman Problem (TSP) decreases non-linearly with the number of displayed points [75]. While our projection-space interactions must also lead to a Hamiltonian path, we follow a VA approach. We act on the user's intervention complete the path to obey our requirements. Consequently, our offered algorithmic support reduces the cognitive load to solve a TSP problem.

8.2 Interactive Matrix Visualizations

For many users, matrix representations are less intuitive than node-link diagrams, and should be augmented with interaction and other visualizations to increase user understanding [53]. In MatLink [33], matrices are enhanced by overlays to improve the performance of path finding tasks in large networks. In GeneaQuilts [11] and Compressed Adj. Matrices [18], the matrix's symmetric structure is compressed and rearranged to guide the user to interesting aspects in layered networks. Semantic zoom interaction can help to navigate matrices which do not fit into the available screen space. In several papers, zooming and dynamic aggregation techniques support the user navigation process in large matrices [3, 9, 19]. Also recently, Wong et. al noted in [82] that the ease of understanding for node-link diagrams and the information density of matrix representations show a mutual benefit.

9 Discussion and Future Work

When compared to the related work on interactive matrix reordering our approach goes beyond the state-of-the-art by incorporating three main aspects that potentially lead to a reordering improvement in one visual analysis system. Firstly, our system guides the user to submatrices, which could potentially be improved by showing row/column similarity in a novel projection view. Secondly, the user may choose to apply an arbitrary automatic matrix reordering algorithm on selected submatrices. We help the user by showcasing thumbnails of the local reorderings to help to anticipate the operation's outcome. Thirdly, the user is free to rearrange the rows and columns—or groups thereof—manually. In addition, our approach allows the hierarchical construction of new matrix reordering algorithms, by applying local optimizations on a global matrix reordering result.

Currently, our approach is restricted to symmetric matrices as they are common in adjacency matrices for undirected graphs or similarity matrices. In the future, we plan to generalize our approach to non-symmetric and even to non-square matrices. Furthermore, we are researching other algorithms, such as clustering, to improve the reordering interactions and guide the steering process. Heuristic graph-based algorithms can be applied to find (near-)optimal linear arrangement paths in either a group of selected vertices or after a manually reordering decision. The question which projection technique is able to reflect the HD structures is addressed in GUIRO pragmatically: We allow the user to choose between 10 implemented projection methods. We found, though, that linear projection techniques are easier to interpret, while non-linear techniques reveal our algorithm shortcomings patterns more easily. It would be interesting to screen the projection view appearances automatically for the patterns described in Sect. 4.2.

In terms of scalability, our system is primarily limited by two design choices: First, in order to allow a wide audience to use GUIRO, we decided for a web-based client/server architecture. This has significant performance implications since our bottleneck is the necessary serialization and marshaling for transmitting our data. Right now, we implemented consistently REST interfaces to access our servers. In the future, though, we will be able to process matrices larger than a few hundred rows and columns by changing the inter-service communication to binary and compressed transmission formats, such as gRCP, Protobuf, or Apache Thrift. Second, our matrix visualization implementation is SVG-based and consequently lets the HTML Document Object Model (DOM) grow exponentially in size with the number of rows/columns in the input data. An obvious solution will be to replace the renderer with an HTML canvas-based or WebGL-based alternative.

10 Conclusion

Matrix data are gathered in many application scenarios. However, only a suitable row and column reordering allows the perception of underlying matrix patterns. While previous work centered on fully-automated reordering algorithms, we propose with GUIRO a Visual Analytics approach. During the process, users can apply global matrix reorderings, while steering them in the local context. By applying several, potentially varying algorithms to distinct local submatrices, the users can construct their own new matrix reordering composition.

In our web- and microservice-based system, we introduce a novel technical solution for gathering user feedback in the process. The projection-based reordering representation allows us to understand and perceive the decisions made by the black-box matrix reordering algorithms and gives rise to foreseeable interactive intervention.

We evaluated our approach in a user study with 12 participants to test the usefulness of GUIRO, a case study-driven evaluation on a social network dataset to demonstrate local reorderings in a global context, and conducted an expert study to validate our approach from an external point of view. Our results reveal, that the presented interaction mechanisms prove to be effective and enhance user understanding, while the matrix reordering composition is powerful, but hard to understand.

In conclusion, the presented visual analytics system serves as a basis for a user-steerable matrix reordering process and allows to understand and steer the black-box matrix reordering algorithms.

References

[1] *BOOST C++ Libraries*. http://www.boost.org.

[2] *Smile (Statistical Machine Intelligence and Learning Engine)*. http://haifengl.github.io/smile/.

[3] J. Abello and F. van Ham. Matrix zoom: A visual interface to semi-external graphs. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, vol. 1, pp. 183–190. IEEE, 2004.

[4] B. Alper, B. Bach, N. H. Riche, T. Isenberg, and J. Fekete. Weighted graph comparison techniques for brain connectivity analysis. In *2013 ACM SIGCHI Conference on Human Factors in Computing Systems, CHI '13, Paris, France, April 27 - May 2, 2013*, pp. 483–492, 2013. doi: 10.1145/2470654.2470724

[5] Z. Bar-Joseph, D. K. Gifford, and T. S. Jaakkola. Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics*, 17(suppl 1):S22–S29, 2001.

[6] F. Beck, M. Burch, S. Diehl, and D. Weiskopf. The state of the art in visualizing dynamic graphs. In *Eurographics Conference on Visualization, EuroVis 2014 - State of the Art Reports, STARs, Swansea, UK, June 9-13, 2014*, 2014. doi: 10.2312/eurovisstar.20141174

[7] M. Behrisch, B. Bach, N. H. Riche, T. Schreck, and J.-D. Fekete. Matrix Reordering Methods for Table and Network Visualization. *Computer Graphics Forum*, 35(3):693–716, jun 2016. doi: 10.1111/cgf.12935

[8] M. Behrisch, M. Blumenschein, N. W. Kim, L. Shao, M. El-Assady, J. Fuchs, D. Seebacher, A. Diehl, U. Brandes, H. Pfister, T. Schreck, D. Weiskopf, and D. A. Keim. Quality Metrics for Information Visualization. *Computer Graphics Forum*, 2018. doi: 10.1111/cgf.13446

[9] M. Behrisch, J. Davey, T. Schreck, J. Kohlhammer, and D. A. Keim. Matrix-Based Visual Correlation Analysis on Large Timeseries Data. *Proc. IEEE Symposium on Visual Analytics Science and Technology (Poster Paper)*, 2012.

[10] M. Behrisch, L. Shao, J. Buchmüller, and T. Schreck. Quality Metrics Driven Approach to Visualize Multidimensional Data in Scatterplot Matrix. In *Eurographics Conference on Visualization (EuroVis) - Poster Paper*, 2015.

[11] A. Bezerianos, P. Dragicevic, J.-d. Fekete, J. Bae, and B. Watson. GeneaQuilts : A System for Exploring Large Genealogies. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1073–1081, 2010. doi: 10.1109/TVCG.2010.159

[12] T. Caelli and S. Kosinov. An eigenspace projection clustering method for inexact graph matching. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(4):515–519, 2004. doi: 10.1109/TPAMI.2004.1265866

[13] G. Caraux and S. Pinloche. Permutmatrix: a graphical environment to arrange gene expression profiles in optimal linear order. *Bioinformatics*, 21(7):1280–1281, 2005.

[14] C. H. Chen. Generalized association plots: information visualization via iteratively generated correlation matrices. *Statistica Sinica*, 12:7–29, 2002.

[15] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th National Conference*, ACM '69, pp. 157–172. ACM, New York, NY, USA, 1969. doi: 10.1145/800195.805928

[16] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 3837–3845, 2016.

[17] J. Diaz, J. Petit, and M. Serna. A survey of graph layout problems. *ACM Computing Surveys*, 34(3):313–356, Sept. 2002. doi: 10.1145/568522.568523

[18] K. Dinkla, M. Westenberg, and J. van Wijk. Compressed adjacency matrices: Untangling gene regulatory networks. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2457–2466, 2012. doi: 10.1109/TVCG.2012.208

[19] N. Elmqvist, T.-N. Do, H. Goodell, N. Henry, and J.-D. Fekete. ZAME: Interactive Large-Scale Graph Visualization. *2008 IEEE Pacific Visualization Symposium*, pp. 215–222, Mar. 2008.

[20] N. Elmqvist and J. S. Yi. Patterns for visualization evaluation. *Information Visualization*, 14(3):250–269, 2015. doi: 10.1177/1473871613513228

[21] B. S. Everitt, G. Dunn, et al. *Applied multivariate data analysis*, vol. 2. Arnold London, 2001.

[22] M. Friendly and E. Kwan. Effect ordering for data displays. *Comput. Statist. Data Anal., this*, 2003.

[23] V. Gemmetto, A. Barrat, and C. Cattuto. Mitigation of infectious disease at school: targeted class closure vs school closure. *BMC Infectious Diseases*, 14(1):695, Dec 2014. doi: 10.1186/s12879-014-0695-9

[24] J. A. George. *Computer implementation of the finite element method*. PhD thesis, Stanford University, 1971.

[25] M. Ghoniem, J.-D. Fekete, and P. Castagliola. On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. *Information Visualization*, 4(2):114–135, July 2005. doi: 10.1057/palgrave.ivs.9500092

[26] J. Y. Goulermas, A. Kostopoulos, and T. Mu. A new measure for analyzing and fusing sequences of objects. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(5):833–848, 2016. doi: 10.1109/TPAMI.2015.2470671

[27] M. Hahsler. An experimental comparison of seriation methods for one-mode two-way data. *European Journal of Operational Research*, 257(1):133–143, 2017.

[28] M. Hahsler, C. Buchta, and K. Hornik. *Infrastructure for seriation*, r package version 1.0-14. ed., 2014.

[29] M. Hahsler, K. Hornik, and C. Buchta. Getting things in order: An introduction to the r package seriation. *Journal of Statistical Software*, 25(3):1–34, March 2008.

[30] D. Harel and Y. Koren. Graph drawing by high-dimensional embedding. In *International symposium on graph drawing*, pp. 207–219. Springer, 2002.

[31] S. G. Hart. Nasa-task load index (nasa-tlx); 20 years later. In *Proceedings of the human factors and ergonomics society annual meeting*, vol. 50, pp. 904–908. Sage publications Sage CA: Los Angeles, CA, 2006.

[32] N. Henry and J.-D. Fekete. MatrixExplorer: a Dual-Representation System to Explore Social Networks. *IEEE Transactions on Visualization and Computer Graphics*, 12:677–684, 2006.

[33] N. Henry and J.-D. Fekete. MatLink: Enhanced Matrix Visualization for Analyzing Social Networks. In *Proceedings of the 11th IFIP TC 13 International Conference on Human-computer Interaction - Volume Part II*, INTERACT'07, pp. 288–302. Springer-Verlag, Berlin, Heidelberg, 2007.

[34] N. Henry, J.-D. Fekete, and M. J. McGuffin. NodeTrix: a hybrid visualization of social networks. *IEEE transactions on visualization and computer graphics*, 13(6):1302–9, 2007.

[35] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Trans. Vis. Comput. Graph.*, 12(5):741–748, 2006. doi: 10.1109/TVCG.2006.147

[36] 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. doi: 10.1111/j.2044-8317.1974.tb00534.x

[37] S. Kaiser and F. Leisch. A toolbox for bicluster analysis in r. 2008.

[38] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pp. 655–665, 2014.

[39] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer. Enterprise data analysis and visualization: An interview study. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2917–2926, 2012.

[40] E. Kandogan, A. Balakrishnan, E. M. Haber, and J. S. Pierce. From data to insight: Work practices of analysts in the enterprise. *IEEE Computer Graphics and Applications*, 34(5):42–50, 2014. doi: doi.ieeecomputersociety.org/10.1109/MCG.2014.62

[41] P. Kerpedjiev, N. Abdennur, F. Lekschas, C. McCallum, K. Dinkla, H. Strobelt, J. M. Luber, S. B. Ouellette, A. Azhir, N. Kumar, J. Hwang, S. Lee, B. H. Alver, H. Pfister, L. A. Mirny, P. J. Park, and N. Gehlenborg. Higlass: Web-based visual exploration and analysis of genome interaction maps. *bioRxiv*, 2018. doi: 10.1101/121889

[42] D. Knoke and J. H. Kuklinski. Network analysis. 1982.

[43] Y. Koren and D. Harel. A multi-scale algorithm for the linear arrangement problem. In *Revised Papers from the 28th International Workshop on Graph-Theoretic Concepts in Computer Science*, WG '02, pp. 296–309. Springer-Verlag, London, UK, UK, 2002.

[44] S. Kosinov and T. Caelli. Inexact multisubgraph matching using graph eigenspace and clustering models. In *In Proceedings of SSPR/SPR*, pp. 133–142. Springer, 2002.

[45] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, 2017. doi: 10.1145/3065386

[46] B. Lee, C. Plaisant, C. S. Parr, J.-D. Fekete, and N. Henry. Task taxonomy for graph visualization. In *Workshop on BEyond time and errors: novel*

evaluation methods for information visualization, pp. 1–5. ACM, 2006.

[47] F. Lekschas, B. Bach, P. Kerpedjiev, N. Gehlenborg, and H. Pfister. Hipiler: Visual exploration of large genome interaction matrices with interactive small multiples. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):522–531, 2018. doi: 10.1109/TVCG.2017.2745978

[48] I. Liiv. Seriation and matrix reordering methods: An historical overview. *Statistical Analysis and Data Mining*, 3(2):70–91, 2010. doi: 10.1002/sam.10071

[49] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003.

[50] E. Mäkinen and H. Siirtola. The barycenter heuristic and the reorderable matrix. *Informatica (Slovenia)*, 29(3):357–364, 2005.

[51] W. T. McCormick, S. B. Deutsch, J. J. Martin, and P. J. Schweitzer. Identification of data structures and relationships by matrix reordering techniques. Technical report, DTIC, 1969.

[52] W. T. McCormick, P. J. Schweitzer, and T. W. White. Problem decomposition and data reorganization by a clustering technique. *Operations Research*, 20(5):993–1009, 1972. doi: 10.1287/opre.20.5.993

[53] C. Mueller, B. Martin, and a. Lumsdaine. Interpreting large visual similarity matrices. *2007 6th International Asia-Pacific Symposium on Visualization*, pp. 149–152, Feb. 2007. doi: 10.1109/APVIS.2007.329290

[54] J. New, W. Kendall, J. Huang, and E. Chesler. Dynamic visualization of coexpression in systems genetics data. *IEEE Transactions on Visualization and Computer Graphics*, 14(5):1081–1095, Sep. 2008. doi: 10.1109/TVCG.2008.61

[55] S. Niermann. Optimizing the ordering of tables with evolutionary computation. *The American Statistician*, 59(1), 2005.

[56] C. Nobre, M. Streit, M. Meyer, and A. Lex. The state of the art in visualizing multivariate networks. *Computer Graphics Forum (EuroVis '19), to appear*, 2019.

[57] M. Okoe, R. Jianu, and S. G. Kobourov. Node-link or adjacency matrices: Old question, new insights. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2018. doi: 10.1109/TVCG.2018.2865940

[58] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[59] F. V. Paulovich, M. C. F. Oliveira, and R. Minghim. The projection explorer: A flexible tool for projection-based multidimensional visualization. In *Proceedings of the XX Brazilian Symposium on Computer Graphics and Image Processing - SIBGRAPI*, pp. 27–36. IEEE CS Press, Belo Horizonte, Brazil, 2007.

[60] C. Perin, P. Dragicevic, and J.-D. Fekete. Revisiting Bertin matrices: New Interactions for Crafting Tabular Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, Nov. 2014. doi: 10.1109/TVCG.2014.2346279

[61] J. Petit. Experiments on the minimum linear arrangement problem. *J. Exp. Algorithmics*, 8, Dec. 2003.

[62] J. Pretorius, H. C. Purchase, and J. T. Stasko. Tasks for multivariate network analysis. In A. Kerren, H. C. Purchase, and M. O. Ward, eds., *Multivariate Network Visualization - Dagstuhl Seminar #13201, Dagstuhl Castle, Germany, May 12-17, 2013, Revised Discussions*, vol. 8380 of *Lecture Notes in Computer Science*, pp. 77–95. Springer, 2013.

[63] R. Rao and S. K. Card. The table lens: merging graphical and symbolic representations in an interactive focus+ context visualization for tabular information. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 318–322. ACM, 1994.

[64] J. Raudeliūnienė, V. Davidavičienė, M. Tvaronavičienė, and L. Jonuška. Evaluation of advertising campaigns on social media networks. *Sustainability*, 10(4):973, 2018.

[65] W. S. Robinson. A method for chronologically ordering archaeological deposits. *American antiquity*, 16(4):293–301, 1951.

[66] R. A. Rossi and N. K. Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[67] J. Sauro and J. S. Dumas. Comparison of three one-question, post-task usability questionnaires. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009, Boston, MA, USA, April 4-9, 2009*, pp. 1599–1608, 2009. doi: 10.1145/1518701.1518946

[68] B. Shneiderman and A. Aris. Network visualization by semantic substrates. *IEEE Trans. Vis. Comput. Graph.*, 12(5):733–740, 2006. doi: 10.1109/TVCG.2006.166

[69] H. Siirtola and E. Mäkinen. Constructing and reconstructing the reorderable matrix. *Information Visualization*, 4(1):32–48, Mar. 2005. doi: 10.1057/palgrave.ivs.9500086

[70] M. Spenke, C. Beilken, and T. Berlage. Focus: the interactive table for product comparison and selection. In *ACM Symposium on User Interface Software and Technology*, pp. 41–50. ACM, 1996.

[71] Y. Tien, Y. Lee, H. Wu, and C. Chen. Methods for simultaneously identifying coherent local clusters with smooth global patterns in gene expression profiles. *BMC Bioinformatics*, 9, 2008. doi: 10.1186/1471-2105-9-155

[72] C. Turkay, A. Lundervold, A. J. Lundervold, and H. Hauser. Representative factor generation for the interactive visual analysis of high-dimensional data. *IEEE Trans. Vis. Comput. Graph.*, 18(12):2621–2630, 2012. doi: 10.1109/TVCG.2012.256

[73] R. B. J. van Brakel, M. W. E. J. Fiers, C. Francke, M. A. Westenberg, and H. van de Wetering. Combat: Visualizing co-occurrence of annotation terms. In *IEEE Symposium on Biological Data Visualization*, pp. 17–24, 2013. doi: 10.1109/BioVis.2013.6664342

[74] S. van den Elzen, D. Holten, J. Blaas, and J. J. van Wijk. Reducing snapshots to points: A visual analytics approach to dynamic network exploration. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):1–10, jan 2016. doi: 10.1109/tvcg.2015.2468078

[75] D. Vickers, M. Butavicius, M. Lee, and A. Medvedev. Human performance on visually presented traveling salesman problems. *Psychological Research*, 65(1):34–45, 2001.

[76] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. van Wijk, J.-D. Fekete, and D. Fellner. Visual analysis of large graphs: State-of-the-art and future research challenges. *Wiley-Blackwell Computer Graphics Forum*, 2011.

[77] S. Wasserman. *Social network analysis: Methods and applications*, vol. 8. Cambridge university press, 1994.

[78] M. Wattenberg. Visual exploration of multivariate graphs. In *Proceedings of the 2006 Conference on Human Factors in Computing Systems, CHI 2006, Montréal, Québec, Canada, April 22-27, 2006*, pp. 811–819, 2006. doi: 10.1145/1124772.1124891

[79] T. Wei. *Corrplot: Visualization of a correlation matrix*, r package version 0.73. ed., Oct 2013.

[80] L. Wilkinson. *The Grammar of Graphics (Statistics and Computing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[81] L. Wilkinson and M. Friendly. The history of the cluster heat map. *The American Statistician*, 2009.

[82] P. C. Wong, P. Mackey, H. Foote, and R. May. Visual matrix clustering of social networks. *Computer Graphics and Applications, IEEE*, 33(4):88–96, July 2013. doi: 10.1109/MCG.2013.66

[83] H.-M. Wu, S. Tzeng, and C.-H. Chen. *Handbook of Data Visualization*, chap. Matrix Visualization, pp. 681–708. Springer, 2008.

[84] D. Yang, Z. Xie, E. A. Rundensteiner, and M. O. Ward. Managing discoveries in the visual analytics process. *SIGKDD Explor. Newsl.*, 9(2):22–29, Dec. 2007.

[85] F. Zeng, R. Tao, Y. Yang, and T. Xie. How social communications influence advertising perception and response in online communities? *Frontiers in psychology*, 8:1349, 2017.