Simultaneous Matrix Orderings for Graph Collections

Nathan van Beusekom, Wouter Meulemans, and Bettina Speckmann

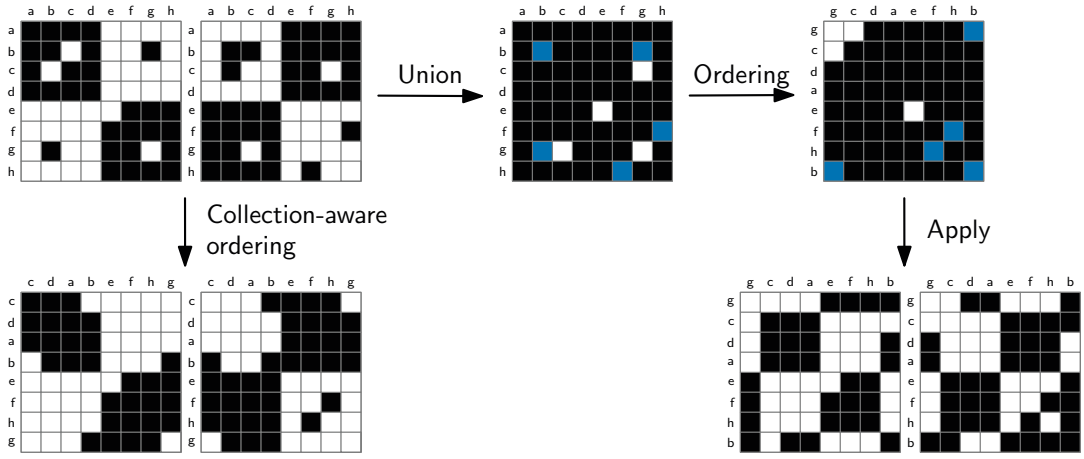

Fig. 1. A collection of two matrices (top left). The state-of-the-art first computes a (weighted) union (top middle and right, blue squares have weight 2), then orders the union, and finally applies this ordering to all matrices in the collection. The union leads to a loss of information, specifically, on those parts of the matrices which are different (bottom right). We propose a collection-aware approach to compute orderings which avoids this loss of information (bottom left). Our approach can be applied to existing ordering methods; examples in this figure use the popular leaf order heuristic.

Abstract—Undirected graphs are frequently used to model phenomena that deal with interacting objects, such as social networks, brain activity and communication networks. The topology of an undirected graph G can be captured by an adjacency matrix; this matrix in turn can be visualized directly to give insight into the graph structure. Which visual patterns appear in such a matrix visualization crucially depends on the *ordering* of its rows and columns. Formally defining the quality of an ordering and then automatically computing a high-quality ordering are both challenging problems; however, effective heuristics exist and are used in practice.

Often, graphs do not exist in isolation but as part of a collection of graphs on the same set of vertices, for example, brain scans over time or of different people. To visualize such graph collections, we need a *single* ordering that works well for all matrices *simultaneously*. The current state-of-the-art solves this problem by taking a (weighted) union over all graphs and applying existing heuristics. However, this union leads to a loss of information, specifically in those parts of the graphs which are different. We propose a *collection-aware* approach to avoid this loss of information and apply it to two popular heuristic methods: leaf order and barycenter.

The de-facto standard computational quality metrics for matrix ordering capture only block-diagonal patterns (cliques). Instead, we propose to use *Moran's I*, a spatial auto-correlation metric, which captures the full range of established patterns. Moran's I refines previously proposed stress measures. Furthermore, the popular leaf order method heuristically optimizes a similar measure which further supports the use of Moran's I in this context. An ordering that maximizes Moran's I can be computed via solutions to the Traveling Salesperson Problem (TSP); orderings that approximate the optimal ordering can be computed more efficiently, using any of the approximation algorithms for metric TSP.

We evaluated our methods for simultaneous orderings on real-world datasets using Moran's I as the quality metric. Our results show that our collection-aware approach matches or improves performance compared to the union approach, depending on the similarity of the graphs in the collection. Specifically, our Moran's I-based collection-aware leaf order implementation consistently outperforms other implementations. Our collection-aware implementations carry no significant additional computational costs.

Index Terms—Matrix ordering, graph visualization, algorithms, quality measures

1 Introduction

Graphs are a common tool to model interacting entities, be it humans in social networks, synapses in human brains, or servers in computer networks. Visualizations are a natural tool to explore the structure of graphs and to analyze the underlying interactions. The majority of current graph analysis tools make use of node-link diagrams to visualize graphs. However, matrix visualizations of graphs – which directly draw the graphs' adjacency matrix – have been shown to be effective for low-level analysis tasks [21] ('are the vertices x and y connected') and for comparisons of (large) graphs [1]. A matrix visualization can highlight local structures in graphs – such as clusters, bi-cliques, or stars – but this relies on a suitable *ordering* of the rows and columns to make these structures manifest as visual patterns. Formally defining the quality of an ordering and then automatically computing a high-quality ordering are challenging problems. A multitude of different techniques have been proposed over the years; see the survey by Behrisch *et al.* [8] for an extensive discussion.

Often, graphs do not appear in isolation, but they are rather part of a collection of graphs which share a common set of vertices. Examples include dynamic graphs (such as brain activity of one person over time)

TU Eindhoven, the Netherlands. E-mail: [n.a.c.v.beusekom,w.meulemans,b.speckmann]@tue.nl.

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

and graphs with distinct edge sets (such as brain activity patterns of different persons). In either case, visualizing these graph collections with matrices requires us to compute a *single* ordering that works well for all graphs *simultaneously*. Using one single ordering for all graphs in a collection facilitates comparisons between the matrices; at the same time, the ordering should allow local structures in each matrix to visually manifest themselves.

There are only a few visual analysis tools for graph collections that use matrix visualizations. The *Cubix* system by Bach *et al.* [3] allows the user to explore the evolution of a dynamic network using a matrix visualization for each time step. These matrix visualizations are then combined into a so-called matrix cube, using a single, possibly user-defined, ordering for all matrices. The user can reorder any of the matrices in the cube using a variety of provided ordering algorithms; the new ordering is computed based on the values in this single selected matrix only and then applied to all others. The *MultiPiles* technique by Bach *et al.* [2] is another approach that facilitates the analysis of time-series of graphs. Here collections of adjacency matrices are "piled" together, using temporal clustering or based on user interactions; the piles are then arranged in a small multiples style layout to facilitate comparisons. Among the large set of user interactions is the option to impose a so-called local order on a pile (collection) of matrices: this ordering is computed for all matrices in the pile using the weighted union of all matrices as input for the leaf order heuristic. Using a union matrix inevitably loses information about the individual matrices, and hence, the ordering algorithms have less information at their disposal. Nevertheless, this approach works well if all matrices in the pile are fairly similar. However, as illustrated in the example in Figure 1, if the graphs contain complementary edges, then the union does not contain enough information to arrive at a good simultaneous ordering.

In this paper we propose a *collection-aware* approach to computing simultaneous matrix orderings for collections of graphs. As most previous work, we focus our attention on undirected graphs, that is, symmetric 0-1 valued adjacency matrices (indicated with 0:white and 1:black squares in all figures). Our basic premise is that a method will produce an ordering which works well for all matrices simultaneously, if the decisions that the method takes are based on the individual matrices – as far as that is possible. Since any method needs to produce a single ordering in the end, eventually information from all matrices in the collection needs to be aggregated. However, the further down the algorithmic pipeline this aggregation is happening, the more of the local structures in the input matrices are preserved. We demonstrate the soundness of our premise and the feasibility of our approach by describing collection-aware variants of two popular matrix ordering methods: the leaf order [4] and barycenter [14, 19, 27] heuristics.

The de-facto standard computational quality metrics for matrix visualizations are based on distances induced by the input graph. Specifically, the *bandwidth*, *profile*, and *linear arrangement* measures count how far each edge is removed from the diagonal (using different schemes to aggregate this information into a single value). By design these measures are optimized by orderings that move matrix cells of edges as close to the diagonal as possible. However, many meaningful patterns that occur in matrix visualizations, representing graph features such as bi-cliques or stars (see the survey by Behrisch *et al.* [8]), are in fact not close to the diagonal and hence are not captured by these measures. The same survey is calling for future research into quantitative measures which evaluate the quality of all patterns. To overcome the current lack of such quantitative measures, Behrisch *et al.* [7] propose to describe and compare matrices via feature vectors (*Magnostics*) that describe specific patterns. These vectors are well-suited to support database queries and user interactions, but they are less suited for computational benchmarks and algorithmic optimization.

We observe that salient patterns in matrix visualizations are formed by clusters of black and white cells – a simple form of spatial auto-correlation. We hence propose to use Moran's I [30], a prominent measure for spatial auto-correlation, as a quality metric for matrix visualizations. Moran's I counts vertical and horizontal adjacencies in three classes: black-black, white-white, and black-white (see Figure 2). Black-black and white-white adjacencies contribute positively to the count, weighted by relative frequency of white or black, while black-white adjacencies contribute negatively. The final score is normalized to lie between -1 (checkerboard pattern) and $+1$ (a white or black matrix). Moran's I is closely related to several quality metrics which are used internally in matrix re-ordering algorithms (see Section 7.2 in the survey by Behrisch *et al.* [9] on quality metrics). However, until now, none of these metrics have been used as a quality metric for the final matrix visualization.

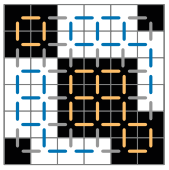

Fig. 2. Moran's I counts vertical and horizontal adjacencies in three classes: black-black, white-white, and black-white.

Contributions. Summarizing the above, the contributions of our paper are twofold. Our primary contribution is the new collection-aware approach hinted at above, with a description of how to apply this approach to the leaf order and barycenter heuristics.

Our second contribution is the introduction of Moran's I as a quality metric for matrix visualizations. Specifically, we explain how Moran's I relates to existing measures and algorithms, we show that Moran's I captures the full range of established patterns and anti-patterns in matrix visualizations, and we explain how to efficiently approximate an optimal matrix ordering for Moran's I.

Throughout the main body of the paper we focus on collections of undirected, unweighted graphs with symmetric adjacency matrices. In Section A in the supplementary material we consider the more general case of directed graphs. There we also argue that for undirected graphs it is optimal (under Moran's I) to use a single ordering for rows and columns. Hence, we restrict our discussion in the main paper to a single ordering. However, our techniques and measures readily generalize to weighted or directed graphs.

We validate the efficacy of our methodology using a short computational experiment using implementations of the collection-aware and union approaches, to showcase that the loss of information indeed occurs in real-world data, and that the collection-aware approach overcomes this problem.

The paper is organized as follows. In Section 2 we briefly summarize notations and definitions. We provide details on measuring quality and distances for matrix orderings and provide our exposition on Moran's I in Section 3. We then describe and review the union approach, and our newly proposed collection-aware approach and its implementation in Section 4. We describe in Section 5 the setup and results of our computational experiments, and close in Section 6 with a review of our findings and possible future work.

Related work. We are not aware of previous work that computes simultaneous orderings for matrices, beyond the union approach of MultiPiles [2]; see above, as well as Section 4 for a discussion of this method. A discussion of related work in terms of ordering quality is deferred to Section 3.

In the graph-drawing literature, we find related work on simultaneously drawing graphs in the node-link diagram style [10]. The goal here typically is to avoid crossings, i.e., draw graphs planarly, on a common vertex set. In other words, can locations for each vertex be found, such that for each graph, all edges of that graph can be drawn using these locations (e.g., with straight lines or few bends per edge) while not introducing intersections. Note that crossings between edges of different graphs are thus allowed. Results here focus on theoretical aspects, establishing computational complexity, i.e., showing that the problem is hard in general [16] but that other variants admit polynomial-time solutions [12, 18, 22]. Recently, simultaneous embeddings were generalized to graphs drawn in 3D, with the goal that different two-dimensional projections preserve user specified distances [24]. Beck *et*

al. [5] present an extensive overview on visualization techniques for dynamic graphs, which also touches upon matrix visualizations.

We focus on simple black-and-white representations of undirected, unweighted graphs. However, there are many possibilities to augment matrix visualizations. For example, cells can be used to display auxiliary data of the edges, including temporal data [15, 34]. Moreover, augmentations with lines can help overcome some of the drawbacks of matrix visualizations, such as the identification of paths [23, 32].

2 Preliminaries

Graphs. A graph $G = (V, E)$ consists of a set V of n vertices and a set $E \subseteq V^2$ of m edges. We assume graphs to be undirected, that is, $(u, v) \in E \Leftrightarrow (v, u) \in E$. Typically, undirected graphs do not contain self-loops $(u, u) \in E$, but we assume that these may occur.

With $N(G, v)$ we denote the neighborhood of vertex v in graph G. Specifically, we interpret it as an n-dimensional 0-1 vector: an entry is 1 if and only if $(v, u_i) \in E$ where u_i is the ith vertex in some arbitrary fixed order of the vertices in V.

Orderings. An *ordering* of a graph G is a permutation of its vertices V, which we represent as a bijective function on the indices, $\rho\colon \{1, \ldots, n\} \to V$. So, $\rho(1)$ is the first vertex in the ordering, and $\rho^{-1}(v)$ is the rank (position) of $v \in V$ in the ordering.[1] We use $\rho(i, j)$ as a shorthand for the pair $(\rho(i), \rho(j))$, that is, a pair of vertices which may or may not constitute an edge in E.

Given an ordering ρ of G, we can create a table with n rows and n columns, where each row and column is associated with a vertex through the ordering. We color each cell $[i, j]$ in row i and column j of this table black if the edge $\rho(i, j)$ is in E, and white otherwise.

Simultaneous orderings. Assume that we are given a set $\mathscr{G} = \{G_1, \ldots, G_k\}$ of k undirected graphs . Each graph $G_i = (V, E_i)$ is defined on the same set V of vertices but has its own set of edges $E_i \subseteq V^2$. Our goal is to find a *simultaneous* ordering ρ for the set $\mathscr{G}$, that is, an ordering of the vertices V that results in good matrix visualizations for all graphs in the set $\mathscr{G}$.

3 Measuring Ordering Quality

Computational quality measures play an important role when designing and evaluating matrix ordering algorithms. Here measures are used essentially in three different ways: to evaluate the final quality of a matrix visualization, as part of the objective function to be optimized by an algorithm, or simply as distance function between two rows of the matrix. In principle each measure can be used in all three settings, however, we observe that so far, certain measures have been used only in one or two of them. We can roughly separate existing measures into two categories: measures that are based on distances in the ordering between vertices and measures that are based on adjacencies of row (or column) vectors in the matrix.

[1] Note that the survey by Behrisch *et al.* [8] defines the ordering function the other way around, from vertex to index; being a bijection, this is but a notational difference. We found that in our exposition, we rely mostly on the resulting row order so our definition avoids excessive use of the inverse.

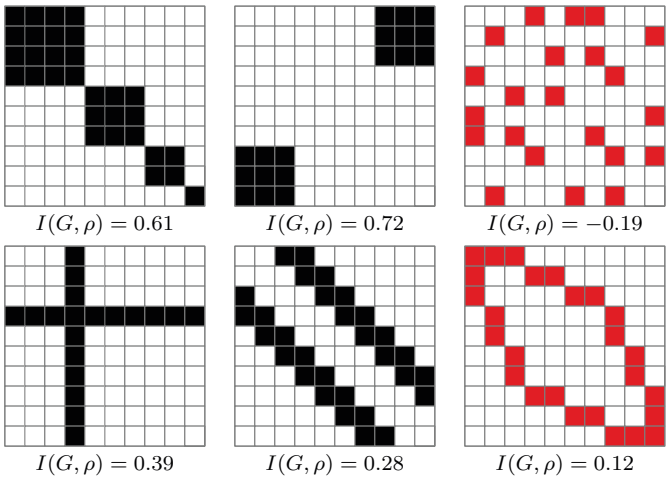

Fig. 3. Patterns (black) and anti-patterns (red) [8] with Moran's I score (higher = more correlated). Top row: block pattern, off-diagonal block pattern, noise anti-pattern; bottom row: line/star pattern, bands pattern, bandwidth anti-pattern.

Ordering distance. Measures in this category focus on combinatorial, connectivity aspects of the input graph. That is, they measure how well the ordering represents the (edge) connectivity in the graph. The measures attempt to capture the idea that nodes which are adjacent or at least close in the graph should be adjacent or close in the order and vice versa. The de-facto standard computational quality metrics are all based on the concept of rank difference (distance in the ordering): $\lambda_\rho(u, v) = |\rho^{-1}(u) - \rho^{-1}(v)|$ should be small for all $(u, v) \in E$. The function λ effectively expresses the deviation of an edge from the diagonal of the matrix. This principle is used to define three common measures [8]:

bandwidth which is the maximum deviation, $\max_{(u,v)\in E} \lambda(u, v)$

profile which measures per vertex the maximum deviation to an adjacent vertex earlier in the order,

$$\sum_{i=1}^{n} (i - \min_{j<i \wedge \rho(j,i)\in E} j) = \sum_{i=1}^{n} \max_{j<i\wedge\rho(j,i)\in E} \lambda(\rho(j), \rho(i))$$

linear arrangement which accounts for all edges,

$$\sum_{u\in V} \sum_{(u,v)\in E} \lambda(u, v) = \sum_{(u,v)\in E} \lambda(u, v)$$

All three measures are focused on keeping edges close to the diagonal and are hence optimized by orderings which form block patterns along the diagonal (see Figure 3 top left). They are used both in optimization functions and, specifically linear arrangement, as a measure for the final quality of a matrix visualization.

However, several meaningful patterns which correspond to salient structures in the input graph are not related to distance from the diagonal. Furthermore, the bandwidth anti-pattern (which does not match a logical structure in the input graph) is in fact a typical result of optimizing for ordering distance. See Figure 3 for an illustration of the most common patterns according to the survey by Behrisch *et al.* [8]. Table 1 lists each of the three ordering distance measures for each pattern and anti-pattern (lower = better). Bandwidth and linear arrangement, for example, assess the off-diagonal block pattern as worse than the bandwidth anti-pattern, and profile cannot distinguish between these. In contrast, Moran's I consistently ranks the patterns higher than the anti-patterns (higher = better).

Adjacency. Measures in this category are usually used to compute distances between two (adjacent) rows of a matrix, based on the directly adjacent cells in the respective rows. Two vertically or horizontally adjacent black squares correspond to two edges which share a vertex. Hence, in some sense these measures promote the clustering of the neighborhood of vertices in the graph into adjacent cells of the matrix. Adjacency measures naturally generalize to the complete matrix and hence are used also as part of optimization functions. So far, measures in this category do not appear to have been used as a quality measure for the final matrix visualization.

Moran's I, which we will describe in greater detail in the next subsection, is an adjacency measure, and so are the *measure of effectiveness* by McCormick *et al.* [28, 29] and the stress measure by Niermann [31]. All standard distance measures for vectors fall into this category as well, such as the Euclidean distance L_2. Note that the squared Euclidean distance is identical to the Manhattan distance for $0-1$ vectors.

Lenstra and Kan [26] observed that an optimal ordering for the measure of effectiveness is equivalent to a traveling salesperson path,

Table 1. Metrics for the patterns and anti-patterns in Figure 3. For the first three metrics lower is better, while for the Moran's I higher is better.

Pattern	Bandwidth	Profile	Linear arrangement	Moran's I
Block	3	10	30	0.61
Off-diagonal Block	9	24	126	0.72
Line/Star	6	24	54	0.39
Bands	3	23	74	0.28
Anti-pattern	**Bandwidth**	**Profile**	**Linear arrangement**	**Moran's I**
Noise	8	28	76	-0.19
Bandwidth	4	24	60	0.12

where each matrix row corresponds to a city and the distance between two rows is measured by the number of pairwise vertical black-black adjacencies. In fact, every adjacency measure is optimized globally via a traveling salesperson path. The *Bond Energy Algorithm* by McCormick *et al.* [28, 29] is in fact a heuristic for TSP using the measure of effectiveness. The popular leaf order heuristic is a heuristic for TSP as well, using the Euclidean distance.

3.1 Spatial auto-correlation: Moran's *I*

We observe that salient patterns are formed by clusters of black cells: moving rows (vertices) close together which have similar neighborhoods. As such, we postulate that promoting patterns is a form of spatial auto-correlation. Spatial auto-correlation measures are global measures of structure in the data. The matrix visualization in this context becomes the data for which we try to measure spatial auto-correlation.

We propose to use Moran's I [30], one of the prominent measures for spatial auto-correlation. This very general measure requires values associated with each cell and a weight matrix which captures how the cells are structurally (visually) related. In our case, we associate the values 1 and 0 with each cell, depending on whether the associated edge is in E or not. We design the weight matrix such that each cell is considered adjacent (with weight 1) to the cells with which it shares a border and unadjacent (weight 0) otherwise. Higher scores in Moran's I indicate a stronger correlation and are thus desirable for an ordering.

As described above, we use Rook's adjacency rather than Queen's adjacency[2]. Our motivation is twofold. First, two cells (i, j) and (i', j') correspond to two edges and thus up to four vertices. If each of these has a different value, we get indeed four vertices and the two edges connect two arbitrary pairs. As such, it does not match to a local structure. However, if the two cells are in the same column or row, there are only three vertices, and thus it describes a small pattern of two vertices sharing a common neighbor. Second, Moran's I with Queen's adjacency fails to capture negative spatial auto-correlation on the prototypical chess board of alternating black and white cells.

In our setting, Moran's I effectively simplifies to the following expression (see below for a derivation).

$$I(G,\rho) = c_B(G)\cdot B(G,\rho) + c_W(G)\cdot W(G,\rho) - 1$$

Here, $B(G,\rho)$ and $W(G,\rho)$ refer to the number of black-black and white-white adjacencies, respectively. See Figure 4 for illustrations. The c-terms are constants, relying only on the number of vertices and edges in G, that weigh the relative impact of these types of adjacencies. Generally speaking, if the matrix has more white cells than black cells, black-black adjacencies have more impact and vice versa.

Derivation. Here we present a brief derivation of our simplified form of Moran's I, starting from its general form. For a detailed derivation, refer to Section A of the supplementary material.

A graph $G = (V, E)$ on n vertices for some fixed ordering ρ implies an $n \times n$ 0-1 matrix M where $M_{ij} = 1$ if $\rho(i, j) \in E$ and 0 otherwise. The cells of M are interpreted as the spatial units for the sake of Moran's I, and contains all necessary information to derive Moran's I. We hence omit dependencies on G and ρ in our derivation here, for sake of notational simplicity.

We use slightly different notation than perhaps conventional for the general form of Moran's I, so as to distinguish between the general form and our simplified form. Specifically, we use r instead of N to denote the number of spatial units (regions) and use a weights matrix T (topology) with sum t instead of w with sum W. Moreover, we refer to the regions using indices a and b rather than i and j.

Moran's I is defined over r spatial units, which have associated values x_a for $a \in \{1, \ldots, r\}$. Furthermore, an $r \times r$ matrix T encodes the weights for (typically neighboring) regions: entry T_{ab} is the weight between region a and b. We use t to denote the sum over all weights in T. Moreover, let $\bar{x}$ denote the average value $\frac{\sum_{a=1}^{r} x_a}{r}$. The general form of Moran's I is as follows [30]:

$$I = \frac{r}{t} \cdot \frac{\sum_{a=1}^{r}\sum_{b=1}^{r} T_{ab}(x_a - \bar{x})(x_b - \bar{x})}{\sum_{a=1}^{r}(x_a - \bar{x})^2}$$

As the cells of M correspond to spatial units, we have $r = n^2$. Let m denote the total number of entries with value 1 in M (the number of edges, with double counting). Hence, $\bar{x} = \frac{m}{n^2}$ and we can rewrite $x_a - \bar{x} = \frac{x_a n^2 - m}{n^2}$. This allows us to simplify the generic form to

$$I = \frac{n^2}{t} \cdot \frac{\sum_{a=1}^{r}\sum_{b=1}^{r} T_{ab}(x_a n^2 - m)(x_b n^2 - m)}{\sum_{i=1}^{r}(x_a n^2 - m)^2}.$$

With Rook's adjacency, T_{ab} is 1 if a and b are adjacent cells in M and 0 otherwise. We thus need to consider only the terms for which T_{ab} is 1, i.e., for adjacent cells in M. Furthermore, since M is a 0-1 matrix, x_a is either 0 or 1: the term $x_a n^2 - m$ is either $n^2 - m$ or $-m$. In the denominator, there are hence two cases to consider (white cells and black cells) and for the numerator there are three cases (a and b describe a white-white, black-black, or black-white adjacency). The contribution to Moran's I is the same per case.

We thus case simplify this summation by simply counting the number of occurrences of each case, and we identify B (black-black), W (white-white) and D (black-white) with this count. There are $2n(n-1)$ adjacencies, but we count every adjacency twice since T is symmetric: $t = \sum T_{ab} = 4n(n-1)$.

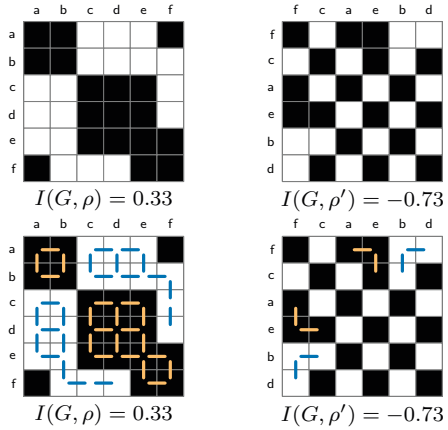

Fig. 4. Visualizations of the same graph based on two different orderings, ρ and ρ'. The bottom row shows the same visualizations, with annotated horizontal and vertical adjacencies. More such lines mean more same-colored adjacencies and thus a higher quality in terms of Moran's I.

[2]This refers to the movement capabilities of chess pieces.

$$I = \frac{n^2}{4n(n-1)} \cdot \frac{2B\cdot(n^2-m)^2 + 2W\cdot(-m)^2 + 2D\cdot(n^2-m)(-m)}{m(n^2-m)^2 + (n^2-m)(-m)^2}$$
$$= \frac{1}{2n(n-1)m(n^2-m)} \cdot (B\cdot(n^2-m)^2 + W\cdot m^2 - D\cdot(n^2-m)m)$$

Since $B+W+D = 2n(n-1)$, we see that it suffices to count only B and W. This allows us to simplify the expression to a sum of B and W, introducing c_B and c_B as the coefficients of B and W:

$$I = B\cdot\frac{n}{2(n-1)m} + W\cdot\frac{n}{2(n-1)(n^2-m)} - 1$$
$$= c_B\cdot B + c_W\cdot W - 1$$

Using Moran's *I*. With Moran's I, we have a measure that is aimed at capturing how well-structured the matrix visualization is. That is, it aims to globally capture patterns, without specifically aiming to specify what a pattern actually constitutes. We indeed see in Table 1 that the patterns score considerably higher than the anti-patterns. As such, this makes the measure more amenable for algorithmic use.

Indeed, we see that we are effectively counting cells in determining similarity of neighborhoods. We denote with $B(G,u,v)$ the number of (vertical) black-black adjacencies we would obtain when u and v are made adjacent in the ordering – the number of common neighboring vertices in G; similarly, $W(G,u,v)$ denotes the white-white adjacencies for u and v – the number of vertices in G that are neighboring to neither u nor v. Measuring the neighborhood similarity between u and v as $s(G,u,v) = c_B(G)\cdot B(G,u,v) + c_W(G)\cdot W(G,u,v)$, we get the following form of the metric:

$$I(G,\rho) = -1 + 2\sum_{i=1}^{n-1} s(G,\rho(i),\rho(i+1))$$

Maximizing Moran's I is equivalent to maximizing the sum of s over adjacent rows. However, many algorithms such as leaf order are based upon minimizing a sum of distances. Hence, we define a Moran's I metric $\delta_I(G,u,v) = 1 - s(G,u,v)$, which gives that $I(G,\rho) = n-2-2\sum_{i=1}^{n-1}\delta_I(G,\rho(i),\rho(i+1))$. Now, maximizing Moran's I corresponds exactly to minimizing the sum of distances. We do observe that δ_I is not a proper metric, since identical rows do not have a distance of zero. This identity of indiscernables is inherently incompatible with Moran's I as two fully black rows and two fully white rows should score differently depending on the number of black cells over the entire matrix. Nonetheless, the triangle inequality holds: $\delta_I(G,u,w) \le \delta_I(G,u,v) + \delta_I(G,v,w)$ rewrites to $1 + s(G,u,w) \ge s(G,u,v) + s(G,u,w)$. However, we know that $I(G,\rho)$ cannot exceed one and thus $s(G,u,v) + s(G,u,w) \le 1$, which proves the claim.

Measure δ_I can generally be used with methods that are based on distance measures between rows such as leaf order. It also works well with algorithms that are designed for metric TSP and rely on the triangle inequality in their approximation guarantees. Furthermore, we believe that this relation between spatial auto-correlation and neighborhood similarities helps to explain why neighborhood measures for computing orderings have been successful and popular in practice.

Relation to other adjacency measures. The measure of effectiveness by McCormick *et al.* [28, 29], when translated to $0-1$ valued (white-black) matrices, simply counts '1' for each vertical or horizontal black-black adjacency, and '0' otherwise. As such it is a less refined form of Moran's I. The stress measure proposed by Niermann [31], which uses Queen's adjacency, is related to Moran's I as well; the final score is the sum of all squared differences between a cell and its neighbors. As already mentioned above, diagonal adjacencies do not capture a graph property, they simply arise from two independent edges. Hence, this stress measure is less suitable to optimize adjacency matrices of graphs. Finally, the popular leaf order heuristic[3] uses the Euclidean distance between row vectors. For a 0-1 valued matrix, the Euclidean distance simply counts '1' for each black-white adjacency and then takes the square root of this sum. Orders produced with the leaf order algorithm hence tend to score very well on Moran's I.

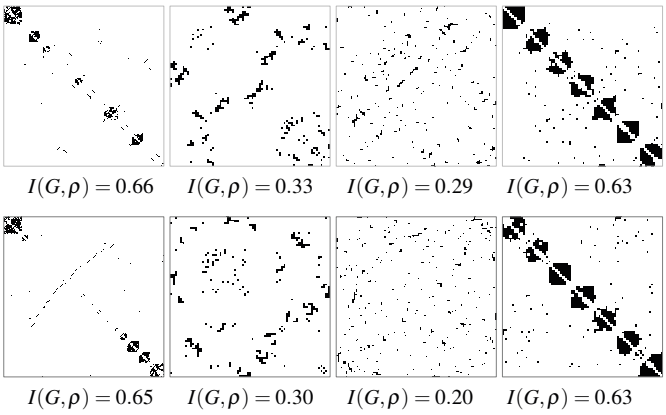

Fig. 5. Optimizing Moran's I using NN-2OPT (top row) and leaf order (bottom row), matrices from [8].

Optimizing Moran's *I*. As mentioned in Section 3, every adjacency measure is optimized via a traveling salesperson path. Since TSP is NP-hard to compute, we implemented the Nearest Neighbor (NN) heuristic using δ_I and further optimized the results using the 2-OPT algorithm [13] as long as Moran's I improves by more than 0.0001. Figure 5 shows the results for the four running-example matrices from the survey paper by Behrisch *et al.* [8]; each was computed in less than half a second. For comparison, we also include the results using leaf order with the L_2 metric. We see that NN-2OPT manages to achieve higher quality in terms of Moran's I.

4 Computing Simultaneous Orderings

There are various ways in which a simultaneous ordering can be computed. In the literature, we find two approaches that rely on computing an ordering for a single graph. A simple method, used by both Cubix [3] and MultiPiles [2], to compute an ordering is to base it on a single graph, using any algorithm; this ordering is then simply applied to the other graphs as well. However, we do not consider such an approach to really address the simultaneous ordering problem, as it uses the structure of only a single graph.

Below we review what we refer to as the *union* approach that is suggested by MultiPiles [2] for the leaf order method and apply it to the barycenter method as well. We then propose our new *collection-aware* approach which overcomes the loss of information that arises in the union approach (see also Figure 1), and show how to apply it to the leaf order method and the barycenter method. As we extend these methods, below is a brief summary of their basic steps (see also the survey [8] for details on these methods).

Leaf order. The leaf order method [4] computes an ordering in three stages: first, the distance between each pair of vertices is determined, based on some distance measure $\delta(G,u,v)$; second, this information is used to construct a hierarchical clustering on the vertices – our implementation is built on reorder.js[4] which uses greedy complete-linkage clustering here; third, the ordering ρ is computed that minimizes $\sum_{i=1}^{n-1}\delta(G,\rho(i),\rho(i+1))$ and "adheres" to the clustering tree. "Adhering" here means that the ordering matches the order in which an in-order traversal visits the leaves of the tree; effectively, we can choose for each internal node of the tree which of its two children will be the left and the right child. Note that this choice is for every node and thus allows completely reversing the leaves in any subtree. As not all permutations adhere to a given hierarchy, this method avoids the complexity that is inherent in the TSP formulation; indeed, this problem can be solved optimally in an efficient manner [11]. Often, this

[3] "[leaf order] consistently generates excellent results visually" Fekete [17]

[4] `https://github.com/jdfekete/reorder.js/`

approach is implemented using for δ some measure of (dis)similarity between $N(G,u)$ and $N(G,v)$.

Barycenter method. The barycenter method [14, 19, 27] (see also the survey [8]) focuses on optimizing "crossings", a standard measure of quality in traditional node-link drawings of graphs. The number of crossings incurred by an ordering ρ of $G=(V,E)$ is defined as follows: we duplicate each vertex v into v^0 and v^1, and draw v^i at $(\rho(v),i)$; we then draw every edge $(u,v)\in E$ as two line segments, u^0v^1 and v^1v^0; the number of crossings caused by these line segments (excluding common endpoints) is the number of crossings that ρ incurs.

The basic barycenter method works in two phases. First, starting from an arbitrary ordering, the ordering is repeatedly updated by sorting the vertices according to the median rank of its neighbors. This is done at least for a fixed number of iterations, possibly followed by additional iterations until the number of crossings no longer decreases ("convergence"). Second, a postprocessing step is applied which tries to swap adjacent vertices in the ordering, while such a swap reduces the number of crossings further.

4.1 The union approach

With MultiPiles, Bach *et al.* [2, Section 4.7] suggest an approach to computing a simultaneous ordering: "MultiPiles can calculate a global ordering which tries to find a topological clustering across all matrices". The paper itself provides no further detail on this method, but as their code is open source[5], we were able to extract the exact algorithm. What follows is first our general interpretation of their approach, followed by their exact implementation on the leaf order heuristic.

The *union* approach takes the (weighted) union over all graphs in $\mathscr{G}$ to arrive at a graph H on the same vertex set V, where each edge has a weight corresponding to the number of graphs of $\mathscr{G}$ it occurs in. Effectively, it is the sum over all 0-1 adjacency matrices. We may apply any ordering algorithm to H and apply the resulting ordering to all graphs in $\mathscr{G}$. For a vertex v, $N(H,v)=\sum_{G\in\mathscr{G}}N(G,v)$ is now a vector of length n where each entry is an integer in $\{0,\dots,k\}$, reflecting the weight of the associated edge.

The drawback of the union approach is a potential information loss, as H does not store information on which of the underlying graphs the edges actually occur in. In the extreme case, H could be a complete graph with unit weights for all edges, even though the underlying graphs have structures; imagine for example, two cliques in one graph and a biclique connecting the vertices of the two cliques in a second graph. In this case, all neighborhoods are identical and as such, there is no information left in H to inform a suitable simultaneous order. This is in fact our example in Figure 1, though we introduced some "noise" in this figure. The advantage is its simplicity: techniques that inherently work on (or straightforwardly generalize to) weighted graphs can readily be applied. It could in principle also be applied to techniques for unweighted graphs, but this only exacerbates the information loss.

Union leaf order. MultiPiles [2] applies the union approach to the leaf order method. This algorithm readily works on weighted graphs, and all it needs is some choice of measure δ. The MultiPiles implementation uses $\delta(u,v)=L_2(N(H,u),N(H,v))=L_2(\sum_{G\in\mathscr{G}}N(G,u),\sum_{G\in\mathscr{G}}N(G,v))$, that is, the Euclidean distance between neighborhoods. In general, we may of course use other measures instead of L_2 in the same fashion. We observe that squaring has no effect on comparisons of distances and thus does not affect the result of the complete-linkage clustering; it may however alter the eventual ordering, since comparisons between sums of distances may change.

But we may also use, for example, a Moran's I-based metric. We shall refer to it in this context also as δ_I, but observe that it is not simply counting the number of black-black and white-white adjacencies. Instead, we compute $\delta_I(u,v)=-\sum_{x\in V}(w(u,x)-\overline{w})(w(v,x)-\overline{w})$ where $w(a,b)$ is the weight of edge (a,b) in H, i.e., the number of graphs in $\mathscr{G}$ it occurs in, and $\overline{w}$ is the average weight over all pairs. Effectively, this is the original Moran's I formula where we forego the normalization terms as they are all the same for a given matrix; we multiply this by -1 as to obtain a distance function. As the implementation relies only on comparisons and sums, the negative values do not cause issues in this method – though we could factor in the normalization and add 1 as we did for δ_I in Section 3 to obtain a nonnegative measure that satisfies triangle inequality.

We observe the loss of information here as follows. Whereas indeed similar neighborhoods across the graphs lead to similar sums, the converse is not true: similar sums do not imply similar neighborhoods across the graphs. As such, the method may promote putting two vertices adjacent in the ordering that look similar in H but are actually not similar in any of the individual graphs.

Union barycenter. The barycenter method is typically applied in an unweighted setting. However, the crossing measure readily generalizes to a weighted setting. That is, rather than counting the number of crossings, we multiply the weights of two intersecting edges and sum the result of all intersecting pairs. We refer to this as the union barycenter.

We observe the information loss here as follows. Whereas crossings in one of the graphs contribute to the cost of an ordering, a counted crossing need not actually be present in any of the graphs in the collection. That is, two edges may cross in the union, but not occur together in any graph, and as such incorrectly contribute to the cost.

Though the second stage is readily affected by this change in measure, we note that it does not truly affect the first stage, beyond testing convergence: the ordering by median ranks of neighbors remains the same. We could apply a weighted median instead, or revert to using the actual (weighted) barycenters. As such, if this is pursued, it may be worthwhile to revisit this idea and in fact compute with the actual weighted barycenter. However, such extension is beyond the scope of our work here, as we focus on avoiding the loss of information that the union approach incurs.

4.2 The collection-aware approach

We propose a new approach to computing simultaneous orderings, one that is *collection-aware*. That is, we push the graph collection actually into the algorithms that compute orderings as much as possible. Specifically, when an algorithm makes decisions about the quality of an ordering, or how to modify an ordering based on neighborhoods, we now base this information on all graphs, rather than some aggregated form. In this manner, we may prevent the information loss that the union approach incurs. How this is done specifically, naturally depends on the algorithm under consideration.

Collection-aware leaf order. As all information is captured in the pairwise distances between vertices, it suffices here to ensure that δ is collection-aware. That is, $\delta(u,v)$ should be low if u and v have similar neighborhoods in many graphs, *and vice versa*. Whereas the union approach gives the former, it does not succeed in the latter.

Rather than $\delta(u,v)=L_2(\sum_{G\in\mathscr{G}}N(G,u),\sum_{G\in\mathscr{G}}N(G,v))$, a collection-aware interpretation would apply these operators the other way around: $\delta(u,v)=\sum_{G\in\mathscr{G}}L_2(N(G,u),N(G,v))$.

Again, we observe that we may replace L_2 with other distance measures as well. Specifically, observe that L_2^2, using the squared Euclidean distance, is different from using L_2 now also in the clustering step. Also, we can again use the Moran's I-based metric δ_I where we can now rely on the metric formulation as presented in Section 3.

Note that this provides no computational overhead in asymptotic terms. As adding vectors is typically slightly faster than computing distances, we may expect a slight overhead. Yet, this is likely overshadowed by the clustering and leaf order stages of the algorithm and as such barely noticeable.

Collection-aware barycenter. In our collection-aware barycenter implementation, we go back to counting crossings (as we assume unweighted graphs in $\mathscr{G}$), but we now do so for each graph separately and then take the sum over all graphs. This makes the second stage collection-aware.

However, as before, changing the way we measure crossings does not readily influence the procedure in the first stage, beyond testing convergence, as this is based purely on the ranks of the neighbors. In a collection-aware approach, however, we should strive to base these

[5] `https://github.com/benjbach/multipiles`, accessed Feb. 2021.

decisions on ordering on all graphs. Rather than using all neighbors in all graphs to determine the target rank of a vertex, we determine the target rank per graph, and aggregate this information. We implement this as follows. For every vertex, we compute the median rank of its neighbors in each graph in $\mathscr{G}$ separately to arrive at a set of median ranks. Subsequently, we compute the median of these median ranks. In the event that a vertex has no neighbors at all in a graph, it is omitted from the set of median ranks. Sorting then proceeds as before, but is now based on the median of medians instead.

5 Experiments

Here we present the results of a brief experimental evaluation that aims to quantify how much improvement our collection-aware approach provides compared to the union approach, via various implementations. Moreover, we also investigate the use of Moran's I as a distance measure for optimization.

5.1 Setup

Algorithms. We have two approaches to solving the simultaneous ordering problem: *(U)* union and *(C)* collection-aware. We have shown how to implement these approaches on two base algorithms: *(LO)* leaf order and *(BC)* barycenter. Finally, LO can work with various distance metrics. In particular, we use Euclidean distance L_2 and the Moran's I-based metric δ_I, as well as their squared variants L_2^2 and δ_I^2. We name each algorithm using a concatenation of these abbreviations. For example, U-LO-L_2 refers to the union leaf order method using L_2, effectively the implementation suggested by MultiPiles; C-LO-δ_I^2 refers to the collection-aware leaf order method using δ_I^2 as a metric; C-BC refers to the collection-aware barycenter method. Our implementation of the ten resulting algorithms is openly available on GitHub[6].

Data. We test our algorithms with three datasets, chosen to obtain a variety of characteristics, ranging from many graphs with few vertices, to few graphs with many vertices; Table 2 provides some basic statistics.

FLT The "flashtap" data that was used for MultiPiles [2][7]. It represents functional brain connectivity in a Parkinson's disease study.

SCH Social interaction between children and teachers at a primary school [20, 33][8].

VIS Publications data in the InfoVis conference [25][9]. We construct a graph on the authors for each year from 2015 to 2020, where an edge between two authors is included if they had a joint paper in this period. We include only authors with publications in at least three years and their co-authors.

The density m/n^2 of these graphs varies (Table 2). Arguably, graphs with low density such as the VIS co-authorship network are better visualized using other visual idioms, for instance, node-link diagrams or hybrid visualizations, but we include them here to investigate our algorithms under diverse circumstances. For each dataset we also measured the change Δ between graphs, which is the number of cells in the matrix that change their value. Though expressed as a fraction of the entire matrix (where FLT changes most, and VIS the least), these values can also be interpreted with respect to the edge density (in which case VIS changes most, relatively speaking, and FLT the least).

Quality measures. We measure the quality of the resulting matrix orderings via the current standard of linear arrangement as well as with Moran's I. Whereas Moran's I is normalized to $[-1, 1]$ by definition, linear arrangement is not. We normalize a linear arrangement value a to $1 - a/M$, where M is the maximum value over all algorithms and graphs of the same dataset. Hence, the *normalized linear arrangement* is a value in $[0, 1]$, where 1 is the best performing matrix, such that in both measures, higher values correspond to higher quality.

Note that we measure our metrics for each graph in the collection separately. As such, we obtain a distribution of ordering quality, for each algorithm-dataset combination. We focus on the minimum, average and median quality. The latter two simply because we aim for high overall quality. The minimum, however, is also specifically useful since it gives us an idea of the worst matrix in the collection. Ideally, we would want to avoid slightly improving many graphs at the expense of greatly reducing the quality of one graph.

Table 2. Overview of datasets. k: number of graphs in collection; n: number of vertices in each graph; m: number of black cells in the matrix; Δ: number of changing edges between graphs. m and Δ are given as a percentage of the number of cells in the matrix (n^2), and we provide the mean μ and standard deviation σ.

Dataset	k	n	m/n^2 (%)		Δ/n^2 (%)	
			μ	σ	μ	σ
FLT	96	29	44.06	11.31	19.67	20.61
SCH	17	242	5.15	1.48	3.31	3.68
VIS	6	536	0.28	0.10	0.21	0.26

5.2 Results

Figure 2 in Section B of the supplementary material shows the results for each algorithm on each dataset in terms on the two measures. Our primary observation here is that using squared versions of metrics changes the results, but does not seem to considerably change the overall performance (in terms of means, medians and minima). Refer to the tables in Section B of the the supplementary material for precise differences; averaged over all datasets the differences do not exceed

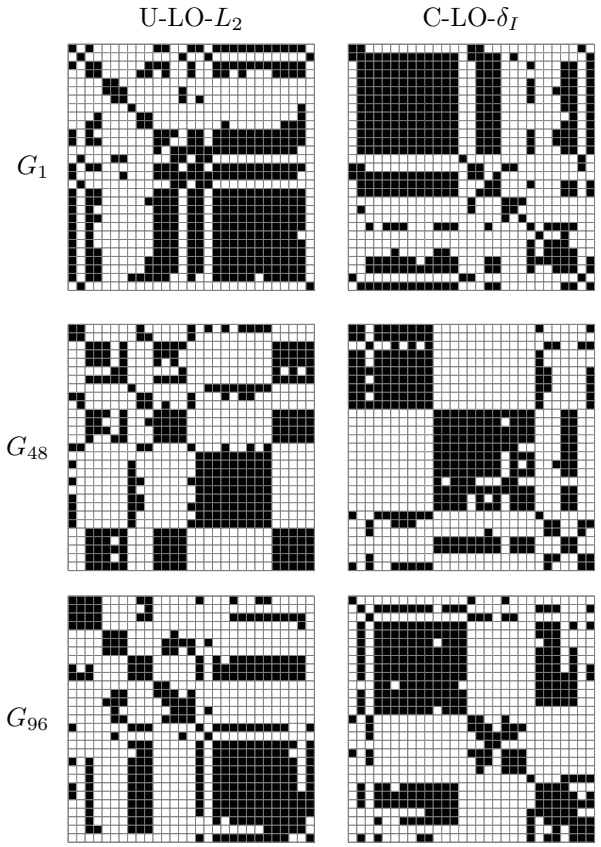

Fig. 6. A comparison of the state-of-the-art (U-LO-L_2) against our main contribution (C-LO-δ_I) on timesteps G_1, G_{48} and G_{96} of the FLT dataset.

[6] https://github.com/nvbeusekom/reorder.js

[7] https://aviz.fr/~bbach/multipiles/, accessed March 2021.

[8] http://www.sociopatterns.org/datasets/primary-school-temporal-network-data/, accessed March 2021.

[9] https://sites.google.com/site/vispubdata/home, accessed March 2021 at version 9.02.

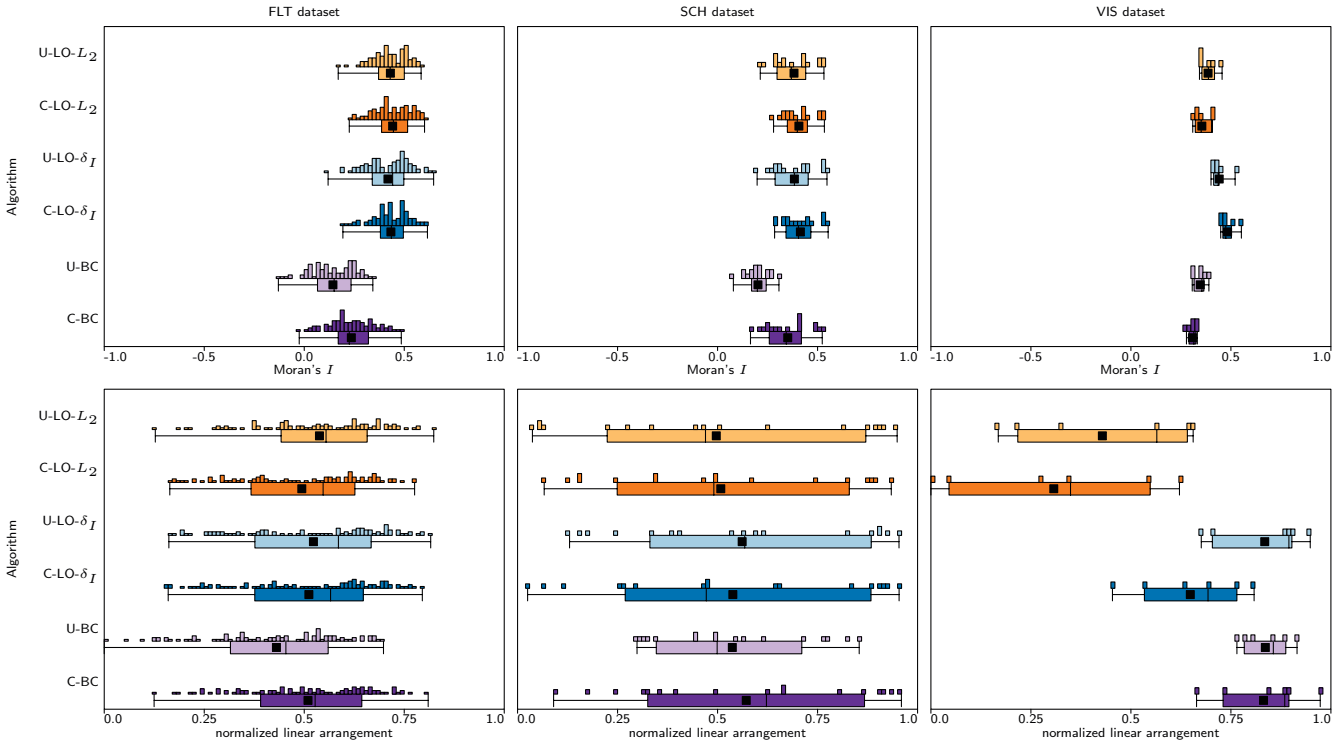

Fig. 7. Performance of the six main algorithms considered for each of the datasets (columns) in terms of Moran's I (top row) and Linear Arrangement (bottom row). For both, higher values indicate higher quality.

0.02 for Moran's I. A notable outlier is the differences in the VIS dataset between using L_2^2 and L_2 for collection-aware leaf order, though we do not investigate this further.

Here, we focus on the six algorithms that do not use the squared distance functions. In Section B of the supplementary material, Figures 3 to 16, we show all matrix visualizations obtained through the six ordering algorithms for the FLT dataset. Figure 6 shows an excerpt of the FLT dataset for U-LO-L_2 (as proposed in [2]) and C-LO-δ_I (our proposed algorithm). Clearly, each of the matrices in isolation can be improved, but each column uses but one ordering by design, which must work well for all 96 graphs in the collection. We may observe that block patterns feature more strongly in the C-LO-δ_I result; the U-LO-L_2 result partitions these into smaller block patterns complemented by matching off-diagonal blocks.

A summary of the results of the six algorithms are shown in Figure 7. Below, we discuss various comparisons based on Moran's I, before briefly discussing linear arrangement.

(U/C)-LO-L_2. Let us first compare the performance of the union and collection-aware implementations of leaf order, using the standard Euclidean distance. Generally, we see very similar performance between these methods. For FLT and SCH, the median and average improve slightly in the collection-aware variant (difference of 0.02 and 0.03 for median, and 0.01 and 0.02 for average); the minimum improves more, 0.06 (FLT) and 0.07 (SCH).

For VIS we actually see a decrease, roughly -0.03 for each statistic. We attribute this to the overall sparsity of the graphs. The Euclidean distance may weigh white-white adjacencies too heavily, whereas they contribute little to improving Moran's I. Using a union gives for a slightly denser graph and thus may help in finding some structure in this specific case.

In this comparison, we conclude that there indeed is some information loss in the union approach. The medium-density graph improves most, hinting that very dense graphs and very sparse graphs suffer less from the information loss that complementary patterns may give – though this may benefit more structural investigation using more (controlled) datasets.

(U/C)-LO-δ_I. Let us now repeat the above comparison, but with algorithms based on the Moran's I metric δ_I. We now see an improvement for each dataset. On average, the minimum Moran's I improves by 0.07 and the median and average by 0.02 and 0.03 respectively. Most different from the previous comparison is the improvement for VIS. From this we conclude that the collection-aware approach actually does help to make find structures in very sparse graphs. That is, the decrease in performance can be attributed to the use of the Euclidean distance rather than the collection-aware approach itself.

C-LO-(L_2/δ_I). From the above, we may conclude that our collection-aware adaptations yield benefits in terms of Moran's I. So, we now briefly consider the choice of metrics between the collection-aware implementations. We see that on average (in terms of minimum, average and median) using δ_I improves Moran's I by 0.04 compared to using L_2. Some improvement was to be expected, since the method now (heuristically) optimizes the quality measure directly. Yet, the effect is somewhat small in that light.

Primarily, we can conclude that the Euclidean distance serves well as a proxy. We see that this proxy suffers in case of very sparse graphs (VIS), where using δ_I improves considerably more (differences of 0.12 to 0.14) than in the other datasets. The equal treatment of types of adjacencies in the Euclidean distance does seem to hinder it, in obtaining high Moran's I.

(U/C)-BC. Let us now briefly turn to the barycenter method. We see here again a strong difference between datasets. Whereas C-BC clearly outperforms U-BC (differences of 0.08 to 0.15) for FLT and SCH, the opposite it true for VIS. Again, we can attribute this to the sparsity of the graph. As many vertices have edges in only one graph, their ranks in the collection-aware method are based primarily on that one graph – which may interfere with other graphs. With the union approach, at least everything is treated centrally and thus the information in the slightly denser union graph is indeed a bit more rich.

Linear arrangement. Though it focuses on block patterns only and does not generally measure the degree of structure of the visualization, linear arrangement is the de-facto standard for automatically measuring ordering quality [8]. We briefly consider it for this reason.

For FLT, we may observe that the U-LO variants outperform C-LO approaches, though the performance between C-LO variants is mostly similar. For the barycenter method, this is rather the other way around: C-BC outperforms U-BC.

For SCH, we see more spread of the linear arrangement within one graph collection, but this may be an effect of the normalization. Though U-LO-δ_I still slightly outperforms C-LO-δ_I, we now see that C-LO-L_2 offers a slight improvement on U-LO-L_2. C-BC still outperforms U-BC in terms of median and average, though its minimum is now considerably lower.

For VIS, we again see that the U-LO variants outperform C-LO approaches, now with a considerable difference. Interestingly, C-BC is now slightly under U-BC as well. We may attribute this again to the sparsity of the dataset.

We conclude that, for linear arrangement, the union approach is typically better than the collection-aware approach to leaf order. For the barycenter method, however, this appears to be opposite, unless the dataset is excessively sparse. This is, perhaps, not surprising since the crossing measure underlying the barycenter methods discourages "long edges" and thus off-diagonal cells. With the collection-aware approach, we see that this indeed avoids information loss, unless there is very little information to begin with – in which case the union method strengthens the signal of the little information that is available.

Overall comparison. Considering the above, we may conclude that our collection-aware adaptations have been successful in improving Moran's I and thereby the structure of the resulting matrix visualizations. We observe that C-LO-δ_I performs best over all methods, with higher minimum, median and average scores compared to all other methods. It specifically performs better on the minimum compared to other LO variants, and considerably improves upon BC variants. Whereas the barycenter method focuses (indirectly) more on block structures and thus linear arrangement, we do not see it consistently outperforming the leaf order methods even in this criterion. We would thus recommend the use of C-LO-δ_I as the most versatile algorithm for solving the simultaneous ordering problem.

6 Conclusion

In this paper we considered the problem of computing simultaneous orderings for graph collections. That is, given a set of graphs, compute a single ordering that works well for visualizing each graph as a matrix. To automatically assess quality, we observed that patterns in the matrix can be seen as a form of spatial auto-correlation and thus proposed the use of Moran's I as a global measure of quality. Moran's I readily implies a distance metric between two vertices that can be used in algorithms such as leaf order, as an alternative to other common functions that measure adjacencies or neighborhood similarity.

Algorithmically, computing simultaneous orderings has received little attention. We generalized the *union* approach that is found in MultiPiles, but observed that this may lead to hiding structural information from the ordering algorithms. Instead, we proposed a generic *collection-aware* approach that avoids such loss of information and showed how to apply this approach to the common leaf order and barycenter methods.

Our experiments demonstrate that our collection-aware approach is effective, especially leaf order based on the Moran's I-inspired metric. Our collection-aware leaf ordering method using δ_I is the most versatile and consistently performs equally or better than the other algorithms, though the magnitude of improvement varies between datasets. For other algorithm comparisons, there is less consistency in relative performance, also interacting with dataset. Our results confirm that the potential information loss as sketched for the union approach indeed occurs, also in real datasets. Though it does not occur to such a degree to really cause arbitrary orderings as in hypothetical constructed cases, it nonetheless leads to inferior orderings when not accounting for the collection of graphs in the ordering algorithms.

Future work. There is not a clear metric, the optimization of which gives visually the best or most useful ordering, even for a single graph. Though we postulate that Moran's I can be effective here, further research is needed on how this concept actually relates to perceiving structure in matrix visualizations. Our focus was with algorithm-compatible measures, but such future work can likely leverage the work on magnostics [7]. Furthermore, it may be worthwhile to investigate the perceptual effects that improving Moran's I brings; for example, Beecham *et al.* [6] study the "just noticeable difference" of Moran's I and their experiment includes weight grids as well. It suggests a degree to which Moran's I must improve for an observer to reliably identify the improved ordering, though in our case, we have black-and-white and symmetric matrices, which may influence the perception.

With our work, we provide (to the best of our knowledge) the first explicit definition of the simultaneous ordering problem for visualizing graphs using matrices. Our general concept of making algorithms collection-aware is applicable to other alternatives. Especially distance-based approaches like leaf order are easily amended. But we could also consider other ways in making algorithms collection-aware, and indeed, some algorithms may be more suitable than others for such adaptation. Can we, for example, in fact directly modify the clustering algorithm in leaf ordering, or change the objective function in its final stage, to explicitly optimize e.g., the smallest sum?

Finally, our experiments show differences depending on the number of graphs and vertices in the dataset. Whereas for a few matrices (or a few MultiPiles) a single ordering can be effective, when using many matrices, one is bound to have to make concessions for many of them. As such, the question is perhaps, whether we could permit small changes to the ordering for each matrix, to better highlight the structure in each graph. In our next steps, we plan to address this challenge of "stable" orderings.

To conclude, perhaps the main take-away of our work here is that simultaneous matrix ordering is a complex algorithmic-visualization problem. We have now taken the first steps in bringing this to the fore and addressing the algorithmic challenges.

References

[1] B. Alper, B. Bach, N. Henry Riche, T. Isenberg, and J.-D. Fekete. Weighted graph comparison techniques for brain connectivity analysis. In *Proc. SIGCHI Conference on Human Factors in Computing Systems*, pp. 483–492, 2013. doi: 10.1145/2470654.2470724

[2] B. Bach, N. Henry-Riche, T. Dwyer, T. Madhyastha, J.-D. Fekete, and T. Grabowski. Small MultiPiles: Piling time to explore temporal patterns in dynamic networks. *Computer Graphics Forum*, 34:31–40, 05 2015. doi: 10.1111/cgf.12615

[3] B. Bach, E. Pietriga, and J.-D. Fekete. Visualizing dynamic networks with matrix cubes. In *Proc. SIGCHI Conference on Human Factors in Computing Systems*, pp. 877—886, 2014. doi: 10.1145/2556288.2557010

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

[5] F. Beck, M. Burch, S. Diehl, and D. Weiskopf. A taxonomy and survey of dynamic graph visualization. *Computer Graphics Forum*, 36(1):133–159, 2017. doi: 10.1111/cgf.12791

[6] R. Beecham, J. Dykes, W. Meulemans, A. Slingsby, C. Turkay, and J. Wood. Map LineUps: effects of spatial structure on graphical inference. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):391–400, 2016.

[7] M. Behrisch, B. Bach, M. Blumenschein, M. Delz, L. von Rueden, J.-D. Fekete, and T. Schreck. Magnostics: Image-based search of interesting matrix views for guided network exploration. *IEEE Transactions on Visualization and Computer Graphics*, 23:31–40, 01 2016. doi: 10.1109/TVCG.2016.2598467

[8] M. Behrisch, B. Bach, N. Henry Riche, T. Schreck, and J.-D. Fekete. Matrix reordering methods for table and network visualization. In *Computer Graphics Forum*, vol. 35, pp. 693–716, 2016.

[9] 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*, 37(3):625–662, 2018. doi: 10.1111/cgf.13446

[10] T. Bläsius, S. G. Kobourov, and I. Rutter. Simultaneous embedding of planar graphs. In R. Tamassia, ed., *Handbook of Graph Drawing and Visualization*, pp. 349–383. CRC Press, 2013.

[11] U. Brandes. Optimal leaf ordering of complete binary trees. *Journal of Discrete Algorithms*, 5(3):546–552, 2007.

[12] S. Cabello, M. J. van Kreveld, G. Liotta, H. Meijer, B. Speckmann, and K. Verbeek. Geometric simultaneous embeddings of a graph and a matching. *Journal of Graph Algorithms and Applications*, 15(1):79–96, 2011.

[13] G. A. Croes. A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812, 1958.

[14] P. Eades and N. C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, 1994.

[15] N. Elmqvist, T.-N. Do, H. Goodell, N. Henry, and J.-D. Fekete. ZAME: Interactive large-scale graph visualization. In *Proc. 2008 IEEE Pacific Visualization Symposium*, pp. 215–222, 2008. doi: 10.1109/pacificvis.2008.4475479

[16] A. Estrella-Balderrama, E. Gassner, M. Jünger, M. Percan, M. Schaefer, and M. Schulz. Simultaneous geometric graph embeddings. In *Proc. 15th International Symposium on Graph Drawing*, LNCS 4875, pp. 280–290, 2008.

[17] J.-D. Fekete. Reorder.js: A JavaScript Library to Reorder Tables and Networks. In *Abstr. 2015 IEEE VIS posters*, 2015. Available at `https://hal.inria.fr/hal-01214274/file/reorder.pdf`.

[18] J. J. Fowler, M. Jünger, S. G. Kobourov, and M. Schulz. Characterizations of restricted pairs of planar graphs allowing simultaneous embedding with fixed edges. *Computational Geometry*, 44(8):385–398, 2011.

[19] E. R. Gansner, E. Koutsofios, S. C. North, and K.-P. Vo. A technique for drawing directed graphs. *IEEE Transactions on Software Engineering*, 19(3):214–230, 1993.

[20] V. Gemmetto, A. Barrat, and C. Cattuto. Mitigation of infectious disease at school: targeted class closure vs school closure. *BMC infectious diseases*, 14:Article no. 695, 2014. doi: 10.1186/s12879-014-0695-9

[21] M. Ghoniem, J.-D. Fekete, and P. Castagliola. A comparison of the readability of graphs using node-link and matrix-based representations. In *Proc. IEEE Symposium on Information Visualization*, pp. 17–24, 2004. doi: 10.1109/INFVIS.2004.1

[22] B. Haeupler, K. Jampani, and A. Lubiw. Testing simultaneous planarity when the common graph is 2-connected. *Journal of Graph Algorithms and Applications*, 17(3):147–171, 2013. doi: 10.7155/jgaa.00289

[23] N. Henry Riche and J.-D. Fekete. MatLink: Enhanced matrix visualization for analyzing social networks. In *Proc. 13th IFIP TC13 International Conference on Human-Computer Interaction*, LNCS 4663, pp. 288–302, 2007.

[24] M. I. Hossain, V. Huroyan, S. Kobourov, and R. Navarrete. Multi-perspective, simultaneous embedding. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1569–1579, 2021. doi: 10.1109/TVCG.2020.3030373

[25] P. Isenberg, F. Heimerl, S. Koch, T. Isenberg, P. Xu, C. Stolper, M. Sedlmair, J. Chen, T. Möller, and J. Stasko. vispubdata.org: A metadata collection about IEEE Visualization (VIS) publications. *IEEE Transactions on Visualization and Computer Graphics*, 23(9):2199–2206, 2017. doi: 10.1109/TVCG.2016.2615308

[26] J. K. Lenstra and A. H. G. R. Kan. Some simple applications of the travelling salesman problem. *Operational Research Quarterly (1970–1977)*, 26(4):717–733, 1975.

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

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

[29] 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.

[30] P. A. P. Moran. Notes on continuous stochastic phenomena. *Biometrika*, 37(1/2):17–23, 1950.

[31] S. Niermann. Optimizing the ordering of tables with evolutionary computation. *The American Statistician*, 59(1):41–46, 2005. doi: 10.1198/000313005X22770

[32] Z. Shen and K.-L. Ma. Path visualization for adjacency matrices. In *Proc. 9th Joint Eurographics / IEEE VGTC Conference on Visualization*, 2007. doi: 10.2312/VisSym/EuroVis07/083-090

[33] J. Stehlé, N. Voirin, A. Barrat, C. Cattuto, L. Isella, J.-F. Pinton, M. Quaggiotto, W. Van den Broeck, C. Régis, B. Lina, et al. High-resolution measurements of face-to-face contact patterns in a primary school. *PloS one*, 6(8):e23176, 2011. doi: 10.1371/journal.pone.0023176

[34] J. S. Yi, N. Elmqvist, and S. Lee. TimeMatrix: Analyzing temporal social networks using interactive matrix-based visualizations. *International Journal of Human–Computer Interaction*, 26(11–12):1031–1051, 2010. doi: 10.1080/10447318.2010.516722