SEQUENCE BRAIDING: Visual Overviews of Temporal Event Sequences and Attributes

Sara Di Bartolomeo Yixuan Zhang Fangfang Sheng Cody Dunne

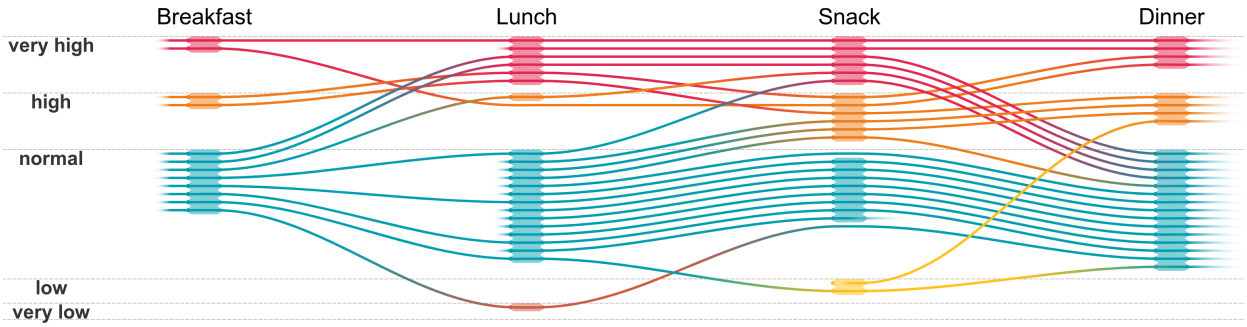

Fig. 1: An example of SEQUENCE BRAIDING applied to blood glucose and meal data of a patient with type 1 diabetes. This temporal event sequence and attribute data is modeled using a layered directed acyclic network, where each node ⬭ represents an individual event (meal) and contiguous edges connect the events from a single day in sequence. Nodes are aligned into columns based on their event (meal) type and into rows based on attribute values, in this case quantitative blood glucose groups. Nodes are also color-coded by attribute group: ■ very high, ■ high, ■ normal, ■ low, and ■ very low.

Abstract—Temporal event sequence alignment has been used in many domains to visualize nuanced changes and interactions over time. Existing approaches align one or two sentinel events. Overview tasks require examining all alignments of interest using interaction and time or juxtaposition of many visualizations. Furthermore, any event attribute overviews are not closely tied to sequence visualizations. We present SEQUENCE BRAIDING, a novel overview visualization for temporal event sequences and attributes using a layered directed acyclic network. SEQUENCE BRAIDING visually aligns many temporal events and attribute groups simultaneously and supports arbitrary ordering, absence, and duplication of events. In a controlled experiment we compare SEQUENCE BRAIDING and IDMVis on user task completion time, correctness, error, and confidence. Our results provide good evidence that users of SEQUENCE BRAIDING can understand high-level patterns and trends faster and with similar error. A full version of this paper with all appendices; the evaluation stimuli, data, and analysis code; and source code are available at `osf.io/mq2wt`.

Index Terms—Temporal event sequence visualization, network visualization, algorithm, evaluation

1 INTRODUCTION

Temporal event sequence data is common across many domains such as health care, activity tracking, sports analytics, cybersecurity incident analysis, and web analytics [13]. The gathering and analysis of this data is important to help users make sense of what has happened and to aid in future decision making. Analysts are often interested in whether there are general patterns and trends in the data. These patterns can be related to the *sequence*, such as the common precursor, co-occurring, and aftereffect events around an event of interest, but also the *attributes* associated with these events [13].

For sequence analysis, *sentinel event alignment* has been used effectively for clearly showing patterns around one [55] or two [63] events of interest. However, overview tasks may require examining all possible alignments of interest. Existing alignment approaches require interaction and time or juxtaposing many separate aligned visualizations for the analysts to build such an overview.

Multiple coordinated visualizations help illustrate overviews of event attributes [56]. However, it is challenging to examine the linkage between related data in juxtaposed views due to limited space for each individual visualization. Moreover, coordination in juxtaposed views requires substantial interactions [24]. Existing approaches for combining event sequence and attribute visualization include separating event types for quantitative attribute groups (e.g., [32, 55]) and using color to encode aggregated attributes (e.g., [39]). Yet, current approaches do not sufficiently support visualizing event sequence data and attributes together. Therefore, we need new holistic overviews that directly integrate attributes as a first-class citizen in sequence visualization.

We aim to address these issues by aligning a large number of temporal events as well as aggregating attribute groups in a single overview-first visualization. The main contributions of this paper are:

1. SEQUENCE BRAIDING, *a novel network visualization technique for temporal event sequences and attributes*. Many temporal events and attribute groups are aligned simultaneously to support overview analysis of sequence and attribute data, while still preserving the ability to follow individual sequences.
2. An *n-layer network layout algorithm with grouping and group ordering constraints* for creating SEQUENCE BRAIDING visualizations, focused on intersection reduction and building on the barycentric approach.
3. *The results of a comparative evaluation* based on user task completion time, correctness, error, and confidence which evidences the efficacy of SEQUENCE BRAIDING.

The structure of the paper is as follows: First, we motivate SEQUENCE BRAIDING using a design case study of type 1 diabetes

- *Sara Di Bartolomeo, Yixuan Zhang, Fangfang Sheng, and Cody Dunne are with Northeastern University. E-mail: {dibartolomeo.s, zhang.yixua, sheng.f, c.dunne}@northeastern.edu.*
- *Yixuan Zhang is now with Georgia Institute of Technology. Email: yixuan@gatech.edu.*

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

treatment, though our approach is general and we show examples of effective usage in other domains. We next discuss related work on temporal event sequence visualization and n-layer graph intersection reduction approaches. From our case study, we then gather design requirements that informed the design, implementation, and layout algorithm behind SEQUENCE BRAIDING. Finally, we discuss our comparative evaluation and the implications of our work for researchers and practitioners. A full version of this paper with all appendices and supplemental material is available at `osf.io/mq2wt` and examples are online at the SEQUENCE BRAIDING `website`[1].

1.1 Motivating Case Study: Type 1 Diabetes

We introduce our visualization technique and algorithms in the context of a design case study on type 1 diabetes treatment.

Type 1 diabetes is an incurable autoimmune disease that affects the patients' ability to regulate their blood glucose levels [58]. The only treatment is for patients or their caregiver(s) to take over the role of their body in monitoring their blood glucose and administering food and injecting insulin to shift their glucose levels up and down, respectively [58]. Our discussion here of type 1 diabetes treatment is necessarily limited. For more information please see [43, 58, 63].

Blood glucose levels can be monitored by periodically testing blood with a glucose meter (SMBG) [44], e.g., daily before eating, or with a continuous glucose monitor (CGM), which provides frequent measurements on the order of every 5 minutes throughout the day [8]. Proper titration of the amount of insulin to administer for treatment and for food, e.g, at a meal, requires careful data analysis that can happen on a monthly or even weekly schedule. This titration can even be done independently by the patients themselves [58, 63]. Regardless of who performs the analysis, it requires that the patient or their caregiver(s) have logged events such as food intake and insulin injection as well as attributes like blood glucose levels at those events. Analyses are generally limited to exploring the previous 14 days of data due to some combination of logging burden, and relevance to current care and can last 15–60 minutes in the clinical setting [63]. However, research in diabetes treatment suggests that clinicians also need information over a longer duration (e.g., three-month clinical visit data) to evaluate how well patients control their diabetes condition [58]. Current approaches do not sufficiently support examining event sequences within a long period of time. Our approach attempts to address this issue.

1.1.1 Type 1 Diabetes Data and Tools

For this study we use data from an anonymized single elementary-school-aged child with type 1 diabetes which was collected by the patient and their caregivers over 877 days. It includes timestamped pre-meal blood glucose measurements (SMBG or CGM), the meal type, the grams of carbohydrates ingested with each meal, and the injected insulin amount in units (U-100, 1 unit= 0.01 mL). Blood glucose measurements are recorded using mg/dL in the USA, also used herein, with target values falling in 80–180 mg/dL. Excursions from this range can result in both imminent and long-term medical complications [2]. In the UK and many other countries, mmol/L is used with a target of 4.4–10 mmol/L. Roughly, mg/dL $= 18\times$ mmol/L. We contribute this data as part of our supplemental material at `osf.io/mq2wt`.

Software tools are available for visualizing diabetes data, e.g., using line charts and high-level aggregates such as time-in-range, but few provide effective multi-day event sequence and attribute overviews which are necessary for insulin titration [63] and none support simultaneous alignment of temporal events and attribute groups for overview analysis. For an overview of diabetes visualizations see Zhang et al. [63].

1.1.2 Type 1 Diabetes Case Study

A subset of this data is shown using SEQUENCE BRAIDING in fig. 1. Since the data is human-generated, the logs can have anomalies in them. While our system is resilient to anomalies, larger human introduced errors such as missing types or attributes are impossible to automatically handle and are therefore discarded. We group the records according to their blood glucose values into 5 predefined ordinal groups — ■ very high, ■ high, ■ normal, ■ low, and ■ very low. The logs are temporally folded by day [13].

[1] `visdunneright.github.io/sequence_braiding/docs/`

2 RELATED WORK

2.1 Temporal Event Sequence Visualization

Much has been written on event sequence visualization given the widespread availability of data and variety of visual representations possible. There have been attempts at defining a design space for event sequences, e.g., Du et al. [13], Chen et al. [10], and Tang et al. [50].

As we have discussed, *aligning* temporal event sequences by *sentinel events* of interest can help display patterns of precursor, co-occurring, and aftereffect events [13]. Single-event alignment was pioneered by LifeLines2 [55] for clinical tasks and has been adopted in the design of, e.g., CareCruiser [21], LifeFlow [60], EventFlow [32], and Peekquence [28]. Single-event alignment has since been applied to many diverse domains [13]. Recently, in IDMVis Zhang et al. [63] extended this approach to support dual-event alignment and provide ways for scaling the intermediate timeline. The value of dual-event alignment for analyzing intermediate events was demonstrated in a controlled experiment by Zhang et al. [64]. These alignment approaches display either an exact or relative time delta around sentinel events.

In order to provide overviews of the entire set of sequences, several tools have used *layered flow graph visualizations*. LifeFlow [60] and EventFlow [32] have used hierarchical icicle plots to aggregate events at multiple layers out from the aligned sentinel event. BaobabView [53] used a comparable approach for visualizing decision trees. Escaping the limits imposed by a strict hierarchy, Outflow [59], Care Pathway Explorer [40], used node-link visualizations of directed acyclic graphs and combined events of a type at each layer, akin to interactive Sankey diagrams [41], TopicStream [31], or the adjacency matrix representation in MatrixWave [66]. However, there are limitations to layered flow graph approaches: (1) there exists a fan-out effect as the number of possible subsequences expands each layer from the alignment, (2) each layer can contain many different types of events making high-level observation difficult, and (3) by necessity the time delta between events on subsequent layers is aggregated or omitted in static views. In SEQUENCE BRAIDING we address issues (1) and (2), and retain (3).

Like our work, egoSlider [61] and Set Streams [1] employ layered node-link graphs, with the objective of using graph topology to highlight patterns in the data. Both use fixed rank assignment by binning events by time period, while SEQUENCE BRAIDING uses an event alignment algorithm. Moreover, we improve upon the simplistic intersection reduction methods used. Interestingly, Hive Plots [26], pursues the same objective with a radial axis that represents a layer for each structurally-derived attribute of interest, with nodes positioned by the attribute value. This limited structural grouping can produce visual signatures to help make patterns stand out. These node-link visualizations share issues (2) and (3) with flow graphs, mentioned above.

Conversely, storyline visualizations such as described by Tanahashi et al. [49] and Tang et al. [50] are designed to represent the relational dynamics of individual entities, each represented by a line. While the overall goal is similar to ours, storylines make use of vertical position only for showing approximate groupings of entities at time points and not exact event or attribute alignment as is our goal with SEQUENCE BRAIDING. Likewise, it is possible to directly visualize multiple sequence alignment results as in Sequence Bundles [27], though it is not designed for showing attribute and event overviews.

As our domain case study is related to type 1 diabetes, we should point out the survey of electronic health record visualizations by Rind et al. [42] and the five Ws for healthcare informatics visualization [65]. However, as SEQUENCE BRAIDING is a general-purpose event sequence and attribute visualization, we will not further detail less relevant domain visualizations.

2.2 N-layer Intersection Reduction Approaches

The barycentric approach is a common heuristic-based method to reduce the number of intersections in a layered graph. The vertical positioning can be computed using either the mean [46] or the median [15] for the barycenter of incident nodes. The use of heuristics is common in

solving the intersection reduction problem [12, 52]. Gansner et al. [19] use another heuristic approach using the median, and cite Warfield [57] as a base for their work. In SEQUENCE BRAIDING, we use Gansner et al. as a base, adding a multiple sequence alignment algorithm for the rank assignment step and grouping constraints. Another approach is based on calculating and drawing the maximum level planar subgraph and reinserting the nonplanar edges at the end — with the nonplanar edges being the ones that introduce crossings [34, 35].

Heuristic approaches do not guarantee an optimal solution. It is possible to formulate the problem as a linear programming problem to find an optimal solution [18, 25, 62]. However, these methods are only suitable for small graphs. We discuss this further in appendix B.

N-layer intersection reduction techniques have been applied to a multitude of different contexts that required different constraints and have different objectives. Waddle [54] proposes the problem of displaying data structures on graphs with nodes that include ports. Metro map layout algorithms often include layered intersection reduction approaches aimed at better readability [4, 37]. For a more complete survey, please see Tamassia's book on graph drawing [48].

3 DESIGN REQUIREMENTS

We first developed requirements using Adrienko & Adrienko's general task typology [3] and a hierarchical task abstraction by Zhang et al. [63].

DR 1: Support many-event alignment — The visualization should support many-event alignment such that general patterns across event sequences can be clearly displayed in a single overview-first visualization. A single visualization avoids needing interaction and time or juxtaposing many aligned visualizations for creating a mental overview.

DR 2: Provide an overview of the distribution of attribute values at each event — The visualization should let the reader have a sense of the distribution of values among events in each event type — e.g., being able to discern the prevalent value, and in what percentage.

DR 3: Allow holistic analysis of a set of sequences — Attribute and sequence overviews should be closely tied for a more holistic analysis.

DR 4: Support arbitrary ordering, absence, and duplication of events — Human-generated data is messy and prone to errors. Our visualization must not make any assumption on the data and be robust with regards to arbitrary ordering, absence, and duplication of events.

DR 5: Specific sequences are uniquely identifiable — Within the overview, the reader can still identify specific sequences and all the events belonging to that sequence.

DR 6: Support reasoning about an individual event type — The visualization should display the distribution of an attribute value, the frequency of the precursor, and aftereffect events.

4 SEQUENCE BRAIDING

SEQUENCE BRAIDING is a method to display multiple sequences of events with associated values, that aims to help the reader detect patterns and trends in the data. Multiple sequences spanning a time window are represented adjacent to each other, with each sequence being displayed as a line. Events which occurred throughout a sequence are represented by nodes placed on the line and aligned according to their type.

The ranks for each event type are computed before building the visualization via pairwise alignment. In accordance with *DR 4*, the use of an alignment algorithm allows our method to represent any types of event occurring in any order. In SEQUENCE BRAIDING, each event has a continuous attribute associated with it. The attribute values are grouped into ordinal groups based either on the data or to minimize edge crossings. We keep a consistent ordering of events according to their associated attribute across the ranks in order to make patterns and outliers emerge. The most apt ordering for the attribute should be used. This can be the numeric value of the attribute (e.g., blood glucose), or an arbitrary ordering (e.g., chess pieces).

SEQUENCE BRAIDING is designed for non-cyclic sequences: if an event occurs twice in a sequence, the event will be repeated. This is necessary in order to represent sequences with repeating events (such as multiple snacks in a day) that do not imply a cycle. Cyclicity can be managed by folding [13] the sequences according to the qualities of a dataset and needs of the designer. In our diabetes case study, given the cyclic nature of daily behaviors we fold the sequences at each midnight. However, folding by week may also be appropriate.

One of the largest advantages that SEQUENCE BRAIDING offers over the other currently available methods is the ability to represent many sequences simultaneously in relatively little space, as in fig. 7 with 200 sequences. Even with more sequences, the overview clearly shows the distribution of events in each attribute group for each event type, and by focusing on the groups it can be read like a bar chart.

We believe SEQUENCE BRAIDING can be generalized to many domains which have sequences with similar characteristics, discussed in section 8.3. Examples of other uses can be found in appendix A.

4.1 Description of the Algorithm

SEQUENCE BRAIDING represents events as nodes in a graph. An edge represents a sequential relationship between two events. The algorithm has two key steps: *rank assignment* & *intersection reduction*. Rank assignment is obtained through a pairwise alignment of the input sequences. Ordering within each rank is then done through a constrained intersection reduction algorithm based on the barycentric approach.

The objective of the rank assignment step is to find the shortest common supersequence of events that ensures they match across sequences. This supersequence allows us to keep events that are often subsequent close together, creating clusters of edges and high-level visual patterns. Likewise, anomalous infrequent events stand out from the rest. The intersection reduction step is then used to improve the readability and visibility of the high-level patterns. This step moves nodes within the prescribed constraints imposed by the rank assignment and their ordinal attribute value groups. The diabetes dataset well demonstrates how these features play well together. Meals that often happen subsequently appear close together and common chains of events and attributes appear grouped, making the trend stand out. Conversely, rarer events catch the eye as outliers not grouped with the others.

The output is a graph layout that supports many-event alignment [*DR 1*], allows the holistic analysis of a set of sequences [*DR 3*], yet ensures specific sequences are uniquely identifiable [*DR 5*].

4.1.1 Rank Assignment: Multiple Sequence Alignment

The shortest common supersequence is the sequence that contains all the sequences of a collection as subsequences. Given two sequences $X =< x_1, ..., x_m >$ and $Y =< y_1, ..., y_n >$, a sequence $U =< u_1, ..., u_k >$ is a common supersequence of X and Y if items can be removed from U to produce X or Y. In our case, we want to find out the shortest common supersequence of our collection of sequences of events. This is known either as the *multiple sequence alignment* or the *shortest supersequence* problem. Sequences can be aligned by adding gaps, assigning a positive score to the matches, or a negative score to all the mismatches or gaps that are introduced. As an example, the result of aligning the two DNA sequences `ATCGGTGAC` and `ACGGTATC` by inserting gaps (-) could be:

```
ATCGGTGAC → ATCGGTGA-C → ATCGGTGATC
ACGGTATC  → A-CGGT-ATC → ..........
```

Similarly, we map each event type to an alphabet letter and produce a general sequence of events that has all the sequences in our dataset as subsequences. There are several ways to perform multiple sequence alignment. A general approach is to use a graph to represent all possible alignments, and use a heuristic to help find a path through the graph that produces an alignment [23, 36]. However, due to the size that the graph can reach, this approach is only feasible for small numbers of sequences. Another one is progressive alignment, in which a multiple sequence alignment is built up gradually using a series of pairwise alignments [17, 47]. Our approach is based on progressive alignment, finding out the best matching sequences through pairwise alignment and progressively aligning all the others. We used the Pairwise DNA Alignment implementation from [45] to obtain the supersequence. Although the algorithm is borrowed from a biomedical domain, pairwise DNA alignment can be used in any domain. This allows us to compute the shortest minimum supersequence of sequences with arbitrary event types for any domain. In this way, if we align these sequences:

```
Breakfast → Lunch  → Snack → Dinner
Lunch     → Dinner → Snack
Breakfast → Snack  → Snack
```

We would obtain this supersequence:

```
Breakfast → Lunch  → Snack → Dinner → Snack
```

The final result will be determined by the weight that is assigned to each possible gap or match. The weights that we used are:

```
MATCH_SCORE    :  10   BEGIN_GAP_PENALTY : 2
MISMATCH_SCORE : -50   GAP_PENALTY       : 1
                       END_GAP_PENALTY   : 2
```

In this way, we are indicating that mismatches are unacceptable, the insertion of gaps is discouraged but accepted, and that we prefer more gaps in the inner part of the sequences instead of outer buffers.

The use of the shortest common supersequence of events allows us to minimize the number of ranks and the distance between common subsequent events across all the sequences. While generating the supersequence, we abstain from making any assumptions about the order of meals, or the number of them. Since we don't make assumptions about the ordering of the events, our method is robust towards anomalies in the data and the messiness of real-world data [*DR 4*].

4.1.2 Building the Graph

We use a layered acyclic directed graph to represent our problem because it allows for a straightforward representation of sequences, and allowed us to map the terminology to already-existing intersection reduction techniques ongraphs. We represent each event as a node, and each relationship of sequentiality between two events as an edge.

The events in a sequence succeed each other in time, and there is no way for an event to be succeeded by another event that is prior in time. Therefore, the graph must be acyclic, which also allows for a consistent rank assignment. We build the graph using the supersequence obtained by the previous step. The depth (rank) of each node is determined by:

```
ref_seq ← supersequence
ℓ ← length of supersequence
for s in sequences do
    s[0].depth = 0
    for i = 1 to s.length do
        cur_node = s[i]
        prev_node = s[i − 1]
        cur_ref = ref_seq.slice(prev_node.depth, ℓ)
        cur_node.depth = cur_ref
                .indexOf(k such that k == cur_node.event_type)
```

We cycle through every node in a sequence, and slice the supersequence based on the depth of the previous node, thus making sure that the graph is acyclic. We then find in the sliced supersequence the next event that corresponds to the event type of the current node.

Each edge that passes through a rank without having a node on it will be assigned an *anchor*: an object that behaves like a node in the sorting process but is not assigned to a group and does not have a graphical representation, and is only needed to assign a proper shape to the edge, in order to make it pass through without overlapping with any nodes.

All nodes belonging to the same group must be adjacent, without gaps or anchors between them. At this point, a rank contains a set of anchors and a set of groups, with each group containing one or more nodes. We then need order the elements in a rank on two levels: between groups and anchors, and between nodes within each group.

4.1.3 Within-rank Node Sorting: intersection reduction heuristic

The vertical ordering is done following a heuristic approach based on Gansner et al.'s work [19] for intersection reduction, that in turn evolves from Sugiyama's method [46]. Gansner et al. describe a general approach including the assignment of ranks in a graph, but our ranks are already assigned at this point, so we only use their technique within each rank to establish a vertical order.

The sorting has to respect the following constraints: (1) If node A belongs to a higher group than node B, node A should always be positioned above node B. (2) Nodes in the same group should be adjacent.

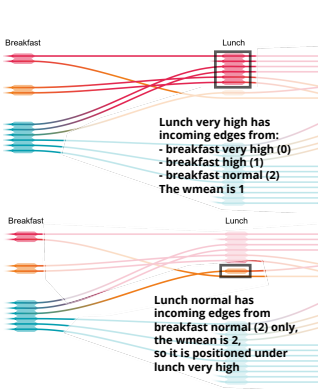

Fig. 2: The vertical positioning of the groups in each rank is computed according to the position, in the previous rank, of the nodes that are sources to the edges incoming in the group.

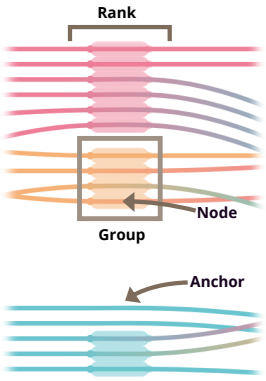

Fig. 3: A rank can contain groups and invisible anchors. A group can contain 1 or more nodes. The process of reordering nodes and groups is repeated first left to right, then right to left, several times.

Differently from the one by Gansner et al., our method produces a result that is less optimized for crossings, but is suited for the specific purpose of highlighting patterns. While their method starts with an initial assignment of the ranks, our ranks are given by the alignment algorithm. In the second step, when the ordering within the ranks is computed, we have additional constraints over a general layered graph that we need to respect: groups of nodes belonging to the same level must be bundled together, and they need to follow the specific order assigned to the general levels (e.g., in the case of the blood glucose dataset, 'very high' nodes must be above 'high' nodes). Beyond a general rank ordering, we also reorder the nodes within each group.

Initially, we define an *order* object: a two dimensional matrix having all the ranks on one dimension and the sorted nodes in each rank on the other dimension. After declaring a maximum number of iterations, we incrementally sort the nodes, alternating between starting from the leftmost rank and from the rightmost rank. The *crossings()* function counts how many intersections there are when a particular order is applied to the graph. We store the values in a temporary structure and find our best ordering by iteratively sweeping left and right:

```
cur_ord ← initial ordering
best_ord ← cur_ord
for i = 0 to max_iterations do
    if i%2 == 0 then
        tmp_ord = sort_nodes_left(cur_ord)
    else tmp_ord = sort_nodes_right(cur_ord)
    if crossings(tmp_ord) ≤ crossings(best_ord) then
        best_ord = tmp_ord
apply(best_ord)
```

The structure is swept at each iteration alternately from left to right and from right to left. Each sorting pass of each one of the ranks will sort items in each rank both at their group level (sorting groups and anchors), and at the level of the nodes inside each group.

```
function sort_nodes_left(ord)
    for i = 0 to ord.length do
        rank ← ord[i]
        for item in rank do
            if item.isgroup then
                for node in item.nodes do
                    node.wvalue ← wvalue(node)
                item.nodes.sort((a,b) →
                        within_group_sort(a,b))
            item.wmean ← wmean(item)
        rank.sort((a,b) → within_rank_sort(a,b))
```

The *wvalue* attribute is computed for all the nodes within a group, and equals to the position in the previous rank of the node preceding the

Fig. 4: 1 day represented using SEQUENCE BRAIDING will be displayed as single line. The day in the visualization had a breakfast in the normal blood glucose range, then lunch and snack in the very high blood glucose range, and then finished the day with a normal dinner.

current node in the same sequence. The *wmean* attribute is computed for all the groups, and it is just the mean of all the *wvalues* of the nodes belonging to the group. Note that while the *within_group_sort* function is just a simple sorting based on the *wvalue*, the *within_rank_sort* has a constraint that forces elements belonging to a higher group to be on top of elements belonging to a lower group.

1: $groups \leftarrow$ ordered list of groups
2: **function** $within\_group\_sort(a,b)$
  return a.wvalue $< b$.wvalue
3: **function** $within\_rank\_sort(a,b)$
4: **if** $groups.indexOf(a.group) < groups.indexOf(b.group)$
5: **return** 1
6: **else return** $a.wmean < b.wmean$

We consider convergence reached when we obtain the same solution with the same number of intersections from three different cycles. *max_iterations* is set to 20 by default, but can be customized. We report a table with performance time in appendix B.4. In all the tests we performed — including up to 3500 edges — the algorithm converged in 13 or less iterations.

4.1.4 Linear Programming for Exact Intersection Reduction

The use of a heuristic for intersection reduction means no guarantees of finding the optimal solution. An approach to find the optimal solution is to formulate a linear program using the same constraints. Similar approaches have already been experimented with [18, 62], and the pros and cons have been discussed by Gansner [19]. We run a series of experiments using a solution based on Zarate et al. [62]. Although a linear programming approach finds an optimal and aesthetically pleasing solution for smaller graphs, this approach does not scale to higher numbers of edges. Indeed, while SEQUENCE BRAIDING is able to effortlessly process a graph with 1400 edges in 2 seconds, our linear programming approach ran out of memory. As representing more sequences was more relevant for us than displaying an optimal solution to the intersection reduction problem, we deemed the use of a heuristic more appropriate. Our linear programming implementation details and timing comparisons can be found in appendix B.

4.2 Implementation

We built a free and open-source web-based visualization tool using JavaScript and the D3 library [6]. *SequenceBraiding.js* uses the output of the graph layout algorithm to produce a visual representation. We use rounded rectangles to display the nodes, and splines for the edges between a node and the next. Each spline has anchor points before and after the traversed nodes that allow us to give the impression of seeing bundles of lines. Multiple gradients are applied to each spline reflecting the colors of the nodes it passes through. Many aspects are customizable. For example, the order of the groups can be forced: although the algorithm can guess it, it is at times necessary to impose an order, e.g., when the it is meaningful to the domain. Event types — or columns — can be filtered out by removing the ones that have fewer events than a threshold. This reduces the event types displayed, but can be useful to filter outliers in datasets with a very high number of events. Additional details can be added by combining complementary views [9]. In fig. 6, we show an example of how we combined multiple complementary views from a plugin that we built, that enables showing additional information and charts for each one of the sequences, and enables sorting by attribute values. Documentation, customizability and interactive usage examples are described on the SEQUENCE BRAIDING `website`[2].

[2] `visdunneright.github.io/sequence_braiding/docs/`

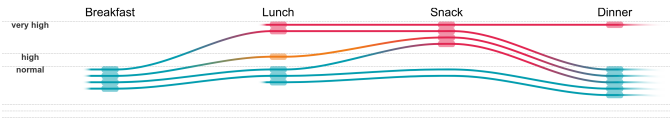

Fig. 5: 6 days represented using SEQUENCE BRAIDING. The lines bundle together when the sequences follow a common pattern.

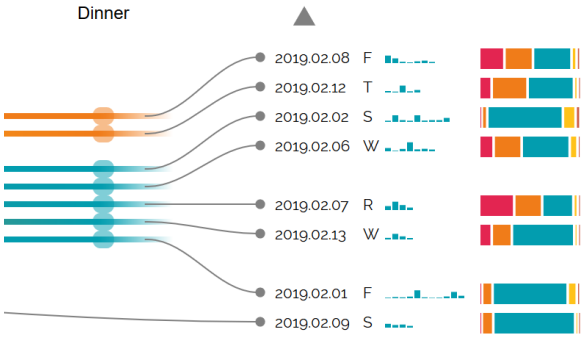

Fig. 6: A plugin for SEQUENCE BRAIDING showing additional details about the sequences. Here, we used the type 1 diabetes dataset: the additional details for each sequences are the full date, day of the week, distribution of the intake of carbohydrates for each meal event, and percentage of time that day spent in pre-set blood glucose ranges.

5 COMPARATIVE EVALUATION

We conducted a mixed methods comparative evaluation of SEQUENCE BRAIDING vs. a baseline visualization for understanding trends and patterns in event sequence data. In a task-based, within-subjects, controlled experiment we evaluated completion time, correctness, and error magnitude. Before starting, the study design and analysis plan were posted at `osf.io/mq2wt`. See the analysis plan there for complete details.

5.1 Baseline for Comparison

We use IDMVis [63] as the baseline for comparison. IDMVis was designed to support type 1 diabetes treatment, using multiple views to support a large portion of the tasks that would take place during a clinical visit. Its main view uses several alignment techniques to enable comparing blood glucose levels at meals across several days, even if the meals did not happen at the same time each day. We only compare SEQUENCE BRAIDING with the main view from IDMVis as our focus is on showing high-level trends in an overview visualization. In fact, SEQUENCE BRAIDING could be a suitable drop-in replacement for the IDMVis main view. We limited our consideration of possible baselines for comparison to tools that: (1) support event sequence visualization, (2) support event alignment or aggregation by attributes, and (3) are open source or have otherwise available code for customization and evaluation. Candidates we considered include Storylines [38], Storyflow [30], and Sequence Bundles [27]. However, we found each unsuitable due to substantially different purposes, constraints, or approaches. More details are available in appendix C.

5.2 Tasks

The controlled experiment included 7 tasks, shown in the first column of fig. 9 (A). We designed these tasks based on Adrienko & Adrienko's systematic approach for visualizing temporal data [3] and the task analysis and abstraction for type 1 diabetes treatment developed by Zhang et al. for IDMVis [63], our baseline for comparison. We considered using well-cited task frameworks such as Brehmer et al.'s multi-level typology [7] and Lee et al.'s taxonomy for graph visualization [29]. However, we found these frameworks to be too generic for helping us design our tasks.

Here we define the terminology used for our tasks. We contrast *elementary tasks* and *synoptic tasks* as per Adrienko & Adrienko [3]. Elementary tasks relate to individual elements of the *reference set*, whereas synoptic tasks involve the whole reference set or its subsets [3].

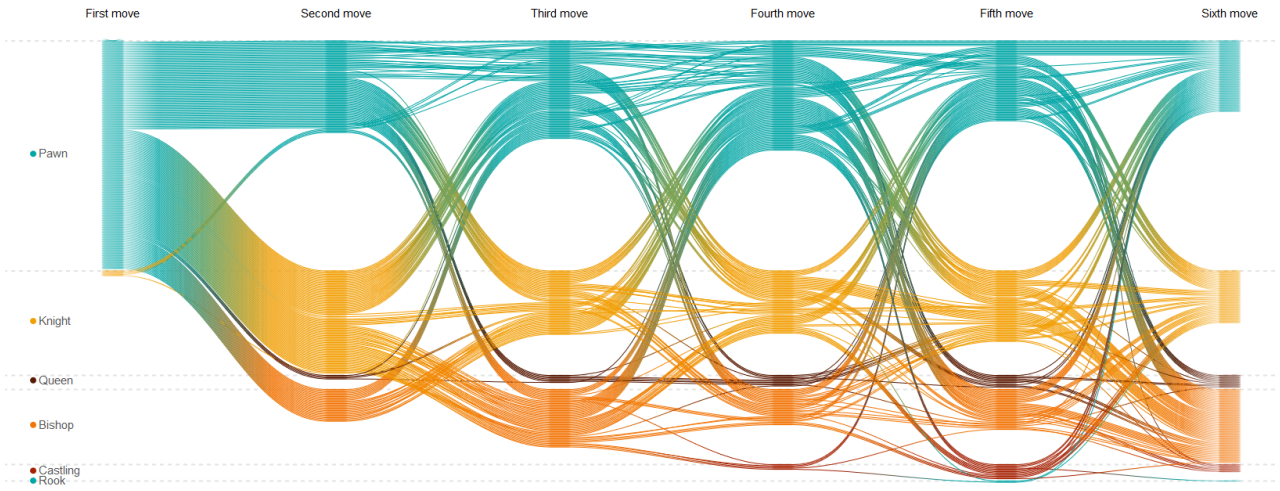

Fig. 7: 200 chess openings displayed with SEQUENCE BRAIDING. Each line represents a sequence of moves of the white player, each group is a chess piece type. Most openings start with a pawn, and very little with the knight. After moving a pawn, it is common to move a knight or a pawn, it is a little less common to move a bishop, and only a little number of openings move the queen on the second move.

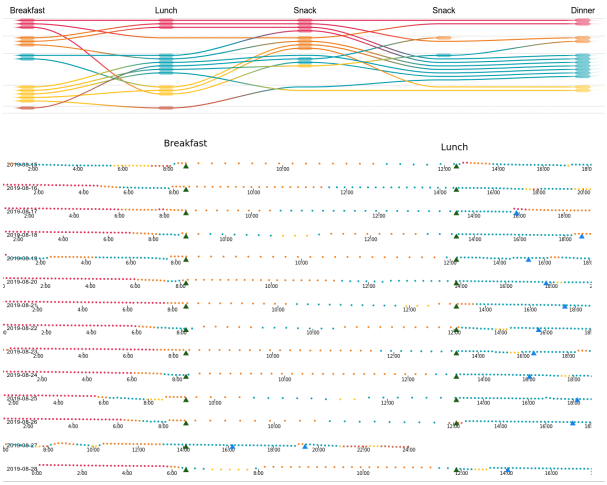

Fig. 8: Comparison of the stimuli used in the experiment (Top: SEQUENCE BRAIDING vs. Bottom: IDMVis).

In SEQUENCE BRAIDING, an element of the reference set can be a single day represented by a line to indicate the time interval or a given event (e.g., meal) visualized by a rounded rectangle. A *lookup* elementary task means finding a characteristic relative to a specified element. An *inverse* task involves having the participant find an element or relevant subset for a given characteristic. For tasks including the term *behavior*, we mean a generalisation of notions such as distribution, variation, and trends — and *behavior comparison* indicates measuring the degree of similarity/difference between behaviors for multiple given reference sets. *Relation seeking* indicates finding occurrences of specified relations between characteristics and determining the corresponding elements or reference sets, while *pattern identification* is the synoptic equivalent of an elementary lookup task.

We further refined the task selection based on the task analysis and abstraction examined by Zhang et al. for IDMVis [63] to support a more realistic comparison and our case study on type 1 diabetes treatment. We include tasks that support overviews; focusing on identifying and comparing trends, patterns, and outliers; as well as investigating data quality such as missing data. We only considered tasks which could be performed using both SEQUENCE BRAIDING and the main view in IDMVis, e.g., SEQUENCE BRAIDING does not display precise timing and IDMVis does not support aligning more than two meal labels at the same time, eliminating tasks that require these encodings.

5.3 Stimuli and Questions

We formulated 7 multiple-choice questions instantiating our selected tasks for the controlled experiment. In our within-subjects study, each participant answered each of these questions using both visualizations generated by SEQUENCE BRAIDING and IDMVis. The order of the questions was counterbalanced to avoid ordering effects using a Latin square design. We used the CGM and meal log event sequence data from IDMVis [63] and its associated study [64] to build our stimuli. Each question uses a different set of 14 days sampled from the dataset. The data contained anomalies due to human error. SEQUENCE BRAIDING is robust towards anomalies, but we wanted to make sure to avoid confusing our participants so we eliminated days with problematic ambiguities. E.g., breakfasts recorded at 9 PM or lunches recorded after dinners. As a result, the selected days for a question are not necessarily consecutive but are from the same rough time period. We also smoothed out places in the CGM curve where rapid changes co-occurred with events relevant to our questions so as to avoid ambiguity.

Mark and font sizes were kept as similar as possible for the two visualizations. The dots in IDMVis and the nodes in SEQUENCE BRAIDING had similar heights. In IDMVis colored dots are used to show time series data and triangles show events, while SEQUENCE BRAIDING uses colored nodes for less frequent events and their attribute values. SEQUENCE BRAIDING also uses lines to show event sequences while IDMVis uses rows. Meal name fonts were also similar sizes. All the stimuli can be found in our supplemental material at `osf.io/mq2wt`.

5.4 Procedure

We used a within-subjects approach. Authorized by the Institutional Review Board of our institution, we recruited participants using flyers at our institution and posting ads on Facebook. We ran a pilot study before recruitment that helped us evaluate duration, feasibility, and adverse events; verify our assumptions; and find flaws in the design. Based on the pilot study, we conducted power analysis (with Type I error rate $\alpha = 0.05$, power $1 - \beta = 0.80$) to determine the number of total participants. We determined we needed at least 25 participants to achieve consistent results for tasks 1, 4, 6, and 7. 16 males, 8 females, and one that preferred not to say participated in our study. The median age was 24 years old ($IQR = 6$), 5 participants held a Doctorate, 7 a Master's degree, 9 a Bachelor's degree, and 4 a high school degree or equivalent. In this in-person study, participants first signed a consent form for recording audio and video, confirming they were at least 18 years of age and comfortable being interviewed in English, and acknowledging the IRB terms. Participants were then given a laptop — a Dell XPS 9570 with a 15-inch non-reflective screen running Firefox 68 on Ubuntu 18.04 —, mouse, keyboard, blank paper, pens, and a calculator to perform simple operations. The calculator was provided so as to minimize the effect of participant numeracy, e.g., for calculating

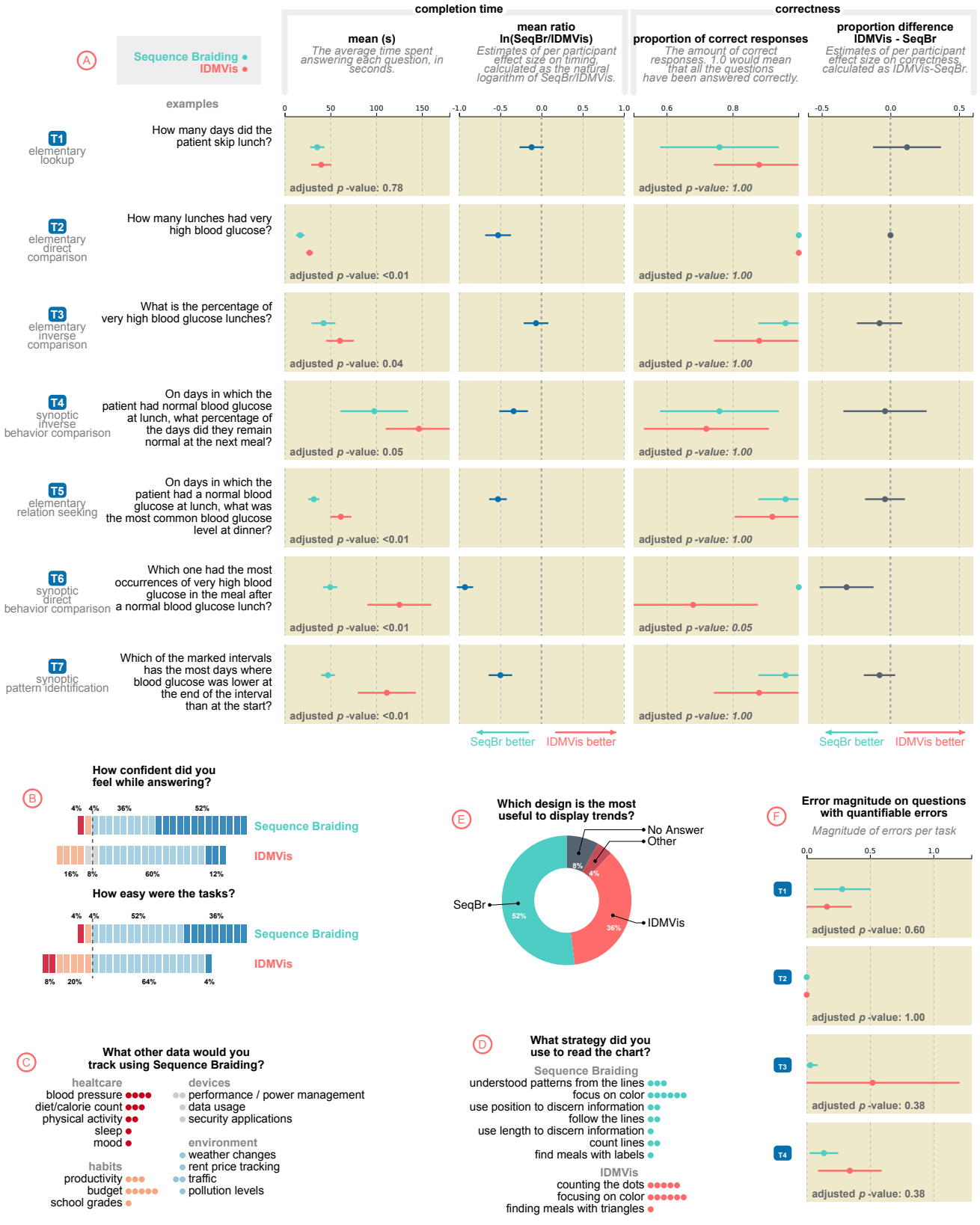

Fig. 9: Results of our evaluation comparing SEQUENCE BRAIDING vs. IDMVis [63]. (A) Completion time and correctness per task. Each row corresponds to the task at left, which is classified based on Andrienko & Andrienko [3]. The specific question instantiating that task for the study is in the second column. (B) Participants' Likert scale responses regarding confidence and ease of use. (C) Participants' answers when asked what other types of data would they use with SEQUENCE BRAIDING. (D) Participants' reported strategies used. (E) Participants' preference for which method was most useful for displaying trends. (F) Error magnitude per task, for those which are quantifiable.

percentages, as numeracy was a possible confounding variable.

In the browser, participants then viewed an online tutorial about how to read the first visualization they would see, followed by answering 7 multiple-choice questions using the associated stimuli. After completing these, participants viewed a tutorial about the second visualization, followed by the other set of 7 associated questions. We alternated between participants starting from IDMVis and starting from SEQUENCE BRAIDING. Our web app registered their answers and completion time for each question answered, showed the correct answer after they confirmed their answer, and stored the data for later analysis. At the end, they filled in a survey with demographic and feedback questions. Each participant was paid with a $10 Amazon gift card.

6 ANALYSIS

6.1 From NHST to Interval Estimation

Null hypothesis significance testing (NHST) tests a null hypothesis and uses the p-value to decide whether to reject that hypothesis. Most descriptions of statistical testing only focus on NHST for claiming a definitive statistical significance [11]. However, p-values are not definitive proofs of significance and it is dangerous to use a predefined threshold like $p < .05$ to separate real results from the false ones. A more refined goal of statistical analysis is to provide an evaluation of certainty or uncertainty regarding the size of an effect [20]. Interval Estimation involves providing information about a range in which the true value lies with a certain degree of probability. Confidence intervals are examples of interval estimates. Reporting both confidence intervals and p-values helps better understand what's happening in the data than p-values alone because they provide complementary information [14].

6.2 Measures and Analysis

For each task, we measured the completion time and accuracy. The accuracy is quantified at two levels of granularity for each question. First, we determined the correctness, a binary measure of whether or not the participant got the correct answer. Second, for the tasks which had numerical answers, we report the amount of error (ER) using $ER = \mid O_p - O_t \mid / O_t$, where the O_p and O_t represent the option that the participant selected and answered correctly, respectively. In this tasks study, the ER was only applicable to the first four.

To analyze our experiment data, we used both NHST and interval estimation. We formed the null hypothesis as follows based on our objective with this evaluation: There is no meaningful difference between SEQUENCE BRAIDING and IDMVis in correctness, error rate, and completion time for tasks involving lookup, comparison, relation-seeking, pattern comparison, and pattern identification of quantitatively abstractions when investigating time-oriented variables.

We first tested for normality with two methods qualitatively and quantitatively by using Q-Q plots and the Shapiro-Wilk test respectively. Our results showed that our data does not have a normal distribution. Therefore, we used chi-square test for independence to analyze correctness and the Wilcoxon signed-rank test as the nonparametric alternative to the dependent t-test to analyze time and error rate. We adjusted all p-values using the Benjamini-Hochberg procedure [5] to control the false discovery rate. We report mean completion time and use the natural logarithm of the ratio of means as effect size — natural log ratios are the only symmetric, additive, and normed indicators of relative change [51]. We also report the mean accuracy and use the difference between the samples means as effect size. Last, we report mean error magnitude. For each of these we report bootstrapped 95% confidence intervals (CIs) to indicate the range of plausible values [16].

7 QUANTITATIVE RESULTS

Chart Ⓐ in fig. 9 shows our results in terms of time and correctness. In general, *there is good evidence that participants are faster using* SEQUENCE BRAIDING *than IDMVis*. The fourth column of Ⓐ shows the proportion of participants who responded correctly to each task. We computed binary proportions and their CIs. A ceiling effect appeared in task 2, in which all participants responded correctly with both visualizations. There was another ceiling effect in task 6 using our method. Chart Ⓕ shows the error magnitude for the tasks with quantifiable error (Tasks 1-4). The ceiling effect reappeared here. Results about correctness were less conclusive, varying across methods and remaining tasks.

Task 1: elementary lookup

time	avg	95% CI	p-value	avg ln(S/I)	95%CI ln(S/I)
SB	35.4	[27.8, 43.3]	0.78	-0.12	[-0.26, 0.03]
IDMVis	39.6	[28.8, 50.8]			
accuracy				**avg ln(I-S)**	**95%CI ln(I-S)**
SB	0.76	[0.58, 0.93]	1.00	0.12	[-0.12, 0.37]
IDMVis	0.88	[0.74, 1.00]			
error rate					
SB	0.28	[0.06, 0.50]	0.60		
IDMVis	0.16	[0.00, 0.35]			

Completion time: We have minimal evidence that SEQUENCE BRAIDING mean completion time is slightly shorter than IDMVis. **Accuracy:** The proportion of correct responses and error rate are not meaningfully different between interfaces.

Task 2: elementary direct comparison

time	avg	95% CI	p-value	avg ln(S/I)	95%CI ln(S/I)
SB	16.8	[12.1, 21.5]	< 0.01	-0.52	[-0.68, -0.37]
IDMVis	27.1	[23.3, 30.8]			
accuracy				**avg ln(I-S)**	**95%CI ln(I-S)**
SB	1.00	[1.00, 1.00]	1.00	0.00	[0.00, 0.00]
IDMVis	1.00	[1.00, 1.00]			
error rate					
SB	0.00	[0.00, 0.00]	1.00		
IDMVis	0.00	[0.00, 0.00]			

Completion time: We have strong evidence that participants are meaningfully faster using SEQUENCE BRAIDING. **Accuracy**: All participants responded correctly using both interfaces.

Task 3: elementary inverse comparison

time	avg	95% CI	p-value	avg ln(S/I)	95%CI ln(S/I)
SB	42.3	[29.3, 55.4]	0.04	-0.07	[-0.21, 0.08]
IDMVis	60.2	[45.0, 74.5]			
accuracy				**avg ln(I-S)**	**95%CI ln(I-S)**
SB	0.96	[0.88, 1.00]	1.00	-0.08	[-0.24, 0.08]
IDMVis	0.88	[0.74, 1.00]			
error rate					
SB	0.03	[-0.03, 0.09]	0.38		
IDMVis	0.52	[-0.16, 1.20]			

Completion time: We have weak evidence that participants are faster using SEQUENCE BRAIDING. **Accuracy**: No meaningful difference in accuracy between interfaces, though the error magnitude with IDMVis had a much larger variance.

Task 4: synopic inverse behavior comparison

time	avg	95% CI	p-value	avg ln(S/I)	95%CI ln(S/I)
SB	97.8	[60.7, 134.9]	0.05	-0.33	[-0.51, -0.16]
IDMVis	146.6	[110.2, 183.0]			
accuracy				**avg ln(I-S)**	**95%CI ln(I-S)**
SB	0.76	[0.58, 0.94]	1.00	-0.04	[-0.34, 0.26]
IDMVis	0.72	[0.53, 0.90]			
error rate					
SB	0.14	[0.02, 0.24]	0.38		
IDMVis	0.34	[0.09, 0.59]			

Completion time: We have good evidence that participants are meaningfully faster using SEQUENCE BRAIDING. **Accuracy:** Participants had similar correctness and error rates using SEQUENCE BRAIDING and IDMVis.

Task 5: elementary relation seeking

time	avg	95% CI	p-value	avg ln(S/I)	95%CI ln(S/I)
SB	31.8	[25.7, 37.9]	< 0.01	-0.53	[-0.63, -0.42]
IDMVis	61.3	[49.8, 72.7]			
accuracy				**avg ln(I-S)**	**95%CI ln(I-S)**
SB	0.96	[0.88, 1.00]	1.00	-0.04	[-0.18, 0.10]
IDMVis	0.92	[0.80, 1.00]			

Completion time: We have strong evidence that participants were meaningfully faster with SEQUENCE BRAIDING. **Accuracy**: Participants had similar correctness in both interfaces.

Task 6: synopic direct behavior comparison

time	avg	95% CI	*p*-value	avg ln(S/I)	95%CI ln(S/I)
SB	49.6	[41.8, 57.4]	< 0.01	-0.93	[-1.02, -0.83]
IDMVis	125.2	[90.3, 160.2]			
accuracy				**avg ln(I-S)**	**95%CI ln(I-S)**
SB	1.00	[1.00, 1.00]	0.05	-0.32	[-0.50, -0.12]
IDMVis	0.68	[0.48, 0.87]			

Completion time: Participants spent up to 2.5 times longer with IDMVis than SEQUENCE BRAIDING and we have strong evidence to support this difference. This is the largest difference in task completion time that we have observed. **Accuracy:** We have good evidence that participants were meanginfully more correct using SEQUENCE BRAIDING than IDMVis.

Task 7: synopic pattern identification

time	avg	95% CI	*p*-value	avg ln(S/I)	95%CI ln(S/I)
SB	47.2	[39.7, 54.8]	< 0.01	-0.50	[-0.64, -0.36]
IDMVis	111.5	[79.8, 143.2]			
accuracy				**avg ln(I-S)**	**95%CI ln(I-S)**
SB	0.96	[0.88, 1.00]	1.00	-0.08	[-0.19, 0.03]
IDMVis	0.88	[0.74, 1.00]			

Completion time: We have strong evidence that participants were meaningfully faster with SEQUENCE BRAIDING. **Accuracy**: Participants had similar correctness with both interfaces.

8 DISCUSSION

8.1 Task Type Matters

Overall we found that *there is good evidence that participants are faster using* SEQUENCE BRAIDING *than IDMVis*. This time improvement was primarily on tasks which involved analyzing the sequentiality of events (T4, T5, T6, T7). The most outstanding time improvement, as well as the only big correctness difference, is found in T6 which involved comparing two time intervals and finding a pattern. Two 14-day IDMVis visualizations incur substantial visual complexity, as the user is effectively looking at 28 small multiples visualizations in two groups. Conversely, SEQUENCE BRAIDING is capable of a much more compact and less complex representation still very fit for the task. Another large difference occurs in T2, a task for which SEQUENCE BRAIDING is effectively a stacked bar chart. It is thus not surprising that position on an unaligned scale is a more effective encoding than color luminance for this quantitative task [33].

8.2 Qualitative Explanations for Quantitative Results

We also collected participants' subjective evaluation of the two visualizations. A high-level summary is shown in fig. 9 Ⓑ, Ⓓ, Ⓔ. Validating a novel algorithm and visualization tool using clock-wall evaluations (e.g., time and accuracy) [33] alone can provide insights on the tool performance. However, our qualitative findings may provide explanations of *why* SEQUENCE BRAIDING outperformed IDMVis.

As one approach to examine why SEQUENCE BRAIDING performs better, we conducted affinity diagramming [22] to analyze the qualitative results. It highlighted two benefits of using SEQUENCE BRAIDING: it (1) helps identify overall patterns, and (2) can be more suitable for more complex datasets with longer duration.

We also looked at specific ratings provided by participants. Participants reported feeling more confident performing tasks using SEQUENCE BRAIDING vs. IDMVis (fig. 9 Ⓑ) — mean 0.63 points higher on a 1–5 scale. Likewise, participants preferred SEQUENCE BRAIDING vs. IDMVis regarding ease of use for understanding trends and patterns Ⓑ — mean 0.79 points higher on a 1–5 scale. Furthermore, 53% of the participants reported SEQUENCE BRAIDING as most useful for displaying trends, vs. 36% for IDMVis Ⓔ.

We also examined the strategies participants claimed to use when reading the visualizations (fig. 9 Ⓒ). In both cases focusing on color was most common. With SEQUENCE BRAIDING, participants also used many other strategies in concert. But only one other strategy was given for IDMVis — counting dots. This suggests that SEQUENCE BRAIDING users would pick and choose from the encodings most relevant or comfortable for their task. Moreover, the strategies reported for SEQUENCE BRAIDING are perhaps at more focused on higher-level patterns vs. inferring patterns by reading small multiples visualizations in IDMVis. It stands to reason that the difference in strategies used influenced the higher confidence scores Ⓑ, perceived ease Ⓑ, and perceived usefulness Ⓔ of SEQUENCE BRAIDING vs. IDMVis — as well as the timing and overall smaller error rate.

8.3 When to Use Sequence Braiding

SEQUENCE BRAIDING was originally developed to easily represent several weeks or months of meal logs for diabetic patients, with the event sequence folded [13] by day. We believe our method is most effective on data with similar characteristics. In particular, our alignment approach will be most helpful in cases where there is no apriori canonical ordering of event types. Our algorithm is robust to human-generated data which is prone to containing inconsistencies or situations with a mixed ordering or acyclically repeating event types. Likewise, the visual design of SEQUENCE BRAIDING is most directly applicable to domains in which attribute value and sequence overviews are important to see, where precursor and aftereffect events must be known, and where it is valuable to show specific sequences individually. However, it would not be useful in cases where specific times of events or time deltas between them are important — at least without added interactivity or additional encodings. There are also algorithmic and visual scalability considerations. We have not calculated exact numbers for how many sequences, event types, or events in a sequence will work in general, but we discuss the overall constraints and refer readers to more examples in appendix B.4. The worst-case runtime complexity of our layout is $O(k \cdot m \cdot n^2 log(n))$, with k iterations, m ranks from alignment, and n sequences. Experimentally, we can lay out 125 sequences in less than a second using JavaScript on a commodity laptop. See appendix B.4 for details. The visualization scales well with more sequences, but worse with longer sequences and many event types. Additional ranks from the alignment require much more screen space than additional lines and associated nodes. If sequences are not necessary to perceive individually, visual scalability can be improved by reducing the gap between lines and their thickness (configuration options). Moreover, the visualization can use scrolling, zooming, or mouseover highlighting to explore relevant subsets or sequences. If appropriate, filtering out infrequent sequences of events also reduces the visual complexity substantially.

9 LIMITATIONS AND FUTURE WORK

Although we claim that SEQUENCE BRAIDING is useful for understanding trends, it does not allow for examining precise details, such as the timing of the events or exact attribute values. In addition, since we use heuristics to position nodes, the solution is not necessarily minimal in terms of intersections. The version we used in the experiment did not include any interactivity in order to reduce the number of variables that could influence the results. Currently, events with missing data (without associated values) are discarded. Initially we kept them in the visualization and displayed them as belonging to an 'unknown' type positioned using only the minimum intersection heuristic. We removed the feature due to the additional complexity added, but adding it back would increase robustness with regards to missing data.

10 CONCLUSION

We presented a novel method for temporal event sequence visualization that focuses on highlighting common patterns through the collection of sequences. We conducted an in-lab experiment to test the effectiveness of SEQUENCE BRAIDING in relation to another state-of-the-art technique, IDMVis [63]. The results suggest that our participants performed faster using SEQUENCE BRAIDING and indicated that SEQUENCE BRAIDING were easier to use for examining specific tasks (e.g., with no apriori canonical ordering of event types) and helped them obtain a holistic overview of the data.

ACKNOWLEDGMENTS

We thank the National Science Foundation for support under CRII award no. 1755901 and our reviewers & colleagues for their feedback.

REFERENCES

[1] S. Agarwal and F. Beck. Set Streams: Visual exploration of dynamic overlapping sets. *Computer Graphics Forum*, 39(3):383–391, 2020. doi: 10.1111/cgf.13988

[2] American Diabetes Association. 6. glycemic targets: Standards of medical care in diabetes—2018. *Diabetes Care*, 41(Supplement 1):S55–S64, 2018. doi: 10.2337/DC18-S006

[3] N. Andrienko and G. Andrienko. *Exploratory analysis of spatial and temporal data: a systematic approach*. Springer Science & Business Media, 2006. doi: 10.1007/3-540-31190-4

[4] M. A. Bekos, M. Kaufmann, K. Potika, and A. Symvonis. *Line Crossing Minimization on Metro Maps*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[5] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300, 1995. doi: 10.1111/j.2517-6161.1995.tb02031.x

[6] M. Bostock, V. Ogievetsky, and J. Heer. D^3: Data-driven documents. *IEEE transactions on visualization and computer graphics*, 17(12):2301–2309, 2011. doi: 10.1109/TVCG.2011.185

[7] M. Brehmer and T. Munzner. A multi-level typology of abstract visualization tasks. *IEEE transactions on visualization and computer graphics*, 19(12):2376–2385, 2013. doi: 10.1109/TVCG.2013.124

[8] B. Buckingham, J. Block, and D. M. Wilson. Continuous glucose monitoring. *Current Opinion in Endocrinology, Diabetes and Obesity*, 12(4):273–279, 2005. doi: 10.1097/01.med.0000168531.02813.e5

[9] C. Chang, B. Bach, T. Dwyer, and K. Marriott. Evaluating perceptually complementary views for network exploration tasks. In *Proc. 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, p. 1397–1407, 2017. doi: 10.1145/3025453.3026024

[10] W. Chen, F. Guo, and F.-Y. Wang. A survey of traffic data visualization. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):2970–2984, 2015. doi: 10.1109/TITS.2015.2436897

[11] G. Cumming. *Understanding the new statistics: Effect sizes, confidence intervals, and meta-analysis*. Routledge, 2013.

[12] T. N. Dang, N. Pendar, and A. G. Forbes. TimeArcs: Visualizing fluctuations in dynamic networks. *Computer Graphics Forum*, 35(3):61–69, 2016. doi: 10.1111/cgf.12882

[13] F. Du, B. Shneiderman, C. Plaisant, S. Malik, and A. Perer. Coping with volume and variety in temporal event sequences: Strategies for sharpening analytic focus. *IEEE Transactions on Visualization and Computer Graphics*, 23(6):1636–1649, 2017. doi: 10.1109/TVCG.2016.2539960

[14] J.-B. du Prel, G. Hommel, B. Röhrig, and M. Blettner. Confidence interval or p-value?: part 4 of a series on evaluation of scientific publications. *Deutsches Ärzteblatt International*, 106(19):335, 2009. doi: 10.3238/arztebl.2009.0335

[15] P. Eades and N. C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, 1994. doi: 10.1007/BF01187020

[16] B. Efron and R. Tibshirani. Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical science*, pp. 54–75, 1986. doi: 10.1214/ss/1177013815

[17] D.-F. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25(4):351–360, Aug. 1987. doi: 10.1007/BF02603120

[18] G. Gange, P. J. Stuckey, and K. Marriott. Optimal k-level planarization and crossing minimization. In U. Brandes and S. Cornelsen, eds., *Graph Drawing*, pp. 238–249, 2011. doi: 10.1007/978-3-642-18469-7_22

[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. doi: 10.1109/32.221135

[20] S. Greenland, S. J. Senn, K. J. Rothman, J. B. Carlin, C. Poole, S. N. Goodman, and D. G. Altman. Statistical tests, P values, confidence intervals, and power: a guide to misinterpretations. *European journal of epidemiology*, 31(4):337–350, 2016. doi: 10.1007/s10654-016-0149-3

[21] T. Gschwandtner, W. Aigner, K. Kaiser, S. Miksch, and A. Seyfang. CareCruiser: exploring and visualizing plans, events, and effects interactively. *Proc. IEEE Pacific Visualization Symposium*, pp. 43–50, 2011. doi: 10.1109/PACIFICVIS.2011.5742371

[22] B. Hanington and B. Martin. *Universal methods of design: 100 ways to research complex problems, develop innovative ideas, and design effective solutions*. Rockport Publishers, 2012.

[23] T. Ikeda and H. Imai. Fast A* algorithms for multiple sequence alignment. *Genome Informatics*, 5:90–99, 1994. doi: 10.11234/gi1990.5.90

[24] W. Javed and N. Elmqvist. Exploring the design space of composite visualization. In *Proc. IEEE Pacific Visualization Symposium*, pp. 1–8, 2012. doi: 10.1109/PACIFICVIS.2012.6183556

[25] M. Jünger, E. K. Lee, P. Mutzel, and T. Odenthal. A polyhedral approach to the multi-layer crossing minimization problem. In G. DiBattista, ed., *Graph Drawing*, pp. 13–24, 1997. doi: 10.1007/3-540-63938-1_46

[26] M. Krzywinski, I. Birol, S. J. Jones, and M. A. Marra. Hive plots—rational approach to visualizing networks. *Briefings in Bioinformatics*, 13(5):627–644, 12 2011. doi: 10.1093/bib/bbr069

[27] M. Kultys, L. Nicholas, R. Schwarz, N. Goldman, and J. King. Sequence Bundles: a novel method for visualising, discovering and exploring sequence motifs. In *BMC proceedings*, vol. 8, p. S8. BioMed Central, 2014. doi: 10.1186/1753-6561-8-S2-S8

[28] B. C. Kwon, J. Verma, and A. Perer. Peekquence: Visual analytics for event sequence data. In *ACM SIGKDD 2016 Workshop on Interactive Data Exploration and Analytics*, vol. 1, 2016.

[29] B. Lee, C. Plaisant, C. S. Parr, J.-D. Fekete, and N. Henry. Task taxonomy for graph visualization. In *Proc. 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*, pp. 1–5, 2006. doi: 10.1145/1168149.1168168

[30] S. Liu, Y. Wu, E. Wei, M. Liu, and Y. Liu. StoryFlow: Tracking the evolution of stories. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2436–2445, Dec 2013. doi: 10.1109/TVCG.2013.196

[31] S. Liu, J. Yin, X. Wang, W. Cui, K. Cao, and J. Pei. Online visual analytics of text streams. *IEEE transactions on visualization and computer graphics*, 22(11):2451–2466, 2015. doi: 10.1109/TVCG.2015.2509990

[32] M. Monroe, R. Lan, H. Lee, C. Plaisant, and B. Shneiderman. Temporal event sequence simplification. *IEEE Transactions on Visualization and Computer Graphics*, 19:2227–2236, 2013. doi: 10.1109/TVCG.2013.200

[33] T. Munzner. Visualization analysis and design, 2014.

[34] P. Mutzel. An alternative method to crossing minimization on hierarchical graphs. In S. North, ed., *Graph Drawing*, pp. 318–333, 1997. doi: 10.1007/3-540-62495-3_57

[35] P. Mutzel and R. Weiskircher. Two-layer planarization in graph drawing. In K.-Y. Chwa and O. H. Ibarra, eds., *Algorithms and Computation*, pp. 72–79, 1998. doi: 10.1007/3-540-49381-6_9

[36] G. Nicosia and G. Oriolo. An approximate A* algorithm and its application to the scs problem. *Theoretical Computer Science*, 290(3):2021 – 2029, 2003. doi: 10.1016/S0304-3975(02)00085-3

[37] M. Nöllenburg. An improved algorithm for the metro-line crossing minimization problem. In D. Eppstein and E. R. Gansner, eds., *Graph Drawing*, pp. 381–392, 2010. doi: 10.1007/978-3-642-11805-0_36

[38] M. Ogawa and K.-L. Ma. Software evolution Storylines. In *Proc. 5th International Symposium on Software Visualization*, SOFTVIS '10, p. 35–42, 2010. doi: 10.1145/1879211.1f879219

[39] A. Perer and D. Gotz. Data-driven exploration of care plans for patients. In *Proc. Extended Abstracts on Human Factors in Computing Systems*, pp. 439–444, 2013. doi: 10.1145/2468356.2468434

[40] A. Perer, F. Wang, and J. Hu. Mining and exploring care pathways from electronic medical records with visual analytics. *Journal of biomedical informatics*, 56:369–378, 2015. doi: 10.1016/j.jbi.2015.06.020

[41] P. Riehmann, M. Hanfler, and B. Froehlich. Interactive Sankey diagrams. *IEEE Symposium on Information Visualization*, pp. 233–240, 2005. doi: 10.1109/INFVIS.2005.1532152

[42] A. Rind, T. D. Wang, W. Aigner, S. Miksch, K. Wongsuphasawat, C. Plaisant, and B. Shneiderman. *Interactive Information Visualization to Explore and Query Electronic Health Records*. Now Foundations and Trends, 2013. doi: 10.1561/1100000039

[43] G. Scheiner. *Think Like a Pancreas: A Practical Guide to Managing Diabetes with Insulin–Completely Revised and Updated*. Da Capo Press, 2012.

[44] O. Schnell, K. Barnard, R. Bergenstal, E. Bosi, S. Garg, B. Guerci, T. Haak, I. B. Hirsch, L. Ji, S. R. Joshi, et al. Clinical utility of SMBG: recommendations on the use and reporting of SMBG in clinical research. *Diabetes Care*, 38(9):1627–1633, 2015. doi: 10.2337/DC14-2919

[45] P. Stothard. The sequence manipulation suite: Javascript programs for analyzing and formatting protein and dna sequences. *BioTechniques*, 28(6):1102–1104, 2000. PMID: 10868275. doi: 10.2144/00286ir01

[46] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981. doi: 10.1109/TSMC.1981.4308636

[47] S.-H. Sze, Y. Lu, and Q. Yang. A polynomial time solvable formulation of multiple sequence alignment. In *Proc. 9th Annual International Confer-*

ence on Research in Computational Molecular Biology, RECOMB'05, p. 204–216, 2005. doi: 10.1007/11415770_16

[48] R. Tamassia. *Handbook of Graph Drawing and Visualization*. Chapman & Hall/CRC, 1st ed., 2016.

[49] Y. Tanahashi, C.-H. Hsueh, and K.-L. Ma. An efficient framework for generating Storyline visualizations from streaming data. *IEEE transactions on visualization and computer graphics*, 21(6):730–742, 2015. doi: 10.1109/TVCG.2015.2392771

[50] T. Tang, S. Rubab, J. Lai, W. Cui, L. Yu, and Y. Wu. iStoryline: Effective convergence to hand-drawn Storylines. *IEEE transactions on visualization and computer graphics*, 25(1):769–778, 2018. doi: 10.1109/TVCG.2018.2864899

[51] L. Törnqvist, P. Vartia, and Y. O. Vartia. How should relative changes be measured? *The American Statistician*, 39(1):43–46, 1985. doi: 10.2307/2683905

[52] S. van den Elzen, D. Holten, J. Blaas, and J. J. van Wijk. Reordering massive sequence views: Enabling temporal and structural analysis of dynamic networks. In *2013 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 33–40, Feb 2013. doi: 10.1109/PacificVis.2013.6596125

[53] S. Van Den Elzen and J. J. van Wijk. BaobabView: Interactive construction and analysis of decision trees. *IEEE conference on Visual Analytics Acience and Technology*, pp. 151–160, 2011. doi: 10.1109/VAST.2011.6102453

[54] V. Waddle. Graph layout for displaying data structures. In J. Marks, ed., *Graph Drawing*, pp. 241–252, 2001. doi: 10.1007/3-540-44541-2_23

[55] T. D. Wang, C. Plaisant, A. J. Quinn, R. Stanchak, S. Murphy, and B. Shneiderman. Aligning temporal data by sentinel events: discovering patterns in electronic health records. In *Proc. SIGCHI conference on Human factors in computing systems*, pp. 457–466, 2008. doi: 10.1145/1357054.1357129

[56] M. Q. Wang Baldonado, A. Woodruff, and A. Kuchinsky. Guidelines for using multiple views in information visualization. In *Proc. working conference on Advanced visual interfaces*, pp. 110–119, 2000. doi: 10.1145/345513.345271

[57] J. N. Warfield. Crossing theory and hierarchy mapping. *IEEE Transactions on Systems, Man, and Cybernetics*, 7(7):505–523, 1977.

[58] J. I. Wolfsdorf. *Intensive Diabetes Management*. American Diabetes Association, fifth edition ed., 2012.

[59] K. Wongsuphasawat and D. Gotz. Exploring flow, factors, and outcomes of temporal event sequences with the Outflow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2659–2668, 2012. doi: 10.1109/TVCG.2012.225

[60] K. Wongsuphasawat, J. A. Guerra Gómez, C. Plaisant, T. D. Wang, M. Taieb-Maimon, and B. Shneiderman. LifeFlow: visualizing an overview of event sequences. *Proc. SIGCHI conference on human factors in computing systems*, pp. 1747–1756, 2011. doi: 10.1145/1978942.1979196

[61] Y. Wu, N. Pitipornvivat, J. Zhao, S. Yang, G. Huang, and H. Qu. egoSlider: Visual analysis of egocentric network evolution. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):260–269, 2016. doi: 10.1109/TVCG.2015.2468151

[62] D. C. Zarate, P. L. Bodic, T. Dwyer, G. Gange, and P. Stuckey. Optimal Sankey diagrams via integer programming. In *2018 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 135–139, April 2018. doi: 10.1109/PacificVis.2018.00025

[63] Y. Zhang, K. Chanana, and C. Dunne. IDMVis: Temporal event sequence visualization for type 1 diabetes treatment decision support. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):512–522, Jan 2019. doi: 10.1109/TVCG.2018.2865076

[64] Y. Zhang, S. Di Bartolomeo, F. Sheng, H. Jimison, and C. Dunne. Evaluating alignment approaches in superimposed time-series and temporal event-sequence visualizations. In *2019 IEEE Visualization Conference (VIS)*, pp. 1–5, Oct 2019. doi: 10.1109/VISUAL.2019.8933584

[65] Z. Zhang, B. Wang, F. Ahmed, I. Ramakrishnan, R. Zhao, A. Viccellio, and K. Mueller. The five Ws for information visualization with application to healthcare informatics. *IEEE transactions on visualization and computer graphics*, 19(11):1895–1910, 2013. doi: 10.1109/TVCG.2013.89

[66] J. Zhao, Z. Liu, M. Dontcheva, A. Hertzmann, and A. Wilson. MatrixWave: Visual comparison of event sequence data. *Proc. 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 259–268, 2015. doi: 10.1145/2702123.2702419

A USAGE EXAMPLES

Here we show several examples of SEQUENCE BRAIDING used with different datasets. These examples do not necessarily share the same complex data properties of the diabetes use case motivating our design (see section 8.3), and thus some may be amenable to visualization using other approaches, e.g., Sankey or icicle visualizations.

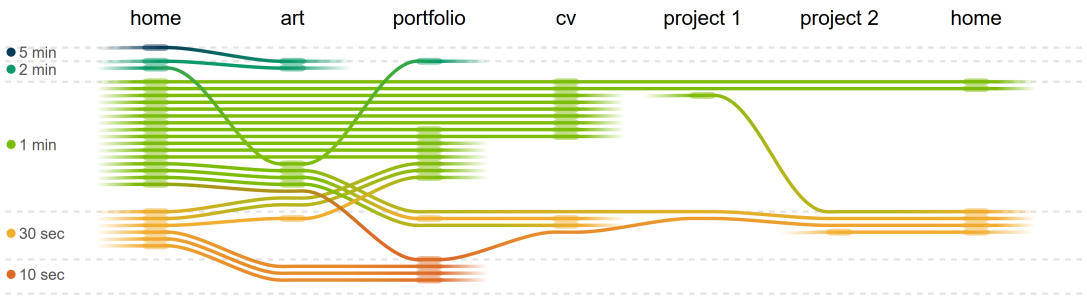

Fig. 10: This figure shows the sequences of pages visited on one of the authors' personal website from March 1, 2019 to March 1, 2020. Each sequence represents 5 users who visited the same sequence of pages, and the timings are averages within the group. Navigation sequences from less than 5 users were filtered out. We filtered out users who visited a single page. The *y* axis represent the time spent on each page, while the events are the pages visited. From the visualization, it can be deduced that users who start looking at pages quickly (30 seconds) tend to do a quick tour and do not end up on pages that attract their attention enough to spend more time on them. Most users spend about 30 seconds on each page. Some users spent a long time on the home page, then proceeded to spend a long time on the art page too.

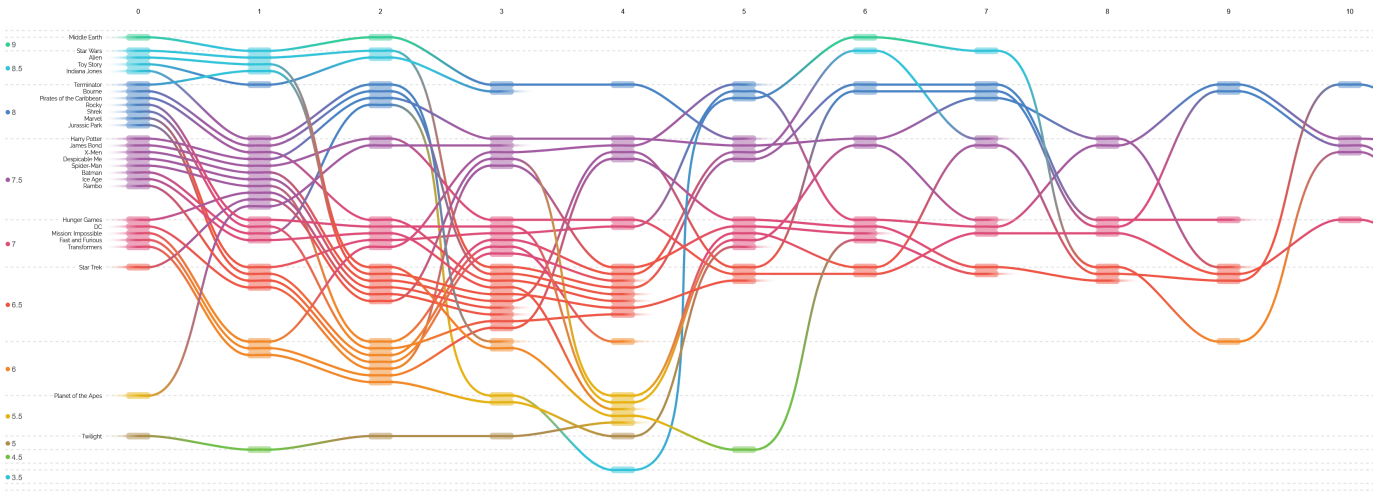

Fig. 11: IMDb rating of movies by movie franchise. The vertical axis is the rating, rounded to the nearest half integer. A node represents a movie, and the position of a node in a sequence represents the index of the movie within the franchise. Movies within a franchise are ordered according to their release date (e.g., Star Wars is 4, 5, 6, 1, 2, 3, etc.).

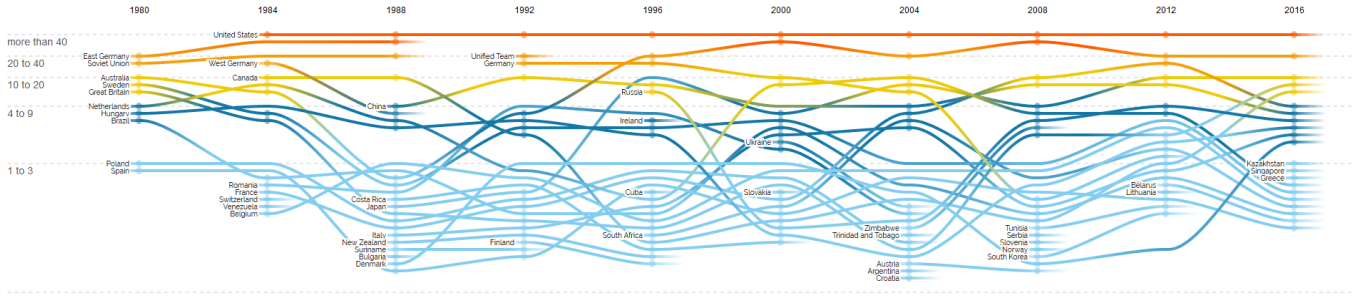

Fig. 12: Number of medals obtained by different countries in the swimming category of the Olympics, throughout the years.

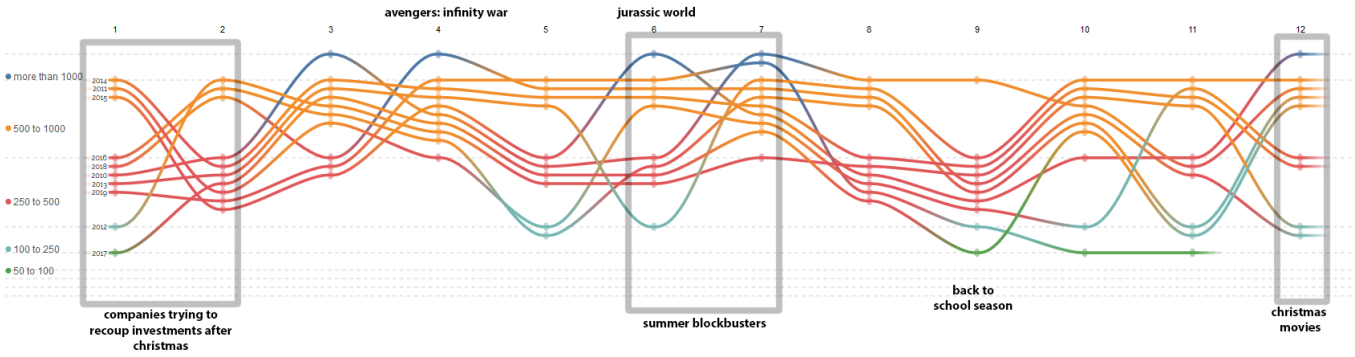

Fig. 13: "Dump months" [a] are a phenomenon known in the movie industry where there are lowered critical and commercial expectations from new movie releases. January and February usually see companies trying to recoup investments done during Christmas, therefore avoiding major releases during that period, while during August and September — the back to school season — companies are both trying to recoup investments from the summer blockbusters released earlier in the summer, and sales are affected by other expenses such as tuition payments and school supplies. The visualization shows the sum of the gross income for the whole domestic movie industry in millions of dollars, per year, from 2010 to 2019. Each node represents the sum, and not a single movie — annotations at the top show movies that had a particularly outstanding income. Note that there are several outliers.

[a] https://en.wikipedia.org/wiki/Dump_months

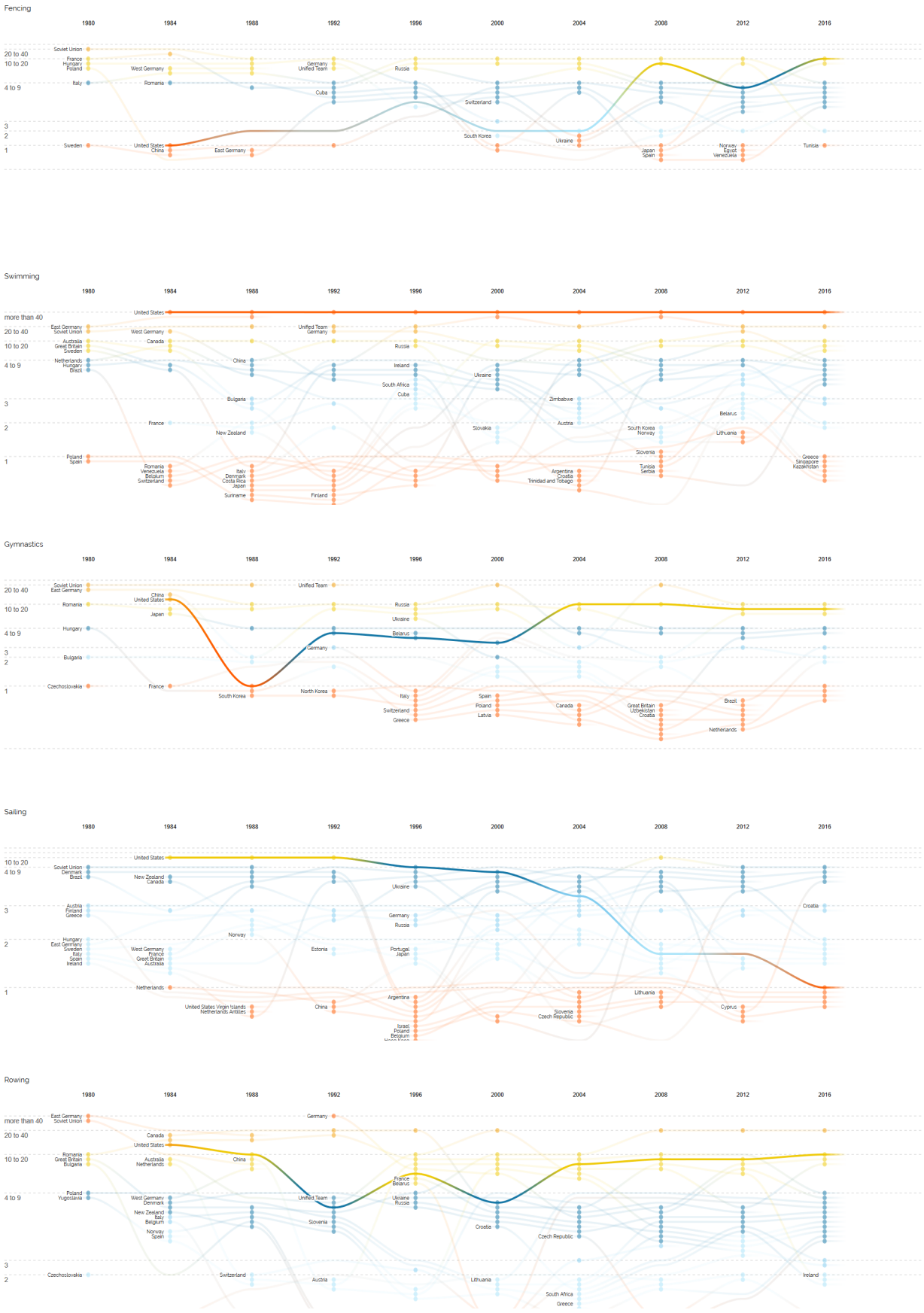

Fig. 14: US performance in various Olympic sports through the years, compared to other countries. The vertical axis is the sum of medals obtained. The US did not participate in the 1980 Olympics.

B Integer Linear Programming formulation

Here we describe how we formulated the intersection reduction problem as an integer linear programming (ILP) problem. Our formulation is largely based on Zarate et al.'s optimal intersection reduction algorithm for sankey diagrams [62], that we deemed most similar to our case. However, differently from Zarate et al., this integer linear programming formulation has:

- An unweighted directed graph instead of a weighted one.
- Ordered grouping constraints. Although Zarate et al. briefly discuss grouping constraints in their paper, Sequence Braiding defines an order between the groups that must be respected throughout the ordering process.

In addition to these difference, we also want to note that the rank assignment step of our algorithm is part of the contribution of Sequence Braiding, that is not covered in this integer linear programming formulation nor in Zarate et al.'s formulation.
The point of this appendix section is to prove that a integer linear programming approach is only suitable for smaller graphs, and would not scale as much as we wanted, while the benefits it provides are not enough to compensate for the lack of scalability.

B.1 Definitions

Throughout this section, we keep our terminology and notation as similar as possible to Zarate et al. [62] for consistency.
Symbols:

- $G = (V, E)$ is a directed acyclic graph with unweighted edges. The nodes of G are partitioned into k ranks.
- $\mathscr{L} = \{0, \ldots, k-1\}$ is the set of all layers, while $\mathscr{L}_1 = \{0, \ldots, k-2\}$ is the set of all layers excluding the last one.
- V_k is the set of all nodes in layer $k \in \mathscr{L}$
- E_k is the set of all directed edges that have a source in k and a target in $k+1$.
- uv is an edge between nodes u and v.
- R is the set of all groups. The groups have a fixed and predefined ordering across all layers.

For each edge in the original graph that extends over multiple ranks, we add without loss of generality an anchor node for each traversed rank and partition the edge into multiple, smaller edges so that each one only connects nodes in two consecutive layers. Therefore, we can assume that for any edge uv, $u \in k$ and $v \in k+1$.

Each node in each rank that is not an anchor has a preassigned level $\ell \in 0, \ldots, m$, and m is the number of layers in the graph. The levels are similar to groupings, except they have a pre-established ordering. R is the set of all levels in the graph.

Decision variables:

- $x_{u_1,u_2} \in \{0,1\}$ is a binary variable that indicates the relative position of two nodes, specifically whether node u_1 is above node u_2 in layer k, for any given u_1 and u_2 in all $k \in \mathscr{L}$. $x_{u_1,u_2} = 1$ if u_1 is above u_2, 0 otherwise.
- $c_{u_1v_1,u_2v_2} \in \{0,1\}$ is a binary variable indicating whether edges u_1v_1 and u_2v_2 cross.

B.2 Formulation

We define the objective function to minimize the sum of the crossings over every pair of edges.

$$\text{Minimize}: \sum_{\substack{k \in \mathscr{L}_1 \\ u_1v_1, u_2v_2 \in E_k \\ u_1v_1 \neq u_2v_2}} c_{u_1v_1,u_2v_2} \tag{1}$$

N		heuristic		ILP	
sequences	edges	cross.	time	cross.	time
5	35	12	0.01s	11	0.02s
10	70	44	0.01s	36	0.03s
15	105	117	0.02s	106	0.05s
25	175	368	0.03s	358	0.18s
50	350	1542	0.11s	1494	1.97s
75	525	3521	0.23s	3352	7.73s
100	700	6331	0.39s	6188	22.42s
125	875	10207	0.56s	9842	57.69s
200	1400	25199	1.76s	-	-
500	3500	156171	18.54s	-	-

Table 1: Comparison between timing and number of crossings obtained by the heuristic-based approach and the integer linear programming approach. Empty cells represent when the computation was unable to terminate (out of memory).

B.3 Constraints

Either u_1 is above u_2, or u_2 is above u_1.

$$x_{u_1,u_2} + x_{u_2,u_1} = 1 \qquad \forall k \in \mathscr{L}, u_1, u_2 \in V_k, u_1 \neq u_2 \tag{2}$$

Transitivity of the "above" relation.

$$x_{u_3,u_1} \geq x_{u_3,u_2} + x_{u_2,u_1} - 1 \qquad \begin{cases} \forall k \in \mathscr{L}, u_1, u_2, u_3 \in V_k \\ u_1 \neq u_2 \neq u_3, u_1 \neq u_3 \end{cases} \tag{3}$$

For each two edges u_1v_1 and u_2v_2, they cross if and only if u_1 is above u_2 and v_2 is above v_1.

$$c_{u_1v_1,u_2v_2} + x_{u_2,u_1} + x_{v_1,v_2} \geq 1 \tag{4}$$

$$c_{u_1v_1,u_2v_2} + x_{u_1,u_2} + x_{v_2,v_1} \geq 1 \tag{5}$$

Each node uniquely belongs to a group, except anchors, that do not belong to any group. The groups have a fixed, predefined ordering. Each node in group g_i must be above all the nodes in group g_{i+1}.

$$x_{u_1,u_2} = 1 \qquad \forall u_1 \in g_i, u_2 \in g_{i+1}, g_i, g_{i+1} \in R \tag{6}$$

Here, eq. (6) is the real difference from Zarate et al. [62]'s formulation, in addition to the absence of weights. In their formulation, they do mention grouping constraints, but while groups in Sequence Braiding have a fixed ordering across all ranks, groups in their work are set of nodes that must be positioned next to each other, independent from the position of the other nodes in the same layer.

B.4 Performance comparison

We tested and compared the heuristic-based approach and the integer linear programming approach on our chess case study from the paper. The integer linear programming solution was solved using Gurobi version 9.0[3], the same integer linear programming solver used by Zarate et al. [62], running on Windows 10. All the computation was performed locally on a 2018 Dell XPS 15, with an Intel Core i7-8750H @ 2.20GHz, 16GB RAM. Sequence Braiding was executed in 64 bit Firefox 74.0.1 running on Windows 10. Table 1 reports the times and number of crossings produced with different sized graphs. The sequences used where the first n sequences from our *chess openings* dataset example used in fig. 7. We also have to note that there's a fundamental hindrance to the heuristic solution: it is implemented in JavaScript and running in a browser, as opposed to the integer linear programming solution computed via Gurobi, a detail that significantly negatively impacts the performance of the heuristic solution. However, since the timing comparisons are already largely in favor of the heuristic-based approach, this detail further shows how much faster the heuristic-based approach is.

[3] https://www.gurobi.com/

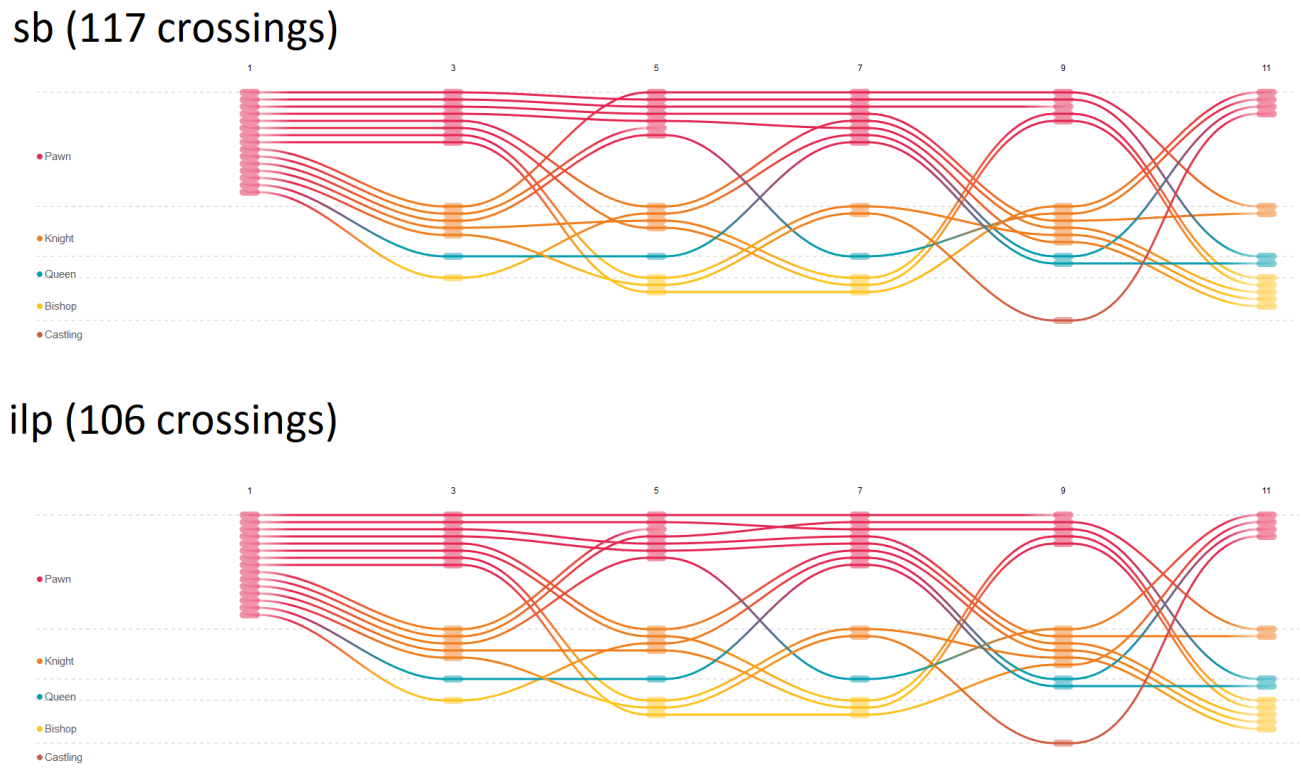

Fig. 15: Comparison between the outputs produced by solving a integer linear programming problem (ilp) and a heuristic-based approach (sb) on 15 sequences.

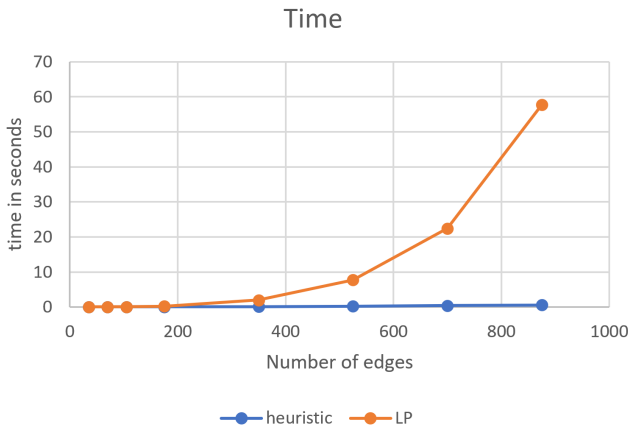

Fig. 16: Timing comparison of the heuristic approach and the integer linear programming approach, from table 1. Cases where the ILP approach could not terminate the computation are filtered out.

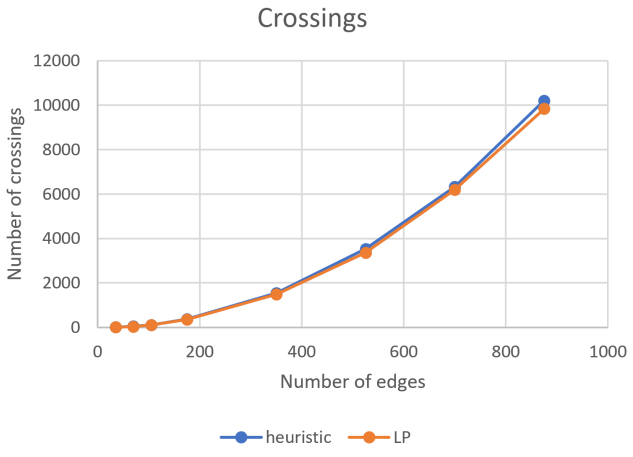

Fig. 17: Comparison of the number of crossings in the solutions found by the heuristic approach and the integer linear programming approach, from table 1. Cases where the ILP approach could not terminate the computation are filtered out.

We added a function into SEQUENCE BRAIDING so that we could generate, from any SEQUENCE BRAIDING example, a integer linear programming formulation of the problem suitable as input for Gurobi, and another function that takes as input a solution produced by Gurobi and represents it as a SEQUENCE BRAIDING visualization. Figure 15 shows an example of the outputs produced by the two methods.

B.5 Discussion

Although the solution found with the integer linear programming approach is optimal, it does not scale as well as the heuristic-based one and is therefore not suitable for large graphs.

As shown in fig. 15, the visual complexity of the output produced by the integer linear programming approach is not enough to justify such a loss in scalability. Figure 16 shows how different is the way in which the two approaches scale in terms of time used to find a solution. Figure 17, instead, shows that the amount of crossings found in each solution is, in the end, not that different: the line representing the integer linear programming approach is better, but only by a small amount.

We deemed that, for our objectives, it was ultimately more important to provide a representation of bigger graphs rather than the optimal intersection reduction solution.

C COMPARATIVE EVALUATION: BASELINE ALTERNATIVES

Before deciding to settle on IDMVis [63] as a baseline for our comparative evaluation, we tested several other alternatives. The following sections explain why we deemed these other options unfeasible.

C.1 Storylines

The project we focused the most on is Storylines [38] due to apparent similarities in purpose to our project.

Storylines does not align nodes based on event type — the horizontal position of the nodes is only based on time. It does have grouping constraints based on a node attribute. The colors are applied by sequence and not by attribute, and the distribution of the event types for every time step is presented using a barchart at the bottom of the visualization. The lack of an option to see the value of the node attribute directly on the sequence meant that we had to show the node label at each node to let users to distinguish nodes with different attributes, adding visual complexity to the final visualization.

Overall, the sample dataset contained in Storylines works well (fig. 18), but making our examples readable required such extensive edits to the source code and design of Storylines that we worried we were departing too much from a viable original baseline. Our initial modified version of Storylines, with a representation of our diabetes dataset, can be seen in fig. 19. The absence of alignment by event and the complexity of understanding the attribute values led us to us discard Storylines as an option.

C.2 Storyflow

Storyflow [30] has the same issue as Storylines: there is no event alignment, the alignment is based on the time. Additionally, in Storyflow, the sequences are bundled together based on an attribute — in fig. 20, this attribute is the co-occurrence of two characters in a scene. This does not allow for an attribute to be displayed on the vertical axis.

C.3 Sequence Bundles

In Sequence Bundles [27] (example in fig. 21) the vertical position of the nodes encodes a node attribute and the horizontal position is the position in the sequence. Individual sequences overlap if they have a node with the same attribute value at the same position in the sequence, making keeping track of individual sequences very difficult. The bundling in Sequence Bundles is not based on event types — the event types are only encoded on the vertical axis as the categorical attribute values — but on position in the sequence. This means that the first event is always going to be aligned with the first event, the second with the second, and so on, regardless of the type. This effectively means that one less attribute is able to be encoded in the visualization. While in SEQUENCE BRAIDING we display for each event both a type and a level, Sequence Bundles would only be able to display one of the two attributes.

Fig. 18: One of the standard Storylines samples, unedited [38].

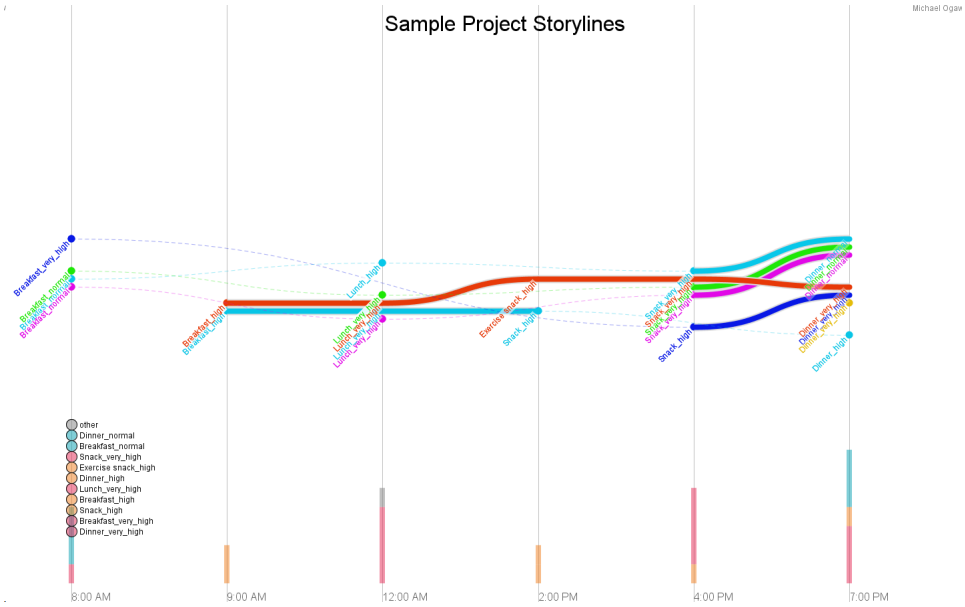

Fig. 19: A subset of our diabetes dataset represented with Storylines [38] after extensive edits to load and represent the dataset.

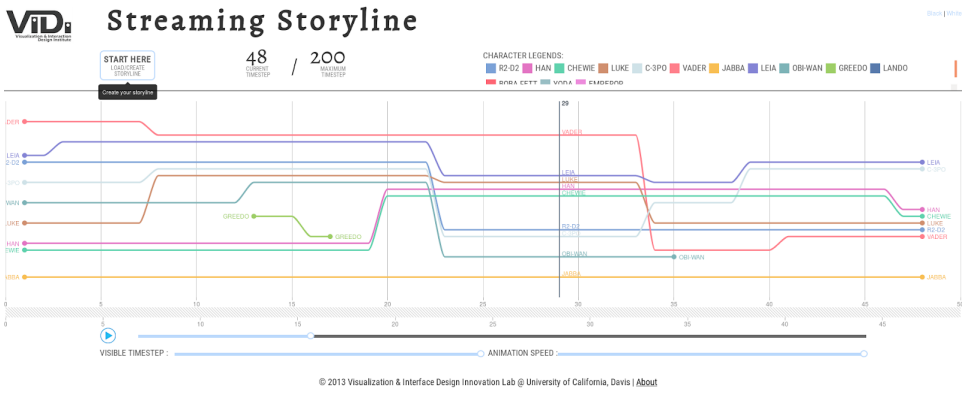

Fig. 20: Storyflow visualizing one of their standard examples [30]. Star Wars characters are grouped together based on an attribute — being in the same scene of the movie.

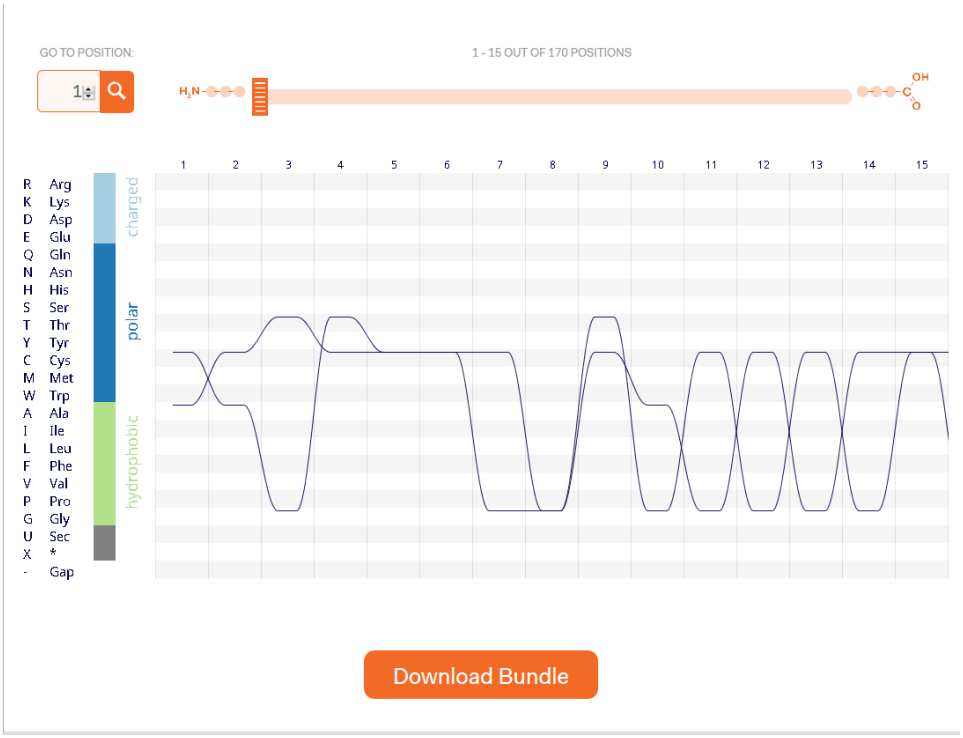

Fig. 21: Sequence Bundles visualizing one of their standard examples [27]. Note the overlapping sequence lines at the various combinations of horizontal sequence position and vertical attribute value.