A Literature-based Visualization Task Taxonomy for Gantt Charts

Sayef Azad Sakin*   Katherine E. Isaacs†

SCI Institute and Kahlert School of Computing, The University of Utah

ABSTRACT

Gantt charts are a widely-used idiom for visualizing temporal discrete event sequence data where dependencies exist between events. They are popular in domains such as manufacturing and computing for their intuitive layout of such data. However, these domains frequently generate data at scales which tax both the visual representation and the ability to render it at interactive speeds. To aid visualization developers who use Gantt charts in these situations, we develop a task taxonomy of low level visualization tasks supported by Gantt charts and connect them to the data queries needed to support them. Our taxonomy is derived through a literature survey of visualizations using Gantt charts over the past 30 years.

Index Terms: Gantt chart—Visualization—Task taxonomy.

1 INTRODUCTION

Gantt charts are a visual idiom for displaying multiple event sequences with dependencies between events in different sequences. Data is organized on two axes, one representing a notion of time or order and the other partitioning the events by some factor relating them, such as different people, roles, or resources assigned to them.

Gantt charts are widely used in project planning, process scheduling, and progress tracking [40]. Project managers use Gantt charts for planning employee work assignments and estimating project completion time [39]. Gantt charts are similarly used in manufacturing for scheduling multi-stage pipelines [21].

Medical practitioners have used Gantt charts to understand treatment responses across individuals [4]. In parallel computing, Gantt charts are used to examine inter-resource dependencies and complex program behavior [20].

However, many of these applications, especially those coming from automated processes such as manufacturing and computing, can generate data with billions of events across tens of thousands of independent sequences—a scale beyond most Gantt chart implementations. A variety of designs have been proposed for when the data can no longer be resolved in a single image. Often the strategy involves aggregating or eliding data and implementing interactions such as zooming, panning, and filtering to present (a subset of) the data in the intuitive Gantt chart form. However, poor support for scale continues to be a significant barrier [7, 20] and querying these large-scale datasets to implement the existing techniques can lead to large latencies, further hampering exploratory analysis [24].

To better understand strategies and barriers to using Gantt charts with large numbers of events or sequences, we develop a literature-based task taxonomy, focusing on low-level visualization tasks that Gantt-focused systems use to support scale. Our taxonomy includes not only the Gantt-based tasks, but also those of highly coupled auxiliary views supporting the analysis of Gantt data. This taxonomy can be matched with user tasks when designing with Gantt charts in single or multi-view systems. We also match these tasks with data

---

*email: sayefsakin@sci.utah.edu

†e-mail: kisaacs@sci.utah.edu

queries supporting them, enabling the analysis of data management techniques for supporting interactions as the datasets scale up. We expect visualization creators to use this taxonomy when considering options for representing large scale Gantt data and when considering implementation needs.

In summary, our contributions are:

- A multi-layer **visualization task taxonomy** for interactive Gantt charts and their common auxiliary views,
- A **classification of data queries** and their relationship with the proposed visualization task taxonomy for representing discrete event data visualized by Gantt charts, and
- A **literature review** of scholarly articles over the last 30 years that have used Gantt charts as significant view.

2 GANTT CHART TERMINOLOGY AND RELATED WORK

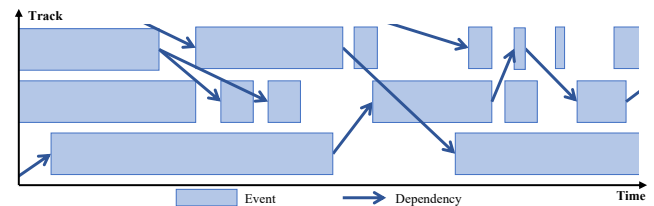We discuss Gantt chart data, encoding, and related surveys.

Figure 1: Gantt chart showing a window of time and three tracks.

Data. The data items represented by Gantt charts are discrete *events* that have a partial order, typically due to some form of temporal attribute. We will refer to this ordering attribute as *time*. Thus, Gantt chart data fall into the general category of *event sequence data*. Gantt charts are typically used when the events have a *duration*, i.e., the start and end times are different.

The events are typically divisible into multiple sequences by one or more of their attributes, e.g., events performed by different people can be organized as per-person sequences.

The literature has used several terms for the attribute used to divide sequences, such as schedule, location, resource, person, process, or task. We choose the term *track* as it is still general but less overloaded in comparison to others (e.g., location, schedule, task, process). While time has an intrinsic order, tracks may have zero or more meaningful orderings.

Gantt charts are commonly used when there are relationships between events of different tracks. For example, in parallel computing, processor 1 might execute a function (event) only after receiving a message created by a function (event) on processor 2. We refer to these kinds of cross-track relationships as *dependencies* and the events they connect as *dependent events*. The set of dependencies and dependent events together forms a *dependency graph*.

Events may have several more attributes associated with them in addition to the temporal, track, and dependency attributes. Some datasets may have multiple attributes that could be considered for time or track. While the track is an event attribute, it may also be treated as the central data unit for another view. For example, if

tracks are employees, "employees" could be the data item of focus for another visualization. Thus, tracks can have additional attributes associated with them.

Encoding. The most prevalent form of a Gantt chart (Figure 1) is a two-dimensional visualization where time is mapped to the horizontal axis and tracks are displayed on the vertical axis. Each event is represented as a rectangular bar drawn according to its start time, end time, and track along these axes. Thus, the length of the bar represents its duration. Interactive Gantt charts allow panning and zooming on both axes.

Other arrangements exist, such as swapping the horizontal and vertical axes [9] or creating three-dimensional Gantt charts with a two-dimensional track layout, sometimes for symmetry (e.g., a circular layout [10]) or geographical [35] relations between tracks. We focus on the two dimensional form only in this work.

Dependencies between events are encoded as line segments or arrows connecting their associated dependent events.

Our examples use straight lines for dependencies, but orthogonal edges are sometimes used, especially in commercial project management tools [2].

While the organization of bars is the most salient feature of a basic Gantt chart, frequently auxiliary encodings focus on the dependencies. By making the dependencies and dependent events more salient, the displays becomes more like a node-link diagram of the dependency graph.

Using a dependency graph to help manage the scale and complexity of the Gantt chart is a common strategy found in our literature review and thus we included it in our analyses.

Related Work. Several works [2,14,36] survey temporal or event sequence data broadly, providing a general overview of time-oriented data visualization techniques.

Parallel computing-specific surveys [12,20] demonstrate Gantt charts for large-scale trace data among others. Peiris et al. [29] developed a task taxonomy for timestamped instantaneous events without dependencies. Our work goes further in depth with Gantt charts specifically.

3 GANTT CHART TASK TAXONOMY AND DATA QUERIES

We conduct a survey of Gantt charts used in visualizations with an emphasis on the visual tasks supported. We present our methodology, the derived tasks, and a translation of the tasks discovered into a set of data queries. Figure 2 presents a heatmap of our visualization tasks and their mapping to data queries where each cell indicates in how many papers the task was observed.

3.1 Methodology

We obtained an initial list of papers from surveys that feature Gantt charts [12,14,20]. We then expanded our list through searches in IEEE Xplore and Google Scholar using the following keywords, sourced from the surveys: *Gantt chart*, *timeline visualization*, *event sequence visualization*, *schedule visualization*, *execution trace visualization*, *trace analysis*, *performance analysis*, *performance visualization*, *parallel program analysis*, *task parallel trace analysis*, and *time-series charts*. We collected papers with promising abstracts, resulting in 137 research articles. We filtered these papers to those including interactive Gantt charts, resulting in a set of 35 papers.

We also identified 11 additional commercially available online tools that use Gantt charts (see supplemental materials). We did not find additional tasks by investigating their websites. We do not include them in Figure 2 as their Gantt view features were not described in detail in their publicly available documentation.

For each paper, we examined at the description of use, task analyses if they existed, the visualization itself, and any use cases or evaluation. The initial round of qualitative coding focused on identifying, distilling, and tagging how the Gantt chart was used—what visualization tasks it was supporting. We then made a second pass

to merge, split, and elaborate on the codes. This two-round process resulted in 29 distinct low-level visualization tasks. We categorized them into groups based on their intended higher-level user goal.

3.2 Gantt chart task taxonomy

We define the 29 visualization tasks, T1 - T29, collected into eight groups by their higher level user goals. We cite example sources from the literature leading to each task. The full list of sources for each task are presented in the supplemental material.

These are *visualization* tasks, focused on the functionality the Gantt chart and its closely related auxiliary views need to provide over what the user is trying to accomplish. Choosing this perspective supports both the evaluation of data management strategies and tasks to support when implementing Gantt charts.

3.2.1 Event Overviews

These tasks involve viewing some or all of the discrete event data, often to get an overview or general context.

T1: Display the full Gantt chart. This is the most basic draw operation of the Gantt chart. For all but the smallest datasets, rendering the chart to fit on screen will involve some sort of aggregation, filtering, or overplotting, but we assume in our data queries that all of the data must be accessed. Often, visualization designers use a highly aggregated view as an overview to help with further navigation [22,27,31,42].

T2: Display event attributes. Basic Gantt charts only directly encode time and track attributes. Color and labels are sometimes used to show additional attributes or auxiliary views may be used. Gantt charts often support requesting additional attributes interactively, initiated by brushing over the chart [3,26] or by clicking [30] or hovering [9] on individual bars. The attributes are then shown in a separate static view [22,26,33], multiple coordinated views [9,27,33], via lenses [21,25,27], or tooltips [4,9,21].

T3. Display a window. This task shows a subset of the data in time and/or track space. It can be an outcome of other tasks (e.g., T10 - Browsing) or of directly specifying the window. Common modes of access include brushing over time [1,33], zooming, panning, textual input, or selecting through a derived dependency diagram [9].

T4. Re-encode. This tasks describes alterations of the Gantt encoding, such as mapping an axis to a different attribute set [15,30] or changing the domain of an existing axis, such as switching the notions or calculation of time [19,21,42].

T5. Compare events from multiple datasets. This task presents multiple datasets. This is presented either with an additional window [22,33] or within the same Gantt window [5].

3.2.2 Analyze Derived Data

Users may wish to explore derived data, such as summaries, aggregations, or clusters. These can help users understand their data at a higher level than individual events.

T6. Display event distributions. This task provides context to event attributes. Common methods include showing percentages of the whole [1], histograms [33], or outliers to some condition [21,34].

T7. Aggregate events. This task combines multiple events into a derived value such as the average, minimum, and maximum of certain attribute values, often time. Gantt chart-based visualizations have displayed the aggregated events several ways, including with a separate aggregate Gantt view [22,28,33], a magnification lens [21], re-encoding the chart area [38], and using juxtaposed marks [15].

T8. Derive event attributes from dependencies. Some analyses involve calculating metrics based on dependencies. For example, in computing contexts, 'inclusive time' can be computed by summing durations of dependencies of an event. These derived metrics may be presented within the Gantt chart or separately [1,33].

T9. Display patterns. Patterns of events and dependencies that can be defined, identified, and displayed [9,19] for some data.

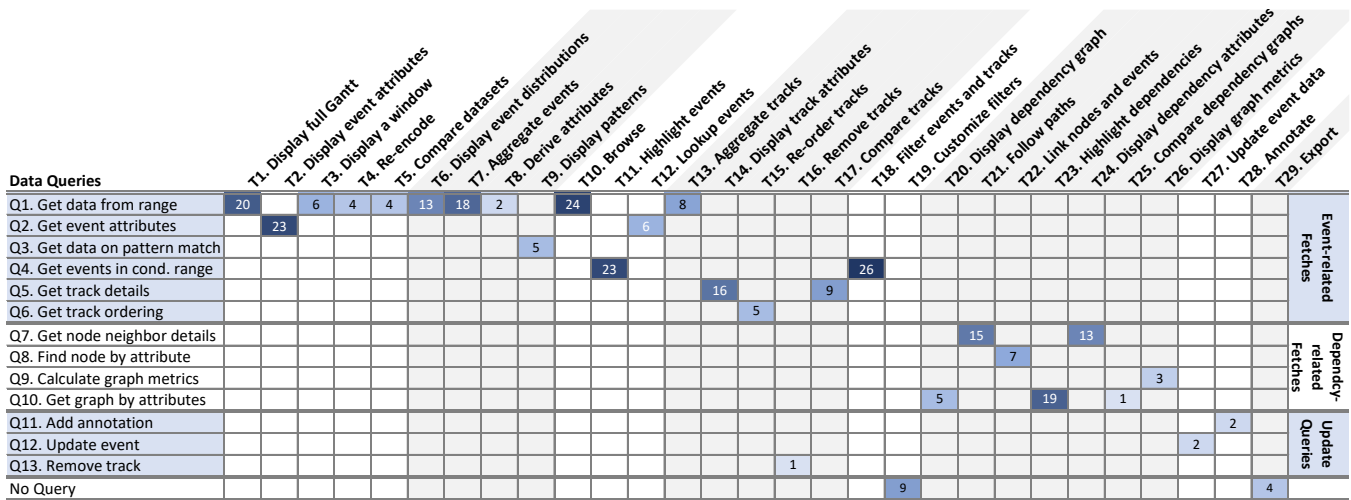| Data Queries	T1. Display full Gantt	T2. Display event attributes	T3. Display a window	T4. Re-encode	T5. Compare datasets	T6. Display event distributions	T7. Aggregate events	T8. Derive attributes	T9. Display patterns	T10. Browse	T11. Highlight events	T12. Lookup events	T13. Aggregate tracks	T14. Display track attributes	T15. Re-order tracks	T16. Remove tracks	T17. Compare tracks	T18. Filter events and tracks	T19. Customize filters	T20. Display dependency graph	T21. Follow paths	T22. Link nodes and events	T23. Highlight dependencies	T24. Display dependency attributes	T25. Compare dependency attributes	T26. Display dependency graphs	T27. Display graph metrics	T28. Update event data	T29. Annotate	Export	
Q1. Get data from range	20		6	4	4	13	18	2		24		8																			Event-related Fetches
Q2. Get event attributes		23										6																			
Q3. Get data on pattern match									5																						
Q4. Get events in cond. range											23							26													
Q5. Get track details														16			9														
Q6. Get track ordering															5																
Q7. Get node neighbor details																					15		13								Dependency-related Fetches
Q8. Find node by attribute																						7									
Q9. Calculate graph metrics																											3				
Q10. Get graph by attributes																				5				19		1					
Q11. Add annotation																													2		Update Queries
Q12. Update event																												2			
Q13. Remove track																1															
No Query																			9											4	

Figure 2: Mapping between data queries and interactive visualization tasks for Gantt charts. Each heatmap cell denotes in how many papers we observed the given visualization task. Tasks observed in each paper are supplemental materials. These counts suggest common tasks to support and possibilities for handling scale. The counts indicate which queries are the most common and may be considered for optimization.

3.2.3 Search

These tasks support various forms of searching.

T10. Browsing. This task encompasses zooming and panning in the Gantt chart, leading to T3 (displaying a window). Most of the visualizations in our literature review support this task.

T11. Highlight events. Highlighting events based on some attribute value is prevalent in Gantt chart visualization and is essentially a locate-type user task. It is complementary to event filtering (T18). Commonly, events are highlighted with color [1, 6, 19, 22, 28, 33, 42], but other encodings, such as additional visual marks [37] or re-organizing the Gantt bars [21, 25], have also been used.

T12. Lookup events. This task involves displaying the context of a particular event within the Gantt chart and/or auxiliary views. It is often the outcome of selection in one view causing related views to update. For example,

clicking on a line of code could re-center the Gantt chart to the corresponding bar. Alternatively, clicking on a Gantt bar could highlight source code [1, 41, 42] or related geographic data in a map [15] (e.g., where the event occurred).

3.2.4 Manipulating Tracks

These tasks involve manipulating tracks in some way, typically interactively. Users may want to manipulate tracks to re-organize the data, simplify the view, or do comparisons.

T13. Aggregate tracks. This task combines multiple tracks, often requiring aggregating the events among them. The task can be pre-computed [28, 33, 37] or performed interactively [21, 22, 32].

T14. Display track detail. This task helps users understand how a track attributes may relate to event data. Gantt charts have supported showing track details through tooltips [37], track-focused lenses [21], color encoding [35], highlighting [1, 11], and drawing additional rows to contain the detailed data [32].

T15. Re-order tracks. Users may want to change the order in which tracks are presented, manually or by attribute. Implementations include draggable rows [8], re-ordering by a specific event attribute [17], and inputting a re-ordering condition through a separate interface [9, 21].

T16. Remove tracks. This task enables users to completely remove individual tracks from the chart. Of our surveyed papers, only Gupta et al. [15] support this task.

T17. Compare tracks. Though users can compare tracks if they are all already on screen, additional measures can be taken to ease visual comparison. Techniques like repositioning tracks [37] or hiding unrelated tracks [22, 23, 39] ease direct comparison. Pinto et al. [31] added percent usage summaries to each track to support aggregate comparison.

3.2.5 Filtering

These tasks support users focusing on specific events.

T18. Filter events and tracks. This task enables filtering events and/or tracks depending on event attributes. For example, in the medical domain this task could involve filtering patients (tracks) with a specific disease progression. In the computing domain, this task might be to filter down to events matching a certain library. This task was implemented by most of our surveyed papers with filtering in time being the most common.

T19. Customize filters. Complicated filters often require some sort of an interface to construct. Several works [11, 13, 21, 22, 30] add a dedicated window with dropdown lists, checkboxes, and sliders to support building these filters.

3.2.6 Exploring dependencies

These tasks support analyzing dependencies. They may involve the dependencies as drawn in the Gantt chart or an auxiliary view.

T20. Display dependency graph. Rather than view only the Gantt chart which prioritizes events and tracks, users may want to focus on a graph (network) that can be constructed from the dependencies. Nodes in this graph can be events or, when shown in a separate view, aggregated groups of events. Displaying the graph can support identifying motifs in the graph that indicate some sort of behavior or pattern. In the literature, displaying the graph has been supported by de-emphasizing the other marks [35] or using an additional view for the graph [1, 9, 33].

T21. Follow paths. This task supports users in following chains of dependencies. It has been supported through Gantt chart navigation [16, 18] or linked through the external graph view [4, 22, 33]. Interactive features, such as collapsing sub-graphs [1, 9, 33] further help to navigate through a very large graph structure.

T22. Link nodes and events. In the full dependency graph, nodes are events. However, manipulating the graph can lead to nodes that represent aggregated events. This task links the node back

to its composite events, typically as a color highlight [9, 33] of both the graph nodes and the Gantt bars.

T23. Highlight dependencies. This task draws a set of dependencies in the Gantt chart (and dependency graph if separate). Often, this selection is done with a specific path (relating to T21) or sub-graph [6, 16, 19, 33, 35, 39].

T24. Display dependency attributes. This task enables representing additional details regarding dependencies in the Gantt chart (and dependency graph if separate). This is often done by color encoding [11, 23] and additional visual marking [4, 6, 9, 19].

T25. Compare dependency graphs. Multiple dependency graphs can arise when splitting a dependency graph or considering other event attributes as dependency sources. This task encompasses techniques supporting their comparison. Sun et al. [37] used a separate component view with treemaps to support this task.

T26. Display graph metrics. Statistics about the dependency graph, such as depth or total number of nodes and edges, can lead to insights about the data. Notions such as the distance between two given nodes or their common ancestor may also be sought. These values are generally shown in a separate detail view [1, 35, 39].

3.2.7 Modifying the data

These tasks support users in altering the dataset, typically for annotation or to clean or fix the data.

T27. Updating event data. This task allows users to change event attribute data. This has been implemented by overwriting the data when a user re-positions an event [5, 21].

T28. Annotate. This task involves adding text and additional markings to the chart as annotations. Annotation data is separate from the base temporal event sequence data. Multiple tools [11, 15] support this task.

3.2.8 Exporting

User may want to save the visualization outside of the tool.

T29. Export. Several visualizations [11, 30, 33, 42] provide a feature to export in a shareable format such as a PNG or SVG.

3.3 Data queries for Gantt chart

We organize the data queries required for the tasks based on the similarity of their data access profile. Note T19 (filter customization) and T29 (export) do not require fetching additional data.

3.3.1 Event-Related Fetch Queries

These queries fetch data associated with events.

Q1. Get data from a range. The range may be multi-dimensional, range e.g., time and track. *Input:* A set of range tuples for each dimension being sub-selected. *Output:* A set of events, each with a start time, end time, and track value. *Supported tasks:* T1, T3, T4, T5, T6, T7, T8, T10, and T13.

Q2. Get the attribute values of an event. *Input:* The event identifier. *Output:* A list of attribute names and their values. *Supported tasks:* T2 and T12.

Q3. Get data matching pattern. This is a class of queries as the calculation to do the pattern match may be intensive and will differ by application and use case. *Input:* A search range similar to Q1 and the pattern to be matched. *Output:* A a set of events, each with a start time, end time, and track value. *Supported tasks:* T9.

Q4. Get events from a range meeting conditions. This query is similar to Q1 and Q3, but additional conditions must be met for the data to be included. *Input:* Ranges as in Q1 and conditions the data must satisfy. *Output:* A set of events that meet the condition, each with a start time, end time, and track value. *Supported tasks:* T11 and T18.

Q5. Get track details. This query can be implemented in two ways, by specifying the tracks directly or by specifying events and seeking their associated tracks. *Input:* A list of track identifiers OR a list of event identifiers. *Output:* Attribute-value pairs for each track in the list OR the set of tracks associated with the input events. *Supported tasks:* T14 and T17.

Q6. Get track ordering. *Input:* a list of tracks and a condition for their ordering, for example, a track attribute. *Output:* The ordered tracks list. *Supported tasks:* T15.

3.3.2 Dependency-Related Fetch Queries

These queries focus on fetching data associated with dependencies or their induced dependency graph. Queries are often performed on nodes, which may be individual or collections of events.

Q7. Get neighbor details of a node. *Input:* a node in the dependency graph. *Output:* all neighboring nodes and their attribute-value pairs. *Supported tasks:* T21 and T24.

Q8. Find a node by attribute. The search attribute is frequently a dependency. *Input:* An attribute name and value. *Output:* Nodes matching that attribute value. *Supported tasks:* T22.

Q9. Calculate graph metrics. The exact query depends on the graph metric. *Input:* None. *Output:* Metrics of the dependency graph such as height and number of nodes. *Supported tasks:* T26.

Q10. Get graph by attributes. This query generates a graph based on a set of attributes. These attributes can affect connectivity or membership (e.g., creating a subgraph based on a provided range). *Input:* List of attributes and attribute ranges. *Output:* The constructed dependency graph. *Supported tasks:* T20, T23, and T25.

3.3.3 Update queries.

These queries update the data. They each return a boolean value indicating whether the update was successfully executed.

Q11. Add annotation. *Input:* Time and track value indicating location and content (text) of the annotation. *Supported tasks:* T28.

Q12. Update event. *Input:* The event identifier and attribute-value pair to be updated. *Supported tasks:* T27.

Q13. Remove track. This query marks a list of tracks for omission in subsequent queries. *Input:* A list of track identifiers. *Supported tasks:* T16.

4 DISCUSSION AND CONCLUSION

We presented a visualization task taxonomy for Gantt charts, an intuitive and popular visual idiom for displaying interdependent event sequences. Our motivation in creating this taxonomy is to support visualization design when using Gantt charts, especially in cases where large data in terms of number of events or tracks necessitate interactions or additional views beyond the core chart. This taxonomy serves as both a language for describing tasks and a collection of design strategies found in the literature.

To further aid both design and research with regards to Gantt charts, we paired our task taxonomy with an analysis of the data queries required to draw them. Careful consideration of both the scaling of the representation, i.e., through the visual and interaction design, and the scaling of the data management to support these queries, are important for designing and implementing effective visualizations to handle the demands of large-scale data produced in areas like manufacturing and computing.

Despite the prevalence of domain applications that can generate billions of events [20], we note that most visualizations in our survey did not reach that scale in their demonstration, with the largest shown datasets limited to millions of events. Thus, there may be tasks not yet represented in the literature which thus do not appear in our taxonomy. Limitations in managing the data for the the common tasks may be hindering creation of additional strategies as using systems in the large scale regime is cumbersome. Our list of data queries can serve as a basis for choosing and developing data management strategies to scale up interactivity in Gantt charts, allowing for further research and observation into how people explore large-scale Gantt data.

REFERENCES

[1] L. Adhianto, S. Banerjee, M. Fagan, M. Krentel, G. Marin, J. Mellor-Crummey, and N. R. Tallent. Hpctoolkit: Tools for performance analysis of optimized parallel programs. *Concurrency and Computation: Prac. & Exp.*, 22(6):685–701, 2010. 2, 3, 4

[2] W. Aigner, S. Miksch, H. Schumann, and C. Tominski. *Visualization of time-oriented data*, vol. 4. Springer, 2011. 2

[3] P. André, M. L. Wilson, A. Russell, D. A. Smith, A. Owens, and M. Schraefel. Continuum: designing timelines for hierarchies, relationships and scale. In *Proc. ACM UIST*, pp. 101–110, 2007. 2

[4] D. Antweiler, D. Sessler, M. Rossknecht, B. Abb, S. Ginzel, and J. Kohlhammer. Uncovering chains of infections through spatiotemporal and visual analysis of covid-19 contact traces. *Comp. & Graph.*, 2022. 1, 2, 3, 4

[5] R. Bell, A. D. Malony, and S. Shende. Paraprof: A portable, extensible, and scalable tool for parallel performance profile analysis. In *Euro. Conf. on Parallel Proc.*, pp. 17–26, 2003. 2, 4

[6] J. Dąbrowski, T. Bartkowiak, P. Wierzchowski, and D. Dąbrowski. Manufacturing line-level root cause analysis and bottleneck detection using the digital shadow concept and cloud computing. In *Int'l Sci.-Tech. Conf. MANUFACTURING*, pp. 98–111. Springer, 2024. 3, 4

[7] T. Davidson, E. Wall, and J. Mace. A qualitative interview study of distributed tracing visualisation: A characterisation of challenges and opportunities. *IEEE Trans. on Vis. & Comp. Graphics*, 2023. 1

[8] J. C. De Kergommeaux, B. Stein, and P.-E. Bernard. Pajé, an interactive visualization tool for tuning multi-threaded parallel applications. *Parallel Computing*, 26(10):1253–1274, 2000. 3

[9] W. De Pauw and S. Heisig. Zinsight: A visual and analytic environment for exploring large event traces. In *Proc. Int. Symp. on Soft. Vis.*, pp. 143–152, 2010. 2, 3, 4

[10] S. Devaux, F. Bouali, and G. Venturini. Datatube4log: A visual tool for mining multi-threaded software logs. In *2014 18th International Conference on Information Visualisation*, pp. 189–195. IEEE, 2014. 2

[11] A. Drebes, A. Pop, K. Heydemann, A. Cohen, and N. Drach-Temam. Aftermath: A graphical tool for performance analysis and debugging of fine-grained task-parallel programs and run-time systems. In *Workshop on Prog. Issues for Het. Multicores*, 2014. 3, 4

[12] N. Ezzati-Jivan and M. R. Dagenais. Multi-scale navigation of large trace data: A survey. *Concurrency and Computation: Prac. & Exp.*, 29(10):e4068, 2017. 2

[13] J. A. Fails, A. Karlson, L. Shahamat, and B. Shneiderman. A visual interface for multivariate temporal data: Finding patterns of events across multiple histories. In *IEEE VAST*, pp. 167–174, 2006. 3

[14] Y. Guo, S. Guo, Z. Jin, S. Kaul, D. Gotz, and N. Cao. A survey on visual analysis of event sequence data. *IEEE Trans. on Vis. & Comp. Graphics*, 2021. 2

[15] S. Gupta, M. Dumas, M. J. McGuffin, and T. Kapler. Movementslicer: Better gantt charts for visualizing behaviors and meetings in movement data. In *IEEE Pacific Vis. Symp.*, pp. 168–175, 2016. 2, 3, 4

[16] B. Haugen, S. Richmond, J. Kurzak, C. A. Steed, and J. Dongarra. Visualizing execution traces with task dependencies. In *Proc. Workshop on Visual Perf. Analysis*, pp. 1–8, 2015. 3, 4

[17] M. T. Heath and J. A. Etheridge. Visualizing the performance of parallel programs. *IEEE Software*, 8(5):29–39, 1991. 3

[18] V. N. Hirakata, M. L. R. Oppermann, V. K. Genro, and A. J. Reichelt. Exploring the gantt chart as a tool to highlight double report in case series published during the first wave of the covid-19 pandemic. *Systematic Reviews*, 11(1):1–7, 2022. 3

[19] K. E. Isaacs, P.-T. Bremer, I. Jusufi, T. Gamblin, A. Bhatele, M. Schulz, and B. Hamann. Combing the communication hairball: Visualizing parallel execution traces using logical time. *IEEE Trans. on Vis. & Comp. Graphics*, 20(12):2349–2358, 2014. 2, 3, 4

[20] K. E. Isaacs, A. Giménez, I. Jusufi, T. Gamblin, A. Bhatele, M. Schulz, B. Hamann, and P.-T. Bremer. State of the art of performance visualization. *EuroVis STARs*, 2014. 1, 2, 4

[21] J. Jo, J. Huh, J. Park, B. Kim, and J. Seo. Livegantt: Interactively visualizing a large manufacturing schedule. *IEEE Trans. on Vis. & Comp. Graphics*, 20(12):2329–2338, 2014. 1, 2, 3, 4

[22] J. Kaldor, J. Mace, M. Bejda, E. Gao, W. Kuropatwa, J. O'Neill, K. W. Ong, B. Schaller, P. Shan, B. Viscomi, et al. Canopy: An end-to-end performance tracing and analysis system. In *Proc. Symp. on Operating Sys. Principles*, pp. 34–50, 2017. 2, 3

[23] H. Koike, T. Takada, and T. Masui. Visualinda: a framework for visualizing parallel linda programs. In *Proc. IEEE Symp. on Vis. Lang.*, pp. 174–178, 1997. 3, 4

[24] Z. Liu and J. Heer. The effects of interactive latency on exploratory visual analysis. *IEEE Trans. on Vis. & Comp. Graphics*, 20(12):2122–2131, 2014. 1

[25] S. Luz and M. Masoodian. Improving focus and context awareness in interactive visualization of time lines. *Pro. HCI*, pp. 72–80, 2010. 2, 3

[26] M. Meyer, F. Beck, and S. Lohmann. Visual monitoring of process runs: An application study for stored procedures. In *IEEE Pacific Vis. Symp.*, pp. 160–167, 2016. 2

[27] W. E. Nagel, M. Weber, A. Arnold, K. Solchenbach, and H.-C. Hoppe. Vampir: Visualization and analysis of mpi resources. *Supercomputer*, 63(FZJ-2015-02424):69–80, 1996. 2

[28] L. L. Nesi, V. G. Pinto, L. M. Schnorr, and A. Legrand. Summarizing task-based applications behavior over many nodes through progression clustering. In *Int. Conf. on Parallel, Dist. & Network-Based Proc.*, pp. 35–42, 2023. 2, 3

[29] Y. Peiris, C.-M. Barth, E. M. Huang, and J. Bernard. A data-centric methodology and task typology for time-stamped event sequences. In *2022 IEEE Evaluation and Beyond-Methodological Approaches for Visualization (BELIV)*, pp. 66–76. IEEE, 2022. 2

[30] V. Pillet, J. Labarta, T. Cortes, and S. Girona. Paraver: A tool to visualize and analyze parallel code. In *Proc. WoTUG-18*, vol. 44 number 1, pp. 17–31, 1995. 2, 3, 4

[31] V. G. Pinto, L. Stanisic, A. Legrand, L. M. Schnorr, S. Thibault, and V. Danjean. Analyzing dynamic task-based applications on hybrid platforms: An agile scripting approach. In *Proc. Workshop on Visual Perf. Analysis*, pp. 17–24, 2016. 2, 3

[32] C. Plaisant, B. Shneiderman, and R. Mushlin. An information architecture to support the visualization of personal histories. *Info. Pro. & Management*, 34(5):581–597, 1998. 3

[33] S. A. Sakin, A. Bigelow, R. Tohid, C. Scully-Allison, C. Scheidegger, S. R. Brandt, C. Taylor, K. A. Huck, H. Kaiser, and K. E. Isaacs. Traveler: Navigating task parallel traces for performance analysis. *IEEE Trans. on Vis. & Comp. Graphics*, 29(1):788–797, 2022. 2, 3, 4

[34] C. Schaubschlaeger, D. Kranzlmüller, and J. Volkert. Event-based program analysis with dewiz. In *Proceedings of the Fifth International Workshop on Automated Debugging (AADEBUG 2003)*, September 2003. 2

[35] L. M. Schnorr, G. Huard, and P. O. Navaux. Triva: Interactive 3d visualization for performance analysis of parallel applications. *Future Generation Comp. Sys.*, 26(3):348 – 358, 2010. 2, 3, 4

[36] G. Shurkhovetskyy, N. Andrienko, G. Andrienko, and G. Fuchs. Data abstraction for visualizing large time series. In *Computer Graphics Forum*, vol. 37 number 1, pp. 125–144, 2018. 2

[37] Y. Sun, Y. Zhang, A. Mosallaei, M. D. Shah, C. Dunne, and D. Kaeli. Daisen: A framework for visualizing detailed gpu execution. In *Computer Graphics Forum*, vol. 40 number 3, pp. 239–250, 2021. 3, 4

[38] B. Topol, J. T. Stasko, and V. Sunderam. Pvanim: A tool for visualization in network computing environments. *Concurrency: Prac. & Exp.*, 10(14):1197–1222, 1998. 2

[39] M. Tory, S. Staub-French, D. Huang, Y.-L. Chang, C. Swindells, and R. Pottinger. Comparative visualization of construction schedules. *Automation in Construction*, 29:68–82, 2013. 1, 3, 4

[40] J. M. Wilson. Gantt charts: A centenary appreciation. *European Journal of Operational Research*, 149(2):430–437, 2003. 1

[41] J. Yan, S. Sarukkai, and P. Mehra. Performance measurement, visualization and modeling of parallel and distributed programs using the aims toolkit. *Software: Prac. & Exp.*, 25(4):429–461, 1995. 3

[42] O. Zaki, E. Lusk, W. Gropp, and D. Swider. Toward scalable performance visualization with jumpshot. *Int. J. HPC Applications*, 13(3):277–288, 1999. 2, 3, 4